Contents

Chapter 1

Introduction

In spite of the advances accomplished throughout the last decades by a number of research teams, Automatic Speech Recognition (ASR) is still a challenging and difficult task. In particular, recognition systems based on Hidden Markov Models (HMMs) are effective under many circumstances, but do suffer from some major limitations that limit applicability of ASR technology in real-world environments. Attempts were made to overcome these limitations with the adoption of Artificial Neural Networks (ANNs) as an alternative paradigm for ASR, but ANN were unsuccessful in dealing with long time-sequences of speech signals. Between the end of the Eighties and the beginning of the Nineties, some researchers began exploring a new research area, by combining HMMs and ANNs within a single, hybrid architecture. The goal in hybrid systems for ASR is to take advantage from the properties of both HMMs and ANNs, improving flexibility, hardware implementability and recognition performance.

The present chapter introduces a formal definition of the ASR problem in terms of statistical pattern recognition, pointing out its major difficulties and challenging new directions. Then, a review of basic HMM concepts, i.e. the state-of-the-art paradigm for acoustic modeling, is provided. Limitations of standard HMMs for ASR are outlined. ANNs and hybrid ANN/HMM paradigms are then briefly considered as a viable alternative to tackle such limitations. Motivations and overview of the present Thesis are finally given.

1.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) (Rabiner and Juang, 1993; Deller et al., 1993; Lee et al., 1996; Junqua and Haton, 1996; De Mori, 1998) has been the most investigated topic in speech processing along the last few decades. Broadly speaking, it can be formulated as the problem of *recognizing* (not *understanding*) the words from a given dictionary uttered by a speaker, relying only on the information contained in the uttered *speech signal* and on prior knowledge on the problem domain. In the Fifties, when the research on ASR began, many researchers believed that the incoming new computer technologies would have made ASR rather an easy task. Unfortunately, a few decades later, we are realizing that the conjecture was false. ASR emerged to be a very hard problem, and nowadays many difficulties and open questions are far from being solved, in spite of the efforts of a number of long-termed research groups throughout the world. Difficulties are related to increasing dictionary size (e.g. more than $50,000$ words), continuous speech recognition versus isolately uttered words, number and vocal characteristics of speakers in Speaker Independent (SI) recognizers (versus simpler Speaker Dependent (SD) systems, capable to recognize only speech uttered by the speaker whose acoustic material was used to train the recognizer itself), spontaneous speech phenomena (um's, ah's, false starts, out-of-vocabulary words, etc.), robustness to environmental conditions (noise and distortion over the channel, multiple microphones spread throughout the room to allow hands-free dictation, etc.), and so on. Most of these problems arise when ASR technology is moved from the laboratory, where simulation experiments often allow for excellent recognition performance, to real-world conditions, where a dramatic degradation in performance is far from being an exception.

Nonetheless, ASR has resulted highly effective in a variety of applicative scenarios: dictation, access to databases, human-machine interface, access to remote automatic services on the telephone line, and control of machines.

The ASR problem can be formulated as a statistical *classification* problem, according to classical pattern recognition (Duda and Hart, 1973; Fukunaga, 1990). Once the *classes* have been defined as sequences W of allowable words from a given dictionary, a parametric representation of the speech signal has been chosen (e.g. a sequence of acoustic feature vectors X), and a *Maximum a Posteriori* (MAP) criterion has been adopted, the classification problem can be stated as finding the sequence of words $\tilde{W}$ which maximizes the quantity $Pr(W \mid X)$. The latter is usually factorized using Bayes' theorem (Duda and Hart, 1973) as:

$$Pr(W \mid X) = \frac{Pr(X \mid W)Pr(W)}{Pr(X)}. \tag{1.1}$$

Given an acoustic observation sequence X, the efforts on the maximization of $Pr(W \mid X)$ can be moved to the search for the class $\tilde{W}$ which maximizes the numerator of the right-hand side of Equation (1.1), i.e. $Pr(X \mid W)Pr(W)$. The quantity $Pr(W)$, usually referred to as the *language model* (LM) (Rabiner et al., 1996; Jelinek, 1998) depends on high-level constraints and linguistic knowledge about allowed word strings for the specific task. The quantity $Pr(X \mid W)$ is known as the *acoustic model*. It describes the statistics of sequences of parametrized acoustic observations in the feature space given the corresponding uttered words. Hidden Markov models (HMMs) (Rabiner, 1989; Huang et al., 1990) are the most popular (parametric) model at the acoustic level. A brief review of HMMs is presented in Section 1.2. Although HMMs are an effective approach to the problem of acoustic modeling in ASR, allowing for good recognition performance under many circumstances, they also suffer from some limitations. These limitations, discussed at the end of Section

1.2, are the rationale behind the research for different paradigms. In particular, ANNs and ANN/HMM hybrids (Section 1.3) are taken into consideration in this Thesis.

Since the ultimate goal of ASR systems is obtaining good recognition performance on the field, a constant, strong emphasis has been put in the speech community on significant evaluation criteria. In the following, experimental results are reported in terms of recognition rate on specific test databases. The most popular evaluation criterion is the *Word Error Rate* (WER), that is defined as

$$WER = 100(Ins + Del + Sub)/N_{words}\% \tag{1.2}$$

where N_{words} is the total number of words in the uttered text, and the number of errors of the recognizer is expressed counting out word insertions (Ins), deletions (Del) or substitutions (Sub), respectively. Its complementary, the *Word Recognition Rate* (WRR)[1], defined as $WRR = (100 - WER)\%$, will also be used.

1.2 Hidden Markov Models

A HMM is a pair of stochastic processes: a *hidden* Markov chain and an *observable* process which is a probabilistic function of the states of the former. This means that observable events in the real world (e.g., acoustic observations) are modeled with (possibly continuous) probability distributions, that are the observable part of the model, associated with individual states of a discrete-time, first-order Markovian process. In general, the latter is not ergodic. The semantics of the model (conceptual correspondence with physical phenomena) is usually encapsulated in the hidden part: for instance, in ASR a HMM can be used to model a word in the task-dependent vocabulary, where each state of the hidden part represents a phoneme (or sub-phonetical unit), whereas the observable part accounts for the statistical characteristics of the corresponding acoustic events in a given feature space (e.g. sampled acoustic signal, represented in a proper way).

More precisely, a HMM is defined by:

1. A set S of Q states, $S = \{S_1, \ldots, S_Q\}$, which are the distinct values that the discrete, hidden stochastic process can take.

2. An *initial state* probability distribution, i.e. $\pi = \{Pr(S_i \mid t = 0), S_i \in S\}$, where t is a discrete time index.

3. A probability distribution that characterizes the allowed transitions between states, that is $\mathbf{a}_{ij} = \{Pr(S_j \text{ at time } t \mid S_i \text{ at time } t-1), S_i \in S, S_j \in S\}$ where the *transition probabilities* $\mathbf{a}_{ij}$ are assumed to be independent of time t.

[1]The WRR is known also as the *unit accuracy*.

4. An *observation* or *feature* space $\mathcal{X}$, which is a discrete or continuous universe of all possible observable events (usually a subset of $\mathbf{R}^d$, where d is the dimensionality of the observations).

5. A set of probability distributions (referred to as *emission* or *output* probabilities) that describes the statistical properties of the observations for each state of the model: $\mathbf{b_x} = \{b_i(\mathbf{x}) = Pr(\mathbf{x} \mid S_i), S_i \in S, \mathbf{x} \in \mathcal{X}\}$.

HMMs represent a learning paradigm, in the sense that examples of the event that is to be modeled can be collected and used in conjunction with a training algorithm in order to *learn* proper estimates of π, $\mathbf{a}$ and $\mathbf{b_x}$. The most popular of such algorithms are the *Forward-Backward* (or *Baum-Welch*) (Rabiner, 1989) and the Viterbi (Rabiner, 1989) algorithms. Whenever continuous emission probabilities are considered, both of them are based on the general Maximum Likelihood (ML) criterion, i.e. they aim at maximizing the probability of the samples given the model at hand. In particular, the Viterbi algorithm concentrates only on the most alike path throughout all the possible sequences of states in the model.

These algorithms belong to the class of *unsupervised learning* techniques, since they perform unsupervised parameter estimation of the probability distributions without requiring any prior labeling of individual observations (within the sequences used for training) as belonging to specific states.

Once training has been accomplished, the HMM can be used for *decoding* or *recognition*. Two different instances occur according to the specific task. Whenever N different HMMs – corresponding to models of N different events or *classes* defined in the feature space – are used (e.g. in ASR), decoding (classification) means assigning each new sequence of observations to the most alike model. On the contrary, when a single HMM is used, decoding (recognition) means finding out the most alike path of states within the model and assigning each individual observation to a given state within the model. If the ML criterion is used, the Forward-Backward or the Viterbi algorithms are still suitable for the recognition task.

One major distinction has to be made between *discrete* HMMs and *continuous density* HMMs (CDHMMs) (Huang et al., 1990). The former use discrete probability distributions to model the emission probabilities, i.e. they rely on the assumption of a finite alphabet of symbols in input or, equivalently, they require a *quantization* of a continuous input space, e.g. via any *clustering* (Duda and Hart, 1973) technique, into a finite size *codebook*. CDHMMs, on the other side, use continuous probability density functions (pdfs), usually referred to as *likelihoods*, to describe statistics of the acoustic features within the HMM states, and are usually best suited for very difficult ASR tasks (i.e, continuous speech dictation with large vocabularies), since they exhibit better modeling accuracy. Gaussian or mixtures of Gaussian components are the most popular and effective choices of pdfs for CDHMMs.

HMMs have been successfully applied in a variety of tasks, mainly in speech recognition. Unfortunately, standard HMMs in conjunction with the above-mentioned training and decoding algorithms suffer from some major intrinsic limitations (see for instance (Bourlard and Morgan, 1994a)):

1. HMMs do require a pre-defined feature space, i.e. acoustic features are extracted *a priori* from the speech signal according to conventional digital signal processing techniques, without any feedback from the recognizer on which features might, or might not, be relevant to the recognition process.

2. Possible correlations among individual features are not taken into account, since concrete implementations of CDHMMs do assume statistical independence among input features, i.e. diagonal covariance matrices.

3. Standard CDHMMs, trained with Forward-Backward or Viterbi algorithms, present poor discriminative power among different models, since they are based on the ML criterion, which is itself non-discriminative.

4. The classical HMMs rely on strong assumptions on the statistical properties of the phenomenon at hand. In particular:

   (a) the stochastic processes involved are modeled by first-order Markov chains;

   (b) the parametric form of the probability density functions that represent the emission probabilities associated with all states is heavily constraining.

5. The number of parameters in HMMs do strongly limit their implementability in hardware.

6. In HMMs attention is focused on learning[2] the training set and not on generalization capabilities; in particular, no regularization theory for HMMs has been developed so far.

The latter fact is particularly critical in real-world, noisy applications where robustness of the ASR system can be expressed, in terms of machine learning theory, as the generalization capability of the recognizer itself. Given these limitations, the use of ANNs with their discriminative training, their capability to perform non-parametric estimation over whole sequences of patterns, and their limited number of parameters (which allows for effective development of neural microchips) appear definitely promising.

[2]It is worth noticing that learning in standard HMMs is *memory-based*, since Gaussian kernels are placed over the feature space in order to "cover" the training sample as much as possible; memory-based architectures are known to yield limited generalization performance whenever the test set significantly differs from the training sample.

1.3 ANNs and ANN/HMM Hybrids

Starting from the late Eighties, many researchers began to use Artificial Neural Networks (ANNs) for ASR. Neural nets were expected to carry out the recognition task (e.g. classification of phonemes or words) when discriminatively trained on acoustic features. Milestones in this respect are (Waibel, 1989; Waibel et al., 1989a; Waibel et al., 1989b; Gori et al., 1989; Haffner et al., 1989; Bengio, 1990; Frasconi et al., 1990; Cosi et al., 1990; Hampshire and Waibel, 1990; Tebelskis et al., 1991; Robinson and Fallside, 1991; Cosi et al., 1992; Robinson, 1992; Bengio et al., 1992a; Robinson, 1994), among the others. Lippmann (Lippmann, 1989) wrote a comprehensive survey of the state of the art in connectionist speech recognition at the end of the Eighties. The topic was of crucial interest in classic conferences and workshops traditionally dedicated to speech recognition and even in specialized workshops. To take the temporal dependencies typical of speech signals into account, two major classes of neural networks were proposed, namely *Time-Delay Neural Network*, and *Recurrent Neural Networks*. These connectionist architectures are surveyed in Chapter 2, where a detailed review of ANNs for ASR is presented. Reviews of ANNs for ASR can be found, for instance, in (Trentin et al., 1998; Trentin et al., 1996a).

In spite of their ability to classify short-time acoustic-phonetic unis, such as individual phonemes, ANNs failed as a general framework for ASR, especially with long sequences of acoustic observations like those required in order to represent words from a dictionary or whole sentences. This is mainly due to the lack of ability to model long-term dependencies in ANNs, even when recurrent architectures are considered. The theoretical motivations underlying this problem were well analyzed in (Bengio et al., 1994). In the early Nineties this fact led to the idea of combining HMMs and ANNs within a single, novel model, broadly known as *hybrid ANN/HMM* (Franzini et al., 1990; Levin, 1990; Bourlard and Wellekens, 1990; Morgan and Bourlard, 1990; Niles and Silverman, 1990; Haffner et al., 1991; Tebelskis et al., 1991; Bengio et al., 1992b). A recently published survey paper (Wilinski et al., 1998) investigates the "border" between ANNs and Markovian models (HMMs for ASR and *Markov Random Fields* for image processing), summarizing approaches where ANNs emulate Markov models and *vice-versa*, and reporting hybrid architectures. Comprehensive surveys of hybrid ANN/HMM systems for ASR can be found in (Trentin and Gori, 1999a; Trentin and Gori, 1999b; Trentin and Gori, 2000). Hybrid models turn out to be effective in applicative scenarios different from ASR, too: see (Trentin and Cattoni, 1997; Trentin and Cattoni, 1999a; Trentin and Cattoni, 1999b) for examples. The hybrid paradigm relies on maintaining an underlying HMM structure, capable of modeling long-term dependencies, with the integration of ANNs, which provide non-parametric universal approximation, probability estimation, discriminative training algorithms, fewer parameters to estimate than those usually required in standard HMMs, efficient computation of outputs at recognition time, and efficient hardware implementability.

Different hybrid architectures and training/decoding algorithms have been proposed in the literature, according to the nature of the ASR task, to the type of HMM, or to the specific role of the ANN within the system. The hybrid approach often allowed for significant improvements in performance with respect to standard HMM-based approaches to difficult ASR tasks. Chapter 4 is a survey of hybrid ANN/HMM systems for ASR, putting together approaches and techniques from a highly specialistic and non-homogeneous literature, and categorizing different architectures within several broad classes. These categories are summarized in Table 1.1, with pointers to the respective Sections.

Category	**Brief description**	**Section**
Early attempts	Approaches that rely on ANN architectures that attempt to emulate HMMs.	4.1.1
ANNs to estimate the HMM state-posterior probabilities	A probabilistic interpretation of the ANN outputs is given, e.g. ANNs perform an estimate of the posterior probability of CDHMM states given the acoustic observations.	4.1.2
Global optimization	Introduction of a training scheme aimed at the optimization of a "global" criterion function, defined at the whole-system (i.e., ANN and HMM simultaneously) level.	4.1.3
Networks as vector quantizers for discrete HMMs	Unsupervised ANNs are used to perform a *quantization* in the acoustic feature space for discrete HMMs.	4.1.4
Other approaches	Hybrid systems based on particular combination techniques between ANNs and HMMs, not belonging to any of the previous categories, and often focused on specific tasks.	4.1.5

Table 1.1: Categorization of hybrid models according to their architectures and nature, along with brief description of characterizing features and pointer to corresponding Section in Chapter 4.

Although state-of-the-art hybrid models seem to yield a gain in performance with respect to standard HMM recognizers under many circumstances, several open problems in ASR still remain far from being solved. Some of them and a sketch of challenging future issues are discussed at the end of Chapter 4. In particular, the latter points out that:

1. In most hybrid architectures ANNs play only a marginal role, while the kernel of the recognizer still relies on a standard HMM. The latter basically brings into the resulting system the usual limitations of HMMs that were mentioned

in Section 1.2. Examples are: connectionist vector quantization for discrete HMMs, neural acoustic front-ends for CDHMMs (e.g. for *speaker normalization* (Trentin et al., 1996b; Trentin and Giuliani, 1997; Trentin and Giuliani, 2001)), or ANNs for rescoring the *N-best* hypothesis yielded by a HMM.

2. In the major cases of connectionist probability estimation for an underlying HMM, namely in Bourlard and Morgan's paradigm (Bourlard and Morgan, 1994a) and its derivates, the level of integration between the two paradigms is deeper and the respective roles potentially allow to overcome the limitations of both, but the lack for a globally-defined, mathematically motivated training scheme strongly weakens the theoretical framework and substantially reduces the improvement in performance on the field.

3. Up to date, no hybrid systems stressed the generalization capabilities of ANNs to tackle the problem of ASR in noisy environments under different, adverse conditions. *A fortiori*, no explicit mechanism was introduced within hybrid algorithms in order to improve generalization and, thus, robustness to noise.

The development of viable, theoretically motivated as well as practically effective solutions to these open problems, along the promising directions pointed out by existing hybrid approaches, forms the motivation for the present Thesis.

1.4 Overview of the Thesis

The Thesis is structured as follows:

- **Chapter 2** reviews major concepts of connectionist models and reports "classical" applications to ASR problems, such as recognition of isolated words or sub-word units (e.g., phonemes), feature extraction, etc. Focus is on conventional architectures (static and dynamic), including fundamental training algorithms based on gradient-descent (for feedforward networks) or on ML estimation (for competitive nets).

- **Chapter 3** introduces a novel, gradient-based training scheme that increases learning and generalization capabilities in feed-forward and recurrent nets. The algorithm, which may be qualitatively described in terms of the introduction of *trainable amplitudes* of the ANN activation functions, modifies the search paths in the weight space toward deeper minima that may improve generalization. The technique, that can also be interpreted as a soft weight-grouping procedure, enforces the robustness of the model to noise.

- **Chapter 4** surveys a number of different ANN/HMM hybrid paradigms for ASR, categorizing them according to Table 1.1, reviewing their architectures and training procedures, and reporting experimental results from literature.

Advantages and limitations of such systems are pointed out and compared. The need for a new framework, capable to combine their advantages while tackling their drawbacks, emerges as a consequence.

- **Chapter 5** develops novel, gradient-ascent global training algorithms for a ANN/HMM hybrid where a connectionist estimate of emission probabilities (or related quantities) associated with the states of the HMM is carried out. Experiments on SI, connected Italian digits recognition are presented, allowing for a comparison among the different algorithms and, in turn, of the latter ones with standard CDHMMs.

- **Chapter 6** combines the training schemes introduced in Chapters 3 and 5, yielding a hybrid model for ASR characterized by a global optimization strategy where an explicit mechanism is included to improve generalization, i.e. robustness to noise and environmental changes. Experiments on real-world data validate the approach through a direct comparison with a CDHMM.

- **Chapter 7** summarizes the main concepts developed throughout the Thesis, discusses some future issues of theoretical/practical interest, and draws conclusive remarks.

Finally, the **Appendix** reports some mathematical results and the proofs of theorems that are used throughout the Thesis.

Chapter 2

Neural Networks for Speech Recognition

After having described in the previous chapter the ASR problem and a probabilistic model for ASR, namely the HMM, this chapter introduces a new type of model based on Artificial Neural Networks (ANNs). ANNs are learning machines trainable from examples, loosely inspired from principles of data processing in the brain. ANNs may be used as nonparametric discriminant functions in classification tasks, as multiple regression models, or as "universal" function approximators. Due to their properties, they were applied in a variety of ASR tasks, either as a stand-alone paradigm or in combination with other models.

This chapter introduces basic principles with a large set of pointers to the literature, where further details about theory and applications to speech can be found. First of all, the Simple Linear Perceptron is presented (Section 2.1), while its natural extension to the Multilayer Perceptron and the *backpropagation* algorithm are described in Section 2.2. Gradient-descent training algorithms are developed in a step-by-step manner, since the understanding of their schemes will be crucial in the calculations involved by the gradient-ascent techniques for the novel ANN/HMM hybrid presented in Chapter 5.

Then, ANNs with unsupervised learning are introduced (Section 2.3), with an emphasis on Competitive Neural Nets (CNNs): their training algorithm is an instance of the maximum-likelihood estimation for mixtures of Gaussian components, exactly like in standard HMMs. In addition, CNNs are the basic model underlying several connectionist vector-quantization modules that may be combined with discrete HMMs (Chapter 4). Networks for dealing explicitly with time sequences, namely Time Delay ANNs and Recurrent nets, are presented in Section 2.4. Applications to ASR and, more in general, to speech processing, are described in various sections.

2.1 Simple Linear Perceptrons and the Widrow-Hoff Algorithm

For the sake of simplicity, as well as for historical reasons, we introduce first a feed-forward connectionist model trainable with supervision: for each input sample, during training the ANN is provided with the corresponding desired output. This means that input vectors are presented at the input of the network one at a time. Based on the input, the network computes an output vector.

Let us consider the training set $T = \{(\mathbf{x}_k, \mathbf{y}_k) \mid k = 1, ..., N\}$, where N input samples $\mathbf{x}_1, ..., \mathbf{x}_N$ have been previously labeled with their corresponding desired (target) outputs $\mathbf{y}_1, ..., \mathbf{y}_N$. The aim is to build a model capable to compute for each input of the training data an output close to the desired one, according to some optimality criterion. A particular family of such models is represented by *Simple Linear Perceptrons* (see figure 2.1).

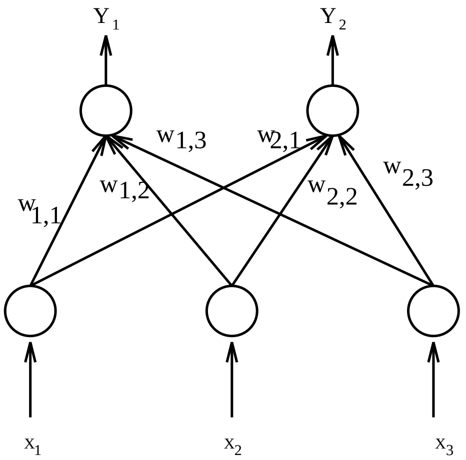

Figure 2.1: *Simple linear Perceptron.*

A simple linear Perceptron has a set of input units S_I acting as placeholders for the components of the current input vector. Each input unit is fully connected with all the neurons of the output units set S_O, via direct links (connections, or *synapses*) characterized by a specific connection weight (*synaptic strength*), denoted with $w_{i,j}$ in the figure (in the following figures, connection weights are not explicitly shown). An input signal is propagated forward along the connections, and multiplied by the corresponding connection weight. All the incoming weighted signals to a given output unit are summed together, to form the input to the unit itself. The unit reacts by producing an activation response which is equal to its input. This model is usually referred to as a *layered network*, with the obvious meaning that units are arranged into subsequent layers: the computation proceeds from one layer to the one that immediately follows in the bottom-up order, but never in a lateral or backward manner. For this reason the network is called "feed forward".

The family of models considered in this section fits the training data as summarized by the following equation (written for the generic i-th output component):

$$\hat{y}_i(\mathbf{x}) = \sum_{j \in S_I} w_{ij} x_j. \tag{2.1}$$

This is a homogeneous linear transformation, but an additive bias can be easily added to the model. Simple Linear Perceptrons can thus be seen as linear regression models or linear discriminators for classification. Nevertheless, the way these networks *learn* from the training set is quite general, and can be extended to the study of other, more complex, ANN architectures.

Once a training set T and a simple linear Perceptron are given, with the obvious assumption that the number of input and output units matches the dimensionality of the input and target vectors, respectively, the *learning problem* can be stated as the search for the network weights $\mathbf{w}$ which allow to optimally fit the data, according to a certain criterion. The latter is usually expressed as a functional of the training data (and of the model) that represents a gain to be maximized or a loss (or risk) to be minimized. A sound and common choice for the criterion function is the sum of squared differences between target outputs and actual outputs:

$$C = \frac{1}{2} \sum_{n=1}^{N} \sum_{k \in S_O} (y_{kn} - \hat{y}_{kn})^2 \tag{2.2}$$

where y_{kn} is k-th component of the n-th target, $\hat{y}_{kn}$ is the output of k-th output unit when the network is presented with the n-th input vector, and the multiplicative factor $\frac{1}{2}$ is introduced for computational convenience, as we shall see below. The minimization of (2.2) is known as the *least square criterion*. A general and broadly used optimization technique for the minimization of expression (2.2) is the gradient descent method. Most network training algorithms are based on it. Although it is not guaranteed that the approach will eventually reach the global minimum of the criterion function, these techniques often produce practically useful behavior. It is worth noting that the estimation of the coefficients of a linear transformation which will minimize a least squares criterion on a training set using gradient descent is an instance of the well-known *Widrow-Hoff algorithm* (Duda and Hart, 1973), used to perform discriminant linear analysis for classification. We will see that gradient descent iterative algorithms are used also for training more complicated ANN architectures.

The iterative approach can be developed in the following way. Given the vector $\mathbf{w}$ of the connection weights at a certain iterative step of the algorithm, with an arbitrary choice at the initial step, a new weight vector $\mathbf{w}'$ is computed according to the expression:

$$\mathbf{w}' = \mathbf{w} + \Delta\mathbf{w} \tag{2.3}$$

where $\Delta\mathbf{w}$ denotes the weight-change vector, that in turn satisfies the following relation:

$$\Delta\mathbf{w} \propto -\nabla_{\mathbf{w}} C. \tag{2.4}$$

In expression (2.4) the symbol $\nabla_{\mathbf{w}} C$ represents the gradient vector of the error functional computed with respect to the weights $\mathbf{w}$. Rewriting equation (2.4) in an explicit form, the gradient descent algorithm takes a move away from the current weights along the direction opposite to the gradient, by prescribing:

$$\Delta\mathbf{w} = -\eta\nabla_{\mathbf{w}} C \tag{2.5}$$

or, equivalently, if individual variations of weights are expressed in terms of the corresponding partial derivatives:

$$\Delta w_{ij} = -\eta\frac{\partial C}{\partial w_{ij}} \tag{2.6}$$

where η is the so-called *learning rate*, a small positive constant that controls the rate of convergence of the algorithm, i.e. the length of the step away from the current point in weight space. Larger values of η allow for faster convergence; however, if the value is too large, the algorithm can diverge (C increases). The optimal value of η depends on the largest eigenvalue of the Hessian matrix of second order derivatives of the cost function with respect to the weights (see next section). The above iterative step is repeated, starting from the initial weights and evolving along a trajectory in the weight space, until convergence to a minimum of the criterion function is reached, or when a stopping criterion is met. The latter can be related to the availability of computational resources, or can be expressed in terms of an estimation of generalization performance at each iteration (*early-stopping*). From a theoretical point of view, the gradient descent technique can always reach a minimum of the criterion function (if the learning rate is reduced during training, that is if $\sum_{t=1}^{\infty}\eta_t = \infty$ and $\sum_{t=1}^{\infty}\eta_t^2 < \infty$), but it may get stuck in local minima without reaching the global minimum. Starting a few times the algorithm from different initial weights is a common practice. The problem of local minima has been widely investigated: (Baldi and Hornik, 1989) discusses the case of linear networks, while (Gori and Tesi, 1992) faces the same problem for the more general case of nonlinear networks.

In practical cases, instead of computing the gradient of function (2.2), with a summation over all the training samples (*batch mode*), an *on-line* variant of the algorithm is used, by taking the partial derivatives of the following error functional:

$$C = \frac{1}{2}\sum_{k\in S_O}(y_k - \hat{y}_k)^2 \tag{2.7}$$

which is locally computed over each training pair $(\mathbf{x}, \mathbf{y})$, and the weight changes are still determined using equation (2.6) every time the network is presented with a new input vector. This approach, also known as *stochastic gradient descent*, proves to be quite effective when the number of training patterns is large (> hundreds). The whole training set is repeatedly fed through the net for a certain number of *epochs*. An important consequence of stochastic gradient is the ability of incremental training, since the network can potentially learn more from new examples without requiring a complete, batch retraining. This is an historical reason why the terms *learning* or *adaptation*, rather than *optimization*, are often used to describe this process. Stochastic gradient descent has also proven to be more effective than standard gradient in many cases, particularly for its generalization ability.

In order to obtain an explicit form for equation (2.6) suitable for an efficient implementation, the partial derivatives of equation (2.7) with respect to a generic weight w_{ij} have to be computed. We can write:

$$\begin{aligned} \frac{\partial C}{\partial w_{ij}} &= \frac{1}{2} \sum_{k \in S_O} \frac{\partial (y_k - \hat{y}_k)^2}{\partial w_{ij}} \\ &= \frac{1}{2} \frac{\partial (y_i - \hat{y}_i)^2}{\partial w_{ij}} \\ &= -(y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial w_{ij}}. \end{aligned} \tag{2.8}$$

Recalling equation (2.1) we can now write the partial derivative of the output $\hat{y}_i$ from the i-th output unit with respect to the weight w_{ij} as:

$$\begin{aligned} \frac{\partial \hat{y}_i}{\partial w_{ij}} &= \frac{\partial f_i(x_i)}{\partial w_{ij}} \\ &= \frac{\partial \sum_{l \in S_I} w_{il} x_l}{\partial w_{ij}} \\ &= x_j \end{aligned} \tag{2.9}$$

where x_i denotes the input to output unit i. Substituting equations (2.8) and (2.9) into equation (2.6) we obtain:

$$\Delta w_{ij} = \eta (y_i - \hat{y}_i) x_j \tag{2.10}$$

that gives the weight changes for each connection weight of the network in terms of the product of the component of the current input to the connection, by the difference between the desired and the actual output from the corresponding neuron. If the difference is small, i.e. the network response resembles the target value, then only a small adjustment of the weight is performed. Larger differences imply larger modifications of the synaptic strengths.

If we define the quantity δ_i as:

$$\delta_i = y_i - \hat{y}_i \tag{2.11}$$

we can rewrite equation (2.10) in the form:

$$\Delta w_{ij} = \eta \delta_i x_j \tag{2.12}$$

which is known as the *delta rule* and provides a more compact representation of the learning rule (2.10).

Equation (2.12) can be written as the difference of two terms as follows:

$$\Delta w_{ij} = \eta y_i x_j - \eta \hat{y}_i x_j \tag{2.13}$$

where the first term is a typical example of *Hebbian learning*, which expresses the tendency to strengthen the synaptic weight whenever the input and the desired output of a neuron (pre- and post-synaptic activations) are high together. The second term, on the contrary, is referred to as *anti-Hebbian learning*, with the intuitive opposite meaning.

A direct and important extension to the simple linear Perceptron model uses a more general activation functions $f_i()$ of the output units. The output $\hat{y}_i$ of the i-th unit will be written $\hat{y}_i = f_i(a_i)$, where a_i is the input to the unit, usually $a_i = \sum_{l \in S_I} w_{il} x_l$. Typically, $f_i()$ is either the identity (usually referred to as the "linear activation function") or a squashing non-linearity, such as the hyperbolic tangent $f_i(a_i) = \tanh(a_i)$ or the sigmoid $f_i(a_i) = 1/(1 + e^{-a_i})$. The only constraint that must be imposed, if the gradient descent technique is still used, is that the activation functions must be continuous and differentiable functions. In this general case equation (2.9) must be rewritten in the following form:

$$\begin{aligned} \frac{\partial \hat{y}_i}{\partial w_{ij}} &= \frac{\partial f_i(a_i)}{\partial w_{ij}} \\ &= \frac{\partial f_i(a_i)}{\partial a_i} \frac{\partial a_i}{\partial w_{ij}} \\ &= f_i'(a_i) \frac{\partial \sum_{l \in S_I} w_{il} x_l}{\partial w_{ij}} \\ &= f_i'(a_i) x_j \end{aligned} \tag{2.14}$$

where $f_i'(a_i)$ is the derivative of the activation function computed over the current input a_i to i-th unit, and equation (2.6) becomes:

$$\Delta w_{ij} = \eta (y_i - \hat{y}_i) f_i'(a_i) x_j \tag{2.15}$$

that easily reduces to equation (2.10) in the linear (identity) case. It is still possible to define the quantity δ_i to be:

$$\delta_i = (y_i - \hat{y}_i) f_i'(a_i) \tag{2.16}$$

that, in turn, is a straight generalization of equation (2.11) and includes the latter as a special case, leading to:

$$\Delta w_{ij} = \eta \delta_i x_j \tag{2.17}$$

that is identical to Equation (2.12). In the next section the above concepts are further developed in the context of multi-layered networks and, surprisingly enough, the calculation of the learning algorithm for these models will lead to expressions substantially identical to the delta rule.

2.2 Multilayer Perceptrons and the Backpropagation Algorithm

The most popular neural network architecture is the *Multilayer Perceptron* (MLP). This is an extension of the simple linear Perceptron with additional layers of units, called *hidden layers.* In the hidden layers an internal, intermediate representation of data is formed. A MLP is fed by an input vector and subsequent computations are passed from layer to layer in the usual *feedforward* manner. This produces an output vector of values of the units of the last (output) layer of the network. Activation functions associated to hidden and output units can be linear or non-linear, and can be different for different units. Input units still act as placeholders for the components of the current input vector. A training algorithm estimates a set of weights to be assigned to the connections between each pair of units belonging to adjacent layers, in order to optimize a training criterion.

Training is generally supervised. Let us consider a labeled training set $T = \{(\mathbf{x}_k, \mathbf{y}_k) \mid k = 1, ..., N\}$. The learning problem for MLPs is to find the weights that result in a (generally non-linear) model that best fits the training data, given a certain criterion function. The most common choice for the criterion is the sum of squared differences between target and actual outputs, just as in the case of simple linear Perceptrons. We will illustrate an on-line gradient descent technique (Werbos, 1974; LeCun, 1986; Rumelhart et al., 1986a) to minimize the cost function:

$$C = \frac{1}{2} \sum_{n \in S_O} (y_n - \hat{y}_n)^2 \tag{2.18}$$

computed after the presentation of a certain input pattern $\mathbf{x}$ associated with a desired output vector $\mathbf{y}$ of a layered network with a set of input units S_I, hidden units S_H, and output units S_O. An example of such a network is shown in figure 2.2. Extension to more hidden layers is straightforward.

The learning algorithm is similar to the one for simple linear Perceptrons. The weight change Δw_{ij} of the connection strength between the j-th hidden unit and the i-th *output* unit is computed as follows:

$$\Delta w_{ij} = -\eta \frac{\partial C}{\partial w_{ij}}. \tag{2.19}$$

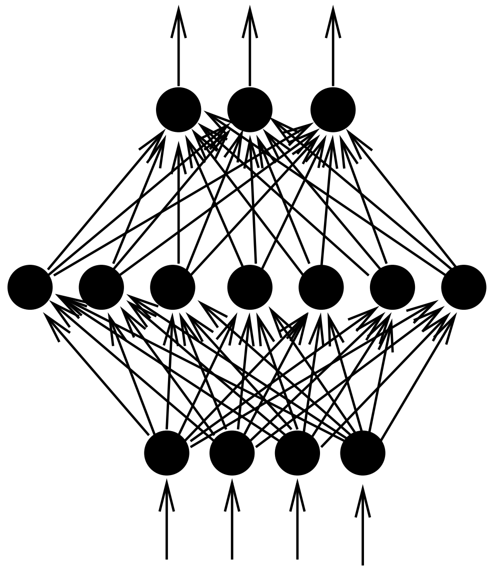

Figure 2.2: *Multi-layer Perceptron.*

The calculation begins as in equation (2.8), by writing:

$$\begin{aligned}\frac{\partial C}{\partial w_{ij}} &= \frac{1}{2}\sum_{n\in S_O}\frac{\partial (y_n-\hat{y}_n)^2}{\partial w_{ij}} \\ &= \frac{1}{2}\frac{\partial (y_i-\hat{y}_i)^2}{\partial w_{ij}} \\ &= -(y_i-\hat{y}_i)\frac{\partial \hat{y}_i}{\partial w_{ij}}\end{aligned} \tag{2.20}$$

where the term $\frac{\partial \hat{y}_i}{\partial w_{ij}}$ can be calculated as:

$$\begin{aligned}\frac{\partial \hat{y}_i}{\partial w_{ij}} &= \frac{\partial f_i(a_i)}{\partial w_{ij}} \\ &= \frac{\partial f_i(a_i)}{\partial a_i}\frac{\partial a_i}{\partial w_{ij}} \\ &= f_i'(a_i)\frac{\partial \sum_{l\in S_H} w_{il}\hat{y}_l}{\partial w_{ij}} \\ &= f_i'(a_i)\hat{y}_j\end{aligned} \tag{2.21}$$

where $f_i'(a_i)$ still denotes the derivative of the activation function computed over the current input a_i to the i-th unit. We define:

$$\delta_i = (y_i-\hat{y}_i)f_i'(a_i) \tag{2.22}$$

and we write:

$$\Delta w_{ij} = \eta \delta_i \hat{y}_j \tag{2.23}$$

as in the delta rule for simple Perceptrons. Let us now consider a unit of the *hidden layer*, say j. The variation of a weight w_{jk} associated to a connection between the k-th input unit and the j-th hidden unit

$$\Delta w_{jk} = -\eta \frac{\partial C}{\partial w_{jk}} \tag{2.24}$$

can be computed considering that

$$\begin{aligned} \frac{\partial C}{\partial w_{jk}} &= \frac{1}{2} \sum_{n \in S_O} \frac{\partial (y_n - \hat{y}_n)^2}{\partial w_{jk}} \\ &= -\sum_{n \in S_O} (y_n - \hat{y}_n) \frac{\partial \hat{y}_n}{\partial w_{jk}} \end{aligned} \tag{2.25}$$

which resembles the (2.20), but takes into consideration the influence that changes in weight w_{jk} have on the whole set of output units, i.e. each hidden unit affects all of the outputs in some way. Now we can write:

$$\begin{aligned} \frac{\partial \hat{y}_n}{\partial w_{jk}} &= \frac{\partial f_n(a_n)}{\partial w_{jk}} \\ &= \frac{\partial f_n(a_n)}{\partial a_n} \frac{\partial a_n}{\partial w_{jk}} \\ &= f_n'(a_n) \frac{\partial a_n}{\partial w_{jk}} \end{aligned} \tag{2.26}$$

where a further development of term $\frac{\partial a_n}{\partial w_{jk}}$ leads to:

$$\begin{aligned} \frac{\partial a_n}{\partial w_{jk}} &= \frac{\partial \sum_{l \in S_H} w_{nl} \hat{y}_l}{\partial w_{jk}} \\ &= \sum_{l \in S_H} w_{nl} \frac{\partial \hat{y}_l}{\partial w_{jk}} \\ &= w_{nj} \frac{\partial \hat{y}_j}{\partial w_{jk}} \end{aligned} \tag{2.27}$$

and, in turn:

$$\frac{\partial \hat{y}_j}{\partial w_{jk}} = \frac{\partial f_j(a_j)}{\partial w_{jk}} \tag{2.28}$$

$$
\begin{aligned}
&= \frac{\partial f_j(a_j)}{\partial a_j}\frac{\partial a_j}{\partial w_{jk}} \\
&= f'_j(a_j)\frac{\partial \sum_{m\in S_I} w_{jm}x_m}{\partial w_{jk}} \\
&= f'_j(a_j)x_k.
\end{aligned}
$$

Substituting eqs. (2.25), (2.26), (2.27) and (2.28) into equation (2.24) we finally obtain:

$$
\begin{aligned}
\Delta w_{jk} &= \eta \sum_{n\in S_O} \left[(y_n - \hat{y}_n)f'_n(a_n)w_{nj}\right] f'_j(a_j)x_k \qquad (2.29)\\
&= \eta\{\sum_{n\in S_O} [w_{nj}(y_n - \hat{y}_n)f'_n(a_n)]\}f'_j(a_j)x_k \\
&= \eta(\sum_{n\in S_O} w_{nj}\delta_n)f'_j(a_j)x_k
\end{aligned}
$$

where δ_n is defined as in equation (2.16), for each output unit. For a generic unit j in the hidden layer we can similarly define:

$$\delta_j = (\sum_{n\in S_O} w_{nj}\delta_n)f'_j(a_j) \qquad (2.30)$$

that brings to evidence the fact that the deltas for hidden units can no longer be computed as a direct function of the difference between the desired target output and the actual network output, because the units themselves have an indirect influence on it. The deltas can rather be expressed as a weighted sum of the deltas already computed at the upper layer, thus back-propagating deltas from one layer to the next (lower) one. In summary, the current input vector is propagated forward through the network, obtaining an output. The error, that is a squared difference between the obtained and the target output, is computed and used to determine deltas at the output layer. Weight changes for the weights of the output layer are immediately computable, according to the delta rule. These deltas are then propagated backward to the hidden layer, allowing for the computation of weight changes by application of equation (2.29). This is usually referred to as error backpropagation, and the resulting algorithm is known as the *Back-Propagation* (BP) algorithm (Rumelhart et al., 1986a). The compact formulation of the latter can be written as:

$$\Delta w_{jk} = \eta\delta_j x_k \qquad (2.31)$$

that, when used together with equation (2.23), is called *Generalized Delta Rule.* If more than one hidden layer is present, the calculation proceeds exactly in the same way, back-propagating the deltas down to the next (lower) hidden layer and yielding a rule identical to equation (2.31). The BP algorithm is the most popular training technique for neural networks, and it has been successfully used in a wide range of applications. The ability of MLPs to model a linear or non-linear discriminant function for classification tasks, or a regressor in multivariate spaces, makes the feed-forward connectionist approach a suitable method for many applications in speech processing. Important examples will be given in the next sections, especially concerning the integration of MLPs in an ASR system based on hidden Markov models. MLPs have been used for recognizing phonemes from portions of spectrograms (Bengio, 1996), for grapheme-to-phoneme translation (Sejnowski and Rosenberg, 1986; Sejnowski and Rosenberg, 1987), and for mapping acoustic features obtained with a microphone array with the same features obtained with a close mouth (Che et al., 1994).

An alternative feed-forward connectionist paradigm to the MLP is represented by the so-called *Radial Basis Functions* (RBF) networks (Poggio and Girosi, 1989; Poggio and Girosi, 1990; Niranjan and Fallside, 1990). Within such models, sigmoids are basically replaced by Gaussian activations, thus realizing a superposition of local-basis kernels or, equivalently, a mixture of Gaussian components (whenever a probabilistic interpretation of the RBF outputs is given). Standard clustering algorithms (Duda and Hart, 1973) and gradient descent optimization techniques may be applied to their training process. We will briefly discuss the application of a RBF architecture to a novel ANN/HMM hybrid framework in Section 5.4.1. From a general point of view, RBF did not receive significant attention from researchers in the speech community, mainly due to their difficulty in dealing with high-dimensional feature spaces (e.g., like the typical acoustic spaces required in ASR).

2.3 Unsupervised Learning

Let us now consider the *training set* $T = \{\mathbf{x}_k \mid k = 1, ..., N\}$, where N input unlabeled samples $\mathbf{x}_1, ..., \mathbf{x}_N$ have been collected. As no class labeling is available, the learning task is aimed at discovering, in an unsupervised manner, inherent properties of the data and the way they are distributed in the feature space. Although the problem of learning a classification scheme from an unlabeled training set is very difficult, statistical pattern recognition techniques for unsupervised parametric mixture estimation and data clustering have been successfully proposed and applied in the last decades. Recently, different connectionist approaches to the problem have been introduced, often strictly related to the statistical ones, but sometimes indeed novel. They usually rely on statistical and geometrical properties of the given data.

Some of the most popular unsupervised networks, namely self-supervised (auto-associative) nets and competitive networks will be introduced in the following. Other unsupervised approaches to connectionist models are ART and ART2 (Carpenter and Grossberg, 1987; Carpenter and Grossberg, 1988), Kohonen's Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ) (Kohonen, 1989; Kohonen, 1990), and Oja's and Sanger's networks (Oja, 1982; Oja and Karhunen, 1985; Oja, 1989; Sanger, 1989), which are capable to perform on-line projection of the input data along their Principal Components.

Good results of LVQ for speech coding were obtained by (Kohonen et al., 1988). In (Mäntysalo et al., 1992) LVQ is used for phoneme-level segmentation of speech signal. Excellent examples of the application of LVQ in ASR can be found in (McDermott and Katagiri, 1989; McDermott and Katagiri, 1991), where multi-category, shift-tolerant phoneme recognition experiments are reported.

2.3.1 Self-supervised networks

Self-supervised networks (Cottrell et al., 1987) are standard feed-forward networks, such as MLPs, trained in a supervised way (for instance using the BP algorithm discussed in section 2.2) on the training set $T' = \{(\mathbf{x}_k, \mathbf{x}_k) \mid k = 1, ..., N\}$ obtained from the original training set T by pairing each pattern with itself. The major constraint on the topology of the network is that one of the hidden layers is built up of a number of hidden units that must be significantly lower than the dimensionality of the input space. An illustrative example is given in figure 2.3, in which the same training pattern $\mathbf{x}$ is applied at the input as well as at the output.

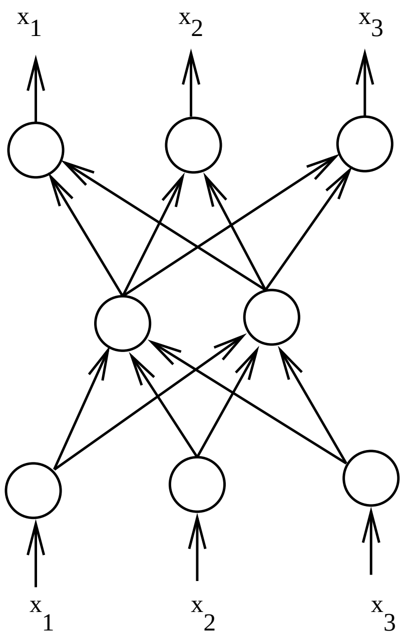

Figure 2.3: *Self-supervised network: the same training patter* $\mathbf{x}$ *is applied at the input as well as at the output.*

Roughly speaking, the network is trained to reconstruct on its output units each input pattern which is fed into the input units, by forcing the pattern to be coded with a limited set of features through the low-dimensional hidden layer. This forces a low-dimensional representation of the information contained in the input data that is sufficient to reconstruct the original patterns. During the training step, the network learns low dimensional representations of the input vectors, preserving as much relevant information as possible. Once training is accomplished, the hidden layer can be used as an output layer to produce compressed patterns. This technique is an effective way to perform feature extraction and, more generally, dimensionality reduction of a given signal representation. In (Elman and Zipser, 1988) it is applied to the extraction of acoustic features from spectrograms. When there is only one hidden layer, this technique has been shown to be equivalent to the popular Principal Component Analysis (Bourlard and Kamp, 1988; Baldi and Hornik, 1989), that is the projection of the data along the space spanned by the principal eigenvectors (with the largest eigenvalues) of the data covariance matrix. The reconstruction error can be used as an indication of the similarity of the input pattern with the patterns of the training set. It has been successfully used in (Gori et al., 1996) for a speaker verification task.

2.3.2 Competitive neural networks

Competitive neural networks (CNNs) (Hertz et al., 1991) are one-layer feed-forward models, the dynamics of which forms the basis for several other unsupervised connectionist architectures (Grossberg, 1976; Lippmann, 1987; Winters and Rose, 1989). In addition, they are a kind of a reference model for several unsupervised connectionist vector-quantization models that are combined with discrete HMMs (see Chapter 4). These ANNs are called *competitive* because each unit competes with all the others for the classification of each input pattern: the latter is indeed assigned to the *winner unit*, which is the closest (according to a given distance measure) to the input itself, i.e. the most representative within a certain set of *prototypes*. This is strictly related to the concept of *clustering* (Duda and Hart, 1973), where each unit represents the centroid (or mean, or codeword) of one of the clusters. The propagation of the input vector through the network consists in a projection of the pattern onto the weights of the connections entering each output unit. Connection weights are assumed to represent the components of the corresponding centroid. The aim of the forward propagation is to establish the closest centroid to the input vector, i.e. the winner unit. During training, a simple weight update rule is used to move the winner centroid toward the novel pattern, so that the components of the centroid represent a sort of moving average of the input patterns belonging to the corresponding cluster. This is basically an on-line version of some partitioning clustering algorithms (Duda and Hart, 1973). In the following, we will derive it as a consequence of *maximum-likelihood estimation* of the parameters of a mixture of Gaussians under certain assumptions. This fact emphasizes the strict relationship with standard HMMs training techniques, e.g. the Baum-Welch algorithm. The simple, basic CNN can then be easily extended by introducing dynamic allocation of units, or combining it with various supervised techniques. A parallel implementation is also straightforward.

Let us consider again the training set $T = \{\mathbf{x}_k \mid k = 1, ..., N\}$, where N input samples $\mathbf{x}_1, ..., \mathbf{x}_N$ are now supposed to be drawn from the *finite mixture* density

$$p(\mathbf{x} \mid \mathbf{\Theta}) = \sum_{i=1}^{C} \Pi_i p_i(\mathbf{x} \mid \theta_i) \tag{2.32}$$

where the parametric form of the component densities $p_i(), i = 1, ..., C$ is assumed to be known, as well as the *mixing parameters* $\Pi_1, ..., \Pi_C$ (*a priori* probabilities of the components) and $\mathbf{\Theta} = (\theta_1, ..., \theta_C)$ is the vector of all parameters associated to each component density. In the present setup we want to use the unlabeled training samples to estimate the parameters $\mathbf{\Theta}$. In classical pattern recognition this is an unsupervised parametric estimation problem. The following discussion will introduce a connectionist approach to the same problem, leading to the formulation of an unsupervised learning rule for CNNs that is consistent (under certain assumptions) with the above mentioned statistical estimation.

Assuming the samples are independently drawn from $p(\mathbf{x} \mid \mathbf{\Theta})$, the likelihood of the observed data T given a certain choice of parameters $\mathbf{\Theta}$ can be written as:

$$p(T \mid \mathbf{\Theta}) = \prod_{j=1}^{N} p(\mathbf{x}_j \mid \mathbf{\Theta}). \tag{2.33}$$

Maximum-likelihood estimation techniques search for the parameters $\mathbf{\Theta}$ that maximize expression (2.33), or equivalently the *log-likelihood*, as:

$$l(\mathbf{\Theta}) = \log p(T \mid \mathbf{\Theta}) = \sum_{j=1}^{N} \log p(\mathbf{x}_j \mid \mathbf{\Theta}). \tag{2.34}$$

Since the logarithm is a monotonic increasing function, parameters that maximize expressions (2.33) and (2.34) are the same. If a certain parameter vector $\mathbf{\Theta}'$ maximizes $l(\mathbf{\Theta})$ then it has to satisfy the following necessary, but not sufficient, condition:

$$\nabla_{\theta_i'} l(\mathbf{\Theta}') = \mathbf{0} \quad i = 1, ..., C \tag{2.35}$$

where the operator $\nabla_{\theta_i'}$ denotes the gradient vector computed with respect to the parameters θ_i', and $\mathbf{0}$ is the vector with all components equal to zero. In other words, we are looking for the zeros of the gradient of the log-likelihood. Substituting equation (2.32) into equation (2.34) and setting the latter equal to zero we can write:

$$\begin{aligned} \nabla_{\theta_i'} l(\mathbf{\Theta}') &= \sum_{j=1}^{N} \nabla_{\theta_i'} \log p(\mathbf{x}_j \mid \mathbf{\Theta}') \\ &= \sum_{j=1}^{N} \nabla_{\theta_i'} \log \sum_{i=1}^{C} \Pi_i p_i(\mathbf{x}_j \mid \theta_i') \\ &= \sum_{j=1}^{N} \frac{1}{p(\mathbf{x}_j \mid \mathbf{\Theta}')} \nabla_{\theta_i'} \Pi_i p_i(\mathbf{x}_j \mid \theta_i') \\ &= \mathbf{0}. \end{aligned} \tag{2.36}$$

Using Bayes' Theorem we have:

$$Pr(i \mid \mathbf{x}_j, \theta'_i) = \frac{\Pi_i p_i(\mathbf{x}_j \mid \theta'_i)}{p(\mathbf{x}_j \mid \mathbf{\Theta})} \tag{2.37}$$

where $Pr(i \mid \mathbf{x}_j, \theta'_i)$ is the *a posteriori* probability of class i given the observation $\mathbf{x}_j$ and the parameters θ'_i. Equation (2.36) can thus be rewritten as:

$$\begin{aligned} \nabla_{\theta'_i} l(\mathbf{\Theta}') &= \sum_{j=1}^{N} \frac{Pr(i \mid \mathbf{x}_j, \theta'_i)}{\Pi_i p_i(\mathbf{x}_j \mid \theta'_i)} \nabla_{\theta'_i} \Pi_i p_i(\mathbf{x}_j \mid \theta'_i) \\ &= \sum_{j=1}^{N} Pr(i \mid \mathbf{x}_j, \theta'_i) \nabla_{\theta'_i} \log \Pi_i p_i(\mathbf{x}_j \mid \theta'_i) \\ &= \mathbf{0}. \end{aligned} \tag{2.38}$$

In the following, we will concentrate our attention on the case in which the component densities of the mixture are multivariate Normal distributions. In this case

$$p_i(\mathbf{x}_j \mid \theta_i) = (2\pi)^{-\frac{d}{2}} \mid \Sigma_i \mid^{-\frac{1}{2}} e^{\{-\frac{1}{2}(\mathbf{x}_j - \mu_i)^t \Sigma_i^{-1} (\mathbf{x}_j - \mu_i)\}}, \tag{2.39}$$

where d is the dimension of the feature space, t denotes the transposition of a matrix, and the parameters to be estimated for the i-th component density are its mean vector and its covariance matrix, that is to say:

$$\theta_i = (\mu_i, \mathbf{\Sigma}_i). \tag{2.40}$$

Substituting equation (2.39) into equation (2.38) we can write the gradient of the log-likelihood for the case of normal component densities as:

$$\begin{aligned} \nabla_{\theta_i} l(\mathbf{\Theta}) &= \sum_{j=1}^{N} Pr(i \mid \mathbf{x}_j, \theta_i) \nabla_{\theta_i} \{\log \Pi_i (2\pi)^{-\frac{d}{2}} \mid \Sigma_i \mid^{-\frac{1}{2}} \\ &\quad -\frac{1}{2}(\mathbf{x}_j - \mu_i)^t \Sigma_i^{-1} (\mathbf{x}_j - \mu_i)\}. \end{aligned} \tag{2.41}$$

Suppose now that the only unknown parameters to be estimated are the mean vectors of the Gaussian distributions, e.g. the covariances are supposed to be known in advance. There are practical situations, for example in data clustering, in which the estimation of the means is sufficient; furthermore, this assumption simplifies the following mathematics, but the extension to the more general case of unknown covariances is rather simple. By setting $\mathbf{\Theta} = (\mu_1, ..., \mu_C)$, equation (2.41) reduces to:

$$\nabla_{\theta_i} l(\mathbf{\Theta}) = \sum_{j=1}^{N} \{Pr(i \mid \mathbf{x}_j, \mu_i) \times \tag{2.42}$$

$$\times \nabla_{\mu_i}[-\frac{1}{2}(\mathbf{x}_j - \mu_i)^t \Sigma_i^{-1}(\mathbf{x}_j - \mu_i)]\}$$
$$= \sum_{j=1}^{N} Pr(i \mid \mathbf{x}_j, \mu_i)\Sigma_i^{-1}(\mathbf{x}_j - \mu_i).$$

Again, we are looking for the parameters $\mathbf{\Theta}' = (\mu_1', ..., \mu_C')$ that maximize the log-likelihood, i.e., that correspond to a zero of its gradient. From equation (2.43), setting $\nabla_{\theta_i} l(\mathbf{\Theta}') = \mathbf{0}$ allows us to write:

$$\sum_{j=1}^{N} Pr(i \mid \mathbf{x}_j, \mu_i')\mathbf{x}_j = \sum_{j=1}^{N} Pr(i \mid \mathbf{x}_j, \mu_i')\mu_i' \tag{2.43}$$

which finally leads to the following central equation:

$$\mu_i' = \frac{\sum_{j=1}^{N} Pr(i|\mathbf{x}_j, \mu_i')\mathbf{x}_j}{\sum_{j=1}^{N} Pr(i|\mathbf{x}_j, \mu_i')} \tag{2.44}$$

which shows that the maximum likelihood estimate for the i-th mean vector is a weighted average of the samples of the training set, where each sample gives a contribution that is proportional to the *estimated* probability of i-th class given the sample itself. Equation (2.44) can not be explicitly solved, but it can be put in a quite interesting and practical iterative form by making explicit the dependence of the current estimate on the number t of iterative steps that have already been accomplished:

$$\mu_i'(t+1) = \frac{\sum_{j=1}^{N} Pr(i|\mathbf{x}_j, \mu_i'(t))\mathbf{x}_j}{\sum_{j=1}^{N} Pr(i|\mathbf{x}_j, \mu_i'(t))} \tag{2.45}$$

where $\mu_i'(t)$ denotes the estimate obtained at t-th iterative step, and the corresponding value is actually used to compute the new estimate at $(t+1) - th$ step. Considering the fact that $Pr(i|\mathbf{x}_j, \mu_i'(t))$ is large when the Mahalanobis distance between $\mathbf{x}_j$ and $\mu_i'(t)$ is small, it is reasonable to estimate an approximation of $Pr(i|\mathbf{x}_j, \mu_i'(t))$ in the following way:

$$Pr(i|\mathbf{x}_j, \mu_i'(t)) \approx \begin{cases} 1 & \text{if } \mathrm{dist}(\mathbf{x}_j, \mu_i'(t)) = \min_{l=1,...,C} \mathrm{dist}(\mathbf{x}_j, \mu_l'(t)) \\ 0 & \text{otherwise} \end{cases} \tag{2.46}$$

for a given distance measure dist() (Mahalanobis distance should be used, but Euclidean distance is usually an effective choice), so expression (2.45) reduces to:

$$\mu_i'(t+1) = \frac{1}{n_i(t)} \sum_{k=1}^{n_i(t)} \mathbf{x}_k^{(i)} \tag{2.47}$$

where $n_i(t)$ is the number of training patterns estimated to belong to class i at step t, i.e. the number of patterns for which $Pr(i|\mathbf{x}_j, \mu_i'(t)) = 1$ according to equation

(2.46), and $\mathbf{x}_k^{(i)}, k = 1, ..., n_i(t)$ is the k-th of these patterns. Note that equation (2.47) is the *k-means* clustering algorithm (Duda and Hart, 1973). To switch to an incremental form for equation (2.47), consider what happens when, after the i-th mean vector has been calculated at step t over $n_i(t)$ patterns, a new pattern $\mathbf{x}_{n_i(t)+1}^{(i)}$ has to be assigned to component i. This occurs when $Pr(i|\mathbf{x}_{n_i(t)+1}^{(i)}, \mu_i'(t))$ given by relation (2.46) equals 1, while $Pr(i|\mathbf{x}_{n_i(t)+1}^{(i)}, \mu_i'(t-1))$ is 0. According to equation (2.47), the new mean vector is:

$$\begin{aligned} \mu_i'(t+1) &= \frac{1}{n_i(t)+1}(\sum_{k=1}^{n_i(t)} \mathbf{x}_k^{(i)} + \mathbf{x}_{n_i(t)+1}^{(i)}) \qquad (2.48) \\ &= \frac{n_i(t)}{n_i(t)+1}\mu_i'(t) + \frac{1}{n_i(t)+1}\mathbf{x}_{n_i(t)+1}^{(i)} \\ &= \mu_i'(t) + \frac{1}{n_i(t)+1}(\mathbf{x}_{n_i(t)+1}^{(i)} - \mu_i'(t)). \end{aligned}$$

We can thus estimate the new mean vector after presentation of the $n_i(t)+1$-th pattern in the following way:

$$\mu_i'(t+1) = \mu_i'(t) + \delta_{t+1}(\mathbf{x}_{n_i(t)+1}^{(i)} - \mu_i'(t)) \qquad (2.49)$$

where $\delta_{t+1} = \frac{1}{n_i(t)+1}$ is a vanishing quantity. Using a fixed, small value η instead of δ_{t+1} and representing each component of the i-th mean vector $\mu_i = (\mu_{i1}, ..., \mu_{id})$ with the weights $w_{i1}, ..., w_{id}$ of the corresponding connections in the CNN model, the following weight update (learning) rule is obtained:

$$\Delta w_{ij} = \eta(x_j - w_{ij}) \qquad (2.50)$$

where w_{ij} is the weight of the connection between input unit j and output unit i (the winner unit), η is the learning rate, and x_j is the j-th component of the pattern, of class i, presented to the network. When a new pattern is fed into the network, its distance is computed with respect to all the output units - using the connection weights as components of the corresponding mean vectors - and, according to relation (2.46), it is assigned to the nearest unit (mixture component). The latter is referred to as the *winner* unit. Weight update, or *learning*, is then accomplished by modifying the connection weights of the winner unit by a direct application of equation (2.50). The weights of the other units are left unchanged. This is a typical *winner-take-all* approach, where the units are in competition for novel input patterns. This is the rationale for using the name *competitive neural networks*, and justifies why they have been successfully used for speech coding as it was mentioned in the introduction to this section. A detailed description of competitive learning and its relationships with statistical mixtures can be found in (Nowlan, 1991). The way used here to derive the learning rule makes it clear that CNNs can be seen as an on-line version of the popular *k-means* clustering algorithm (Duda and Hart, 1973) and, in turn, of the Baum-Welch training scheme applied to a 1-state ergodic CDHMM where a one-to-one relationship holds between CNN output units and HMM Gaussian component of the (unique) emission probability distribution.

The same mathematical formulation can be extended to the case of unknown covariance matrices for the component densities of the mixture, and even for the case of unknown class prior probabilities (the mixing parameters). A very good review is presented in (Duda and Hart, 1973). The extension provides us iterative formulae similar to equation (2.45), for estimating the means and the other unknown parameters. This means that an on-line (incremental) connectionist version of these estimates could also be derived.

2.4 ANNs for Time Sequences

There are problems, especially in speech processing, in which samples are temporal sequences of patterns, instead of individual independent samples. To take these temporal dependencies into account, two major classes of neural networks have been proposed, namely *Time-Delay Neural Network*, and *Recurrent Neural Networks*. While the former is still a MLP with units fed by an input value and a number of its predecessors, the latter generalizes the basic feed-forward architecture of MLPs by allowing arbitrary connections between units, e.g. loops and backward connections.

2.4.1 Time-delay neural networks

Time-delay neural networks (TDNNs), also known as *tapped delay lines*, represent an effective attempt to train a static MLP for time-sequence processing, by converting the temporal sequence into a spatial sequence over corresponding units. The idea is very simple, but useful in a variety of ASR applications, mostly for phoneme recognition (Waibel, 1989; Waibel et al., 1989a; Bourlard and Morgan, 1994a). An example of a TDNN is shown in figure 2.4. The input layer has been enlarged to accept as many input patterns as the (fixed) sequence length to be processed at each time step. Input vectors enter the network from the leftmost set of input units. At each time step, inputs are shifted to the right through the unit delay line that links each set of input units to the right-adjacent one, and the next input pattern is fed into the leftmost position. The same extension can also be applied to subsequent layers, introducing a tapped-delay mechanism between hidden units (e.g. only the first block of units in the tapped line actually receives input from the previous layer), giving the ability to deal with more complicated time dependencies.

The BP algorithm can be used to train such a network. Relevant features are extracted from the original input, not based on *a priori* knowledge, but on an automatic selection during the network training. The connections holding between

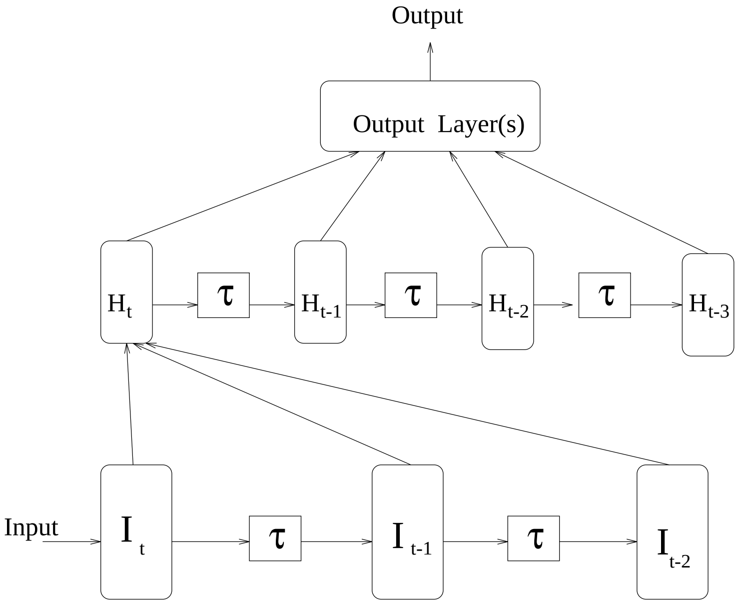

Figure 2.4: *Time-Delay Neural Network. Input is fed into the leftmost set of input units (I_t) at time t. Previous inputs (I_{t-1}, I_{t-2})are shifted to the right, with unit delays represented by boxes labeled with τ. A similar mechanism holds in the hidden layer ($H_t, ...$). An integration over time of the input sequence is carried on by the leftmost set of hidden units, while the output layer of the net integrates over time the activations of the hidden units.*

adjacent layers can also be arranged in order to provide hidden units with selective inputs from groups of units in the previous layer (receptive fields). The resulting dynamics can be interpreted as a sub-sampling from the original temporal representation of the input data, with the effect of averaging over a specific time slice. In this way, different layers of the network provide the overall processing with representations of different temporal scales.

It should be noticed that the TDNN can be seen as a particular case of the *Convolutional Neural Network* (Le Cun and Bengio, 1995) when used on "unidimensional" input (this is the case with sequences of acoustic vectors).

TDNNs have been successfully applied in ASR. Lang and Hinton (Lang and Hinton, 1988) obtained a 7.8% error rate in multi talker classification of the isolated letters "B, D, E, V", using acoustic material collected among 100 male speakers. Waibel et al. (Waibel et al., 1989b) were able to recognize isolated consonants uttered by a Japanese speaker with a low error rate (4.1%), using a combination of specialized TDNNs. A significant 1.4% error rate in vowel recognition was obtained in the same experiments.

2.4.2 Recurrent neural networks

Recurrent neural networks (RNNs) provide a powerful extension of feed-forward connectionist models by allowing to introduce connections between arbitrary pairs of units, independently from their position within the topology of the network. Self-recurrent loops of a unit onto itself, as well as backward connections to previous layers, or lateral links between units belonging to the same layer are all allowed. An example is given in figure 2.5.

RNNs behave like dynamic systems. Once fed with an input, the recurrent connections are responsible for an evolution in time of the internal state of the network. RNNs are particularly suited for sequence processing, due to their ability to keep an internal trace, or memory, of the past. This memory is combined with the current input to provide a context-dependent output.

A first type of RNNs is applied to static patterns (i.e., the input is fixed) and its dynamic converges to specific attractors. For instance, in *Boltzmann Machines* (Hinton et al., 1984) the recurrent connections are symmetric (bidirectional propagation of signal is allowed along the connection, i.e. pairs of adjacent units have an influence on each other). An interpretation of Boltzmann Machines in terms of statistical mechanics has been proposed, along with a learning algorithm based on *simulated annealing* (Kirkpatrick et al., 1983). An important instance of Boltzmann Machines is the Hopfield Net (Hopfield, 1982; Hopfield and Tank, 1986), basically constituted of a single layer of fully and symmetrically connected linear units, able to act as an auto-associative memory. Limitations of Boltzmann Machines reside in the requirement of symmetric (non-directional) recurrent connections, and in the considerable computation time required to perform the simulated annealing. In addition, although they are historically and conceptually relevant, they are practically

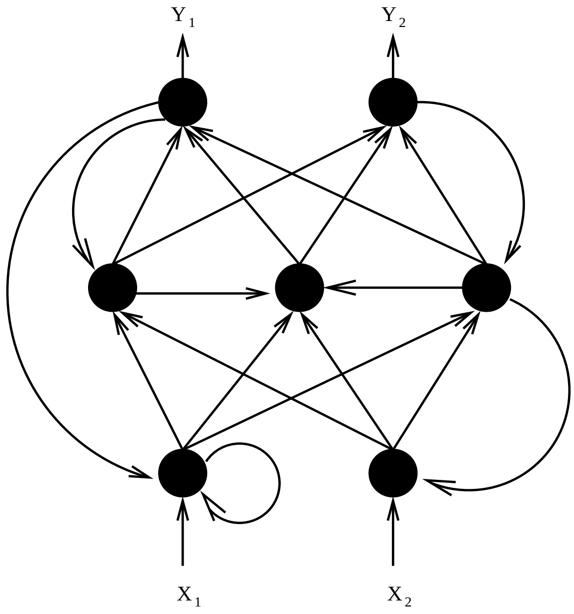

Figure 2.5: *A recurrent neural network.*

not feasible for speech and sequence processing.

Another family of RNNs is sometimes referred to as *partially recurrent nets*. In this case the basic architecture is that of a standard MLP, with the addition of a set of recurrent connections from the units in a given layer to the corresponding units of a previous layer (or in the same layer). Recurrent connections propagate the signal back to the units of one of the layers of the MLP, or to an additional *context* or *state* layer. An example is shown in figure 2.6. The units that receive signal from the recurrent connections act either as preprocessors, filtering the current (forward propagated) input with the previous signal, or as a register that keeps a memory of previous history. The weights of the recurrent connections are generally fixed and set equal to a constant, chosen in order to calibrate the amount of previous information to be taken into account. The standard BP algorithm is used to train the underlying MLP architecture, but without the computation of the full gradient on the parameters, since the effect of the past activities through recurrent connections is not taken into account. Partially recurrent nets were introduced in (Jordan, 1989; Elman, 1990; Mozer, 1993) and resulted in a wide range of applications in sequence processing (both in recognition and in generation). (Robinson and Fallside, 1991) is a significant example of application of partially recurrent nets within a speaker-independent phoneme and word recognition system.

More generally, a recurrent neural network can have arbitrary directed connections, and all weights can be learned during the training. Different approaches to the problem of training recurrent connections have been proposed in the literature in the last years, mostly based on gradient-descent techniques. Particularly remarkable are Recurrent Back Propagation (Pineda, 1989), Back-Propagation for

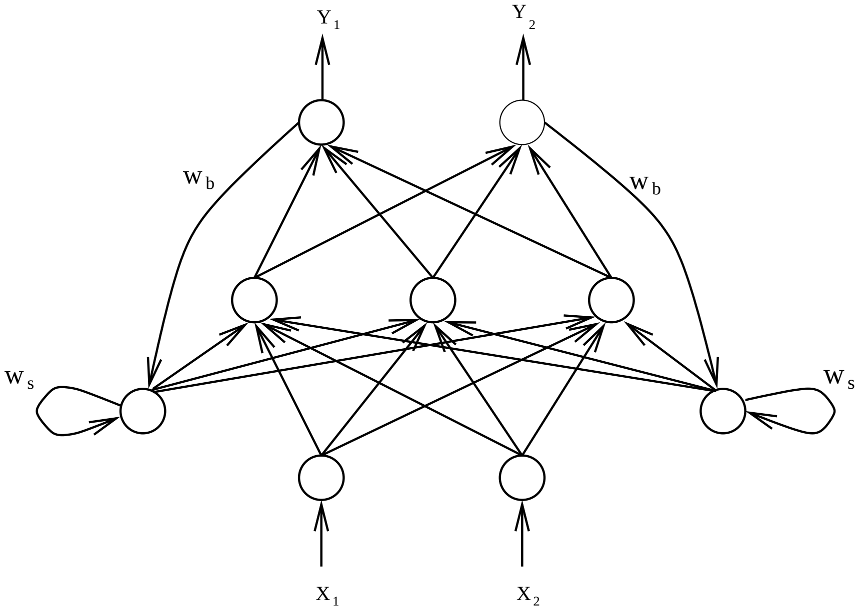

Figure 2.6: *An example of partially recurrent neural network, with two context units with backward connections from the output layer and self connections. The weights w_s and w_b are constant.*

Sequences (BPS) (Gori et al., 1989), Real Time Recurrent Learning (Williams and Zipser, 1989b; Williams and Zipser, 1989a) and Time-dependent Recurrent Back-Propagation (Werbos, 1988; Pearlmutter, 1989; Sato, 1990).

A training method for general recurrent architectures is now briefly described. It can be easily derived from the standard BP algorithm for feed-forward networks. In spite of its apparent simplicity, this technique is quite effective whenever the length of the sequences to be learned is not too large, and we are willing to wait for the end of a sequence before updating parameters. This is often the case when a whole training set with many sequences is available, i.e. when no on-line learning is required. The algorithm is called *Back-Propagation Through Time* (BPTT) or *unfolding in time* (Minsky and Papert, 1969; Rumelhart et al., 1986a). A schematic representation of BPTT for a simple one-layer recurrent network is shown in figure 2.7.

The general idea is the following. A discrete time sequence processing task is assumed. Suppose that the activity (state) of a given unit j at time t is given by:

$$\hat{y}_j(t) = f_j(\sum_{l \in L_j} w_{jl} \hat{y}_{s_l}(t - d_l) + x_j(t-1)) \tag{2.51}$$

where we denote with w_{jl} the weight of the connection between s_l-th and j-th units, with d_l the delay (number of time steps) that affects the propagation of signal between these units, and with $x_j(t-1)$ the external input to unit j at time $t-1$ (it is assumed to be zero if no input is applied to unit j at that time). The sum over

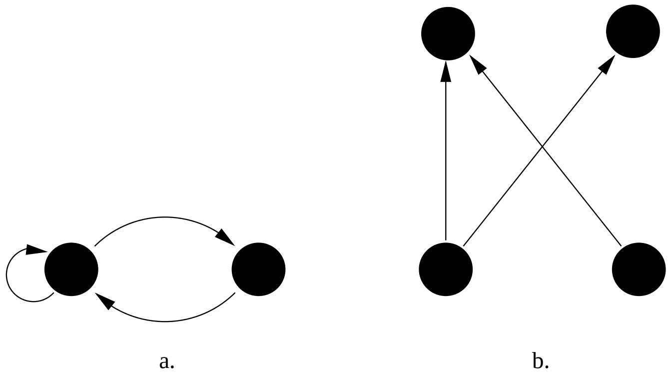

Figure 2.7: *Unfolding in time: a simple one-layer recurrent network (a) is unfolded for a length of two time steps (b).*

l is extended to all the links in L_j (whose destination unit is the j-th unit). Each unit can receive an external input or produce output (or neither) at certain time steps during the processing. The network is then unfolded by turning it into a feed-forward network, by introducing one copy of the original units for each time step. All forward connections present in the original net are replicated in the unfolded net. All recurrent connections are substituted by forward connections between a copy of the source unit at a given time step and the copy of the destination unit in the unfolded layer corresponding to the destination time step.

The unfolded net can then be trained using the BP algorithm, with the relevant constraint that all copies of a given connection w_{jk} are required to share the same weight value. This is accomplished by computing all the weight changes produced by BP at different layers, and then training using a common, unique weight change equal to the sum (or, equivalently, to the average) of the individual changes. It should be noticed that the above algorithm can deal with sequences of different lengths in the training set, by unfolding into a different number of layers for each case.

Recurrent nets have been applied in a variety of speech processing tasks. Particularly remarkable are the results obtained for phoneme recognition by Robinson et al. (Robinson and Fallside, 1988; Robinson and Fallside, 1991; Robinson, 1994), where RNNs are used as state-space machines, capable to compute an output and the next state, given the input and the current state. This results in a non-linear extension of linear control theory. A schematic representation of this RNN model is shown in figure 2.8. The approach gave the best recognition performance on phoneme recognition with the TIMIT corpus and was also used for phone probability estimation. A different recurrent network for the recognition of plosives, nasals and fricatives on the TIMIT database was described in (Bengio et al., 1991; Flammia, 1991). Robinson et al. (Robinson et al., 1993) have integrated RNNs into a hybrid ANN/HMM speech recognizer, where the networks are used to compute posterior state probabilities (see section 4.1.2). The system had slightly lower performance than some

HMM based system in the *Wall Street Journal* dictation task, but it required far less model parameters to train.

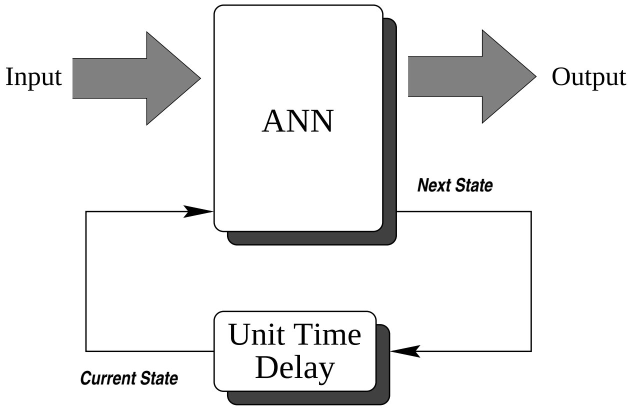

Figure 2.8: *Scheme of Robinson's dynamic net*

2.5 Concluding Remarks

Feed-forward ANNs have been applied to acoustic feature extraction, phoneme recognition, letter-to-sound translation, mapping the acoustical properties of a speech signal to the vocal tract geometry, the computation of observation posterior probabilities, etc. Recurrent neural networks have been applied to phoneme recognition, feature transformation, probability computation. Recurrent ANNs should be, in principle, more powerful than just feedforward ANNs because they allow to model history. In practice, modeling history may turn out to be a difficult task. We will return to this point in Chapter 4.

In order to combine the function approximation capabilities of ANNs with the power of HMMs to model long-term time dependencies, hybrids of ANNs and HMMs have been introduced. These structures may have the advantage of adding the discriminative power of ANNs to HMM models, or to train ANNs to perform observation transformations for improving the quality of HMM models. Due to their particular relevance concerning the topics of the present Thesis, hybrid models are separately treated in a detailed manner in Chapter 4.

Chapter 3

A Jump Into Hyperspace that Increases Noise-tolerance in Connectionist Models

Network training algorithms have heavily concentrated on the learning of connection weights. Little effort has been made to learn the *amplitude* of activation functions, which defines the range of values that the function can take. This chapter introduces novel algorithms to learn the amplitudes of non-linear activations in layered networks, without any assumption on their analytical form. Three instances of the algorithms are developed: (i) a common amplitude is shared among all nonlinear units; (ii) each layer has its own amplitude; (iii) neuron-specific amplitudes are allowed. The algorithms can also be seen as a particular double-step gradient-descent procedure, as gradient-driven adaptive learning rate schemes, or as weight-grouping techniques that appear consistent with the Bayesian framework. As a side effect, a self-pruning mechanism of redundant neurons may emerge. The introduction of a pseudo-parameter λ (i.e., the amplitude) within the training scheme modifies the search space and, accordingly, the search paths for the minimum of the criterion function within an emerging hyperspace, resulting in increased learning and generalization properties and allowing for robustness toward noise. Experimental results on function approximation, classification, and regression tasks, with synthetic and real-world speech data, validate the approach to a large extent.

3.1 From Trainable Amplitudes to improved Generalization

Network training algorithms for feedforward and recurrent models have heavily concentrated on the learning of connection weights. The most common example is the well-known *Back-Propagation* (BP) algorithm (Werbos, 1974; LeCun, 1986; Rumelhart et al., 1986b). Little effort has actually been made towards the development of methods to *learn the activation functions* associated with the units of the network. Major exceptions are represented by the use of an adaptive *bias*, i.e. a scalar which is added to the argument of sigmoidal activations and that can be trained along with the weights using BP, or by a trainable *smoothness* (Chen and Chang, 1996), controlling the shape (steep versus flat) of the sigmoids.

With the remarkable exceptions of (Yamada and Yabuta, 1992; Hu and Shao, 1992), whose contributions are briefly surveyed below, little attention has been paid so far to the *amplitude* of the activation functions (Trentin, 1999b; Trentin, 1999a), that is a measure of the (possibly open) interval in $\mathcal{R}^1$ which defines the range of values that the function can take (the image of the input domain). For instance, the amplitude of the classic sigmoid, defined as $f(x) = \frac{1}{1+\exp(-x)}$, is equal to 1, since the range of $f(x)$ is the open interval $(0, 1)$. The amplitude of the hyperbolic tangent is 2, since its values range from -1 to 1. Once the analytical form of the activations is chosen, along with its fixed amplitude, it is commonly assumed that learning the connection weights is sufficient to search the minima of the cost function that should be minimized by the learning process. From a theoretical point of view this assumption is correct: given a certain unit of the network, if an activation $f(x)$ with amplitude λ exists for which a minimum is reached in conjunction with a vector $\mathbf{w}$ of weights for the connections starting from the unit itself, then the minimum is also reached if an activation with amplitude λ' is used in conjunction with weights $\frac{\lambda}{\lambda'}\mathbf{w}$. In practice, on the contrary, the training algorithm may be unable to lead to largely different weights within a given layer, whenever weights are randomly initialized in a uniform manner over a small interval (as it is common practice) and a common learning rate is used for a certain number of training steps. At least, a different learning rate would be needed for each activation and, in turn, it should be made trainable. Section 3.4 points out that the equivalence between a network with fixed amplitude and one with adaptive amplitude is only partial, and that differences mostly concern the training scheme.

There is a number of situations where researchers have implicitly recognized the need for differentiated amplitudes, once a family of activation functions has been chosen. In similar circumstances, the introduction of trainable amplitudes would be likely to enhance the overall training process, by avoiding *a priori* choices for the amplitudes, and allowing for an adaptive scaling of the latters to the nature and dimensions of the training data at hand (herein included the scales required to improve internal representations in the hidden layers). Here is a list of some of the most common situations of this kind.

i. The range of output (target) values may definitely exceed the amplitude of the sigmoidal activations or, on the contrary, it may be far smaller. This problem is usually overcome by using linear output units, but this approach may represent a loss in terms of the potential flexibility brought by nonlinearities. In addition, linear activations do not constrain outputs within the desired range. A solution is to multiply the non-linear activations by a constant amplitude, tuned up by hand according to the training dataset (this is equivalent to the necessity of a pre-scaling procedure, pointed out by (Bhat et al., 1990)). Although this solution sometimes works, it would be much better to use an automatically trainable amplitude, capable of tuning itself to the nature of the data, especially if different behaviors are expected on different databases (i.e. the range of possible target values is not fixed and known in advance) and on-line incremental learning is to be accomplished. Interestingly enough, it usually turns out that optimal amplitudes are a bit different from the actual range of desired output values, e.g. wider sigmoids can facilitate the convergence on a given learning problem because target values do not require weight changes to be operated in proximity of the (possibly saturating, see the next point) tails of the activations.

Similar arguments also concern the nonlinearities of the hidden layers, where the best scale of values for internal representations, generally different from layer to layer and even among units belonging to the same layer, are not fixed or known *a priori*. The requirement of different scales of values for weights in different groups, e.g. in different layers, is thoroughly elucidated in (MacKay, 1992b) within a Bayesian framework: this topic is discussed in detail in Section 3.4.

ii. One major problem with learning algorithms based on the computation of partial derivatives of sigmoidal functions is the phenomenon of *saturation* of the sigmoids (Haykin, 1994). This means that input values along the tails of a sigmoid tend to produce little, often unappreciable values of the derivative of the function, thus resulting in close-to-zero derivative of the cost function. This practically halts the gradient-descent optimization process, at least making it unacceptably slow. In literature, the problem is faced by multiplying the sigmoids for a constant scalar, larger then the original amplitude, sometimes coinciding with a well-known (and often effective) "magic number", namely 1.716 (Guyon, 1991) (the input to the sigmoids of the next layer is in turn normalized by multiplying it by a factor such as 2/3). Saturation can be avoided by using a trainable amplitude instead of a fixed one, e.g. starting up with a value large enough to discourage saturation and letting it reach its own magic number, progressively learning it. Note, however, that if saturation occurs at any time during the training, the weight-updating algorithm (e.g. BP) gets stuck, but the amplitude-updating rule does not: the latter will possibly tend to yield wider activations, thus allowing for de-saturation and restart of the BP (and maybe returning to smaller amplitudes after a few iterations, once weight values have sufficiently moved the input away from the tails of the sigmoid).

iii. Feed-forward or recurrent networks which use Gaussian kernels, e.g. Radial Basis Functions (RBF) (Moody and Darken, 1989; Poggio and Girosi, 1990), may suffer from numerical instability because of the fact that Gaussians evaluated over a given feature vector often yield close-to-zero outputs and minimal differences of values over relatively large ranges in the feature space, if no explicit technique is adopted to tackle the problem. One such widely used technique relies on *scaling*, that consists in multiplying the outputs of the kernels by a constant scale factor, which is large enough with respect to the original magnitude of the outputs. Once again, the scale factor can be substituted by an adaptive amplitude of Gaussian functions, capable of scaling the kernels (possibly on a unit by unit basis) up to an "optimal" value, according to a certain criterion.

iv. In a *mixture of experts* (Jacobs et al., 1991) (or, more generally, in a system based on the combination of several connectionist models) different experts are trained to develop internal representations and produce outputs that are likely to lay in different ranges. These are generally unknown before learning takes place, since only the gating network (that has to be trained along with the experts) develops the necessary knowledge to assign regions of the feature space (and corresponding outputs) to individual networks, according to some credit assignment criterion. Our experience with mixtures of recurrent nets in some speech processing tasks (Trentin and Giuliani, 2001) (see the first example in Section 3.3.2) emphasized the fact that radically different amplitude of the units in the hidden layers are developed, starting from a common configuration, during the learning process of a mixture.

In addition, using trainable amplitudes allows *negative amplitudes*, i.e. automatically inverted activation functions (Figure 3.1 shows an example for the sigmoidal case, with an opportune offset), that can be combined within the same network with standard (non-inverted) activations, as in the window-like response functions described by (Giraud, 1996). Examples of this effect are given in Section 3.3.

This paper introduces novel algorithms to learn the amplitudes of non-linear activation functions in layered networks. No assumption is made on the analytical form of the activations, which may or may not depend upon the trainable amplitude and can possibly vary from unit to unit. This means that, for instance, alternate layers of Gaussian, sigmoid and linear activations are allowed. The proposed algorithms rely on a stochastic gradient descent (Bengio, 1996) technique for the minimization of the least-squares criterion over the training set. Generalization to other cost functions can be accomplished without affecting the major steps of the calculations.

Three instances of the algorithm are discussed and developed in Section 3.2. In the first one (referred to as "Case 1") a single value of amplitude is considered over the whole network. This means that all the units which require an adaptive

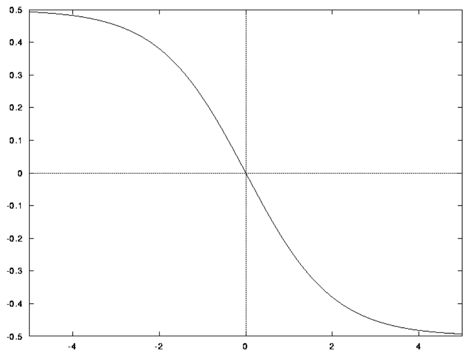

Figure 3.1: *Sigmoid with negative amplitude and offset (0.5) along the ordinate.*

amplitude do share a common value for it. This is the situation that occurs whenever a common scaling value is to be learned, e.g. in 2-layer networks with linear outputs (either Multilayer Perceptrons (MLP) or RBF). In this case the algorithm can be formulated in a particularly compact, recursive form.

The second instance ("Case 2") of the algorithm extends the previous Case by allowing for different amplitudes at different layers. Activations of units in a given layer may thus not depend on the adaptive amplitude, or share a common amplitude among them. A recursive formulation similar to that of Case 1 is developed. This Case covers networks where the scaling of nonlinear activations varies on a layer by layer basis.

Finally, the third instance ("Case 3") covers the most general situation, where individual amplitudes may be learned for each of the units in the network. The development of the algorithm is reminiscent of the BP algorithm to some extent, and its implementation is indeed facilitated by the adoption of the same *deltas* that are propagated back from layer to layer. Although Case 3 may appear to be the most flexible, the number of parameters during training results increased, with implications on the learning process and on generalization abilities, as can be noted by the experimental results presented in Section 3.3.2.

The present approach differs from previous works. (Yamada and Yabuta, 1992) propose an autotuning method for controlling the shape of activations in MLPs. All the neurons of the network are required to share the same nonlinear function. The latter is restricted to a special case of a sigmoidal function, which depends on a parameter Y_g (unique over the net) the value of which determines both the slope and the amplitude of the sigmoid at the same time (changes in amplitude occur as a side-effect of changes in shape, e.g. using decreasing values of Y_g induces a convergence of the function to linearity). (Hu and Shao, 1992) introduce an algorithm to learn *simultaneously* the parameters a and b of another special case of nonlinearity, namely a sigmoid in the form $\frac{a}{1+e^{-bx}}$, shared among the neurons of the network. The output of such activations is strictly limited to the half-range

$[0, a]$ (or $[a, 0]$ if a is negative). Finally, (Chen and Chang, 1996) describe a steepest descent algorithm for autotuning the shape of activations, including a parameter representing the "saturation level". The approach prescribes, once again, a well-defined type of a sigmoidal function to represent nonlinearities within the network. Parameters defining the shape of the function are trained on a neuron-by-neuron basis. Although the algorithm is successfully tested by realizing a direct controller, the analytical examples given in (Chen and Chang, 1996) can hardly illustrate the impact of the trainable amplitude on the capability of the overall learning machine (the shape of the sigmoid appears to play a major role), also because trainable nonlinearities are considered only at the output layer.

Anyway, in all of the above approaches, emphasis is on the *shape*, rather than on the amplitude, of sigmoids. The approach proposed in this paper focuses on the amplitude, in order to face the problems listed above, covering networks with general form of activations (linear and nonlinear, the latter meant to be of arbitrary analytical form), possibly mixed up into the same architecture, some depending on the parameter λ and some others not. An offset to shift nonlinearities along the ordinate axis is explicitly considered, reinforcing the concept of open-range outputs. Simplicity of implementation and integration within an existing training scheme for learning the weights (e.g. BP) is also valuable.

Experimental results in function approximation, classification, and regression tasks confirm the expectations to a great extent. Section 3.3 illustrates the effects of trainable amplitudes versus standard training techniques with fixed amplitudes for different network architectures. First, an artificial function approximation problem where a 3-layer *student network* attempts to learn the mapping defined by a randomly-initialized *teacher network* is analyzed. Since the teacher network has much fewer hidden units than the student net, this is also a good instance of a *self-pruning* mechanism of redundant neurons which may emerge as a side effect of the proposed algorithms.

Three examples are then given of connectionist approaches to different speech processing tasks, using real-world data. In the first instance, *speaker identification* (a classification problem) is accomplished, i.e. people recognition on the basis of their voice within a given set of talkers. Speech signals were used to train speaker-specific 3-layer models (with Gaussians in the first hidden layer, sigmoids in the second, and linear outputs), either with fixed or trainable amplitudes.

In the second instance, *speaker normalization* in the acoustic feature domain is attempted in order to increase the recognition performance of a pre-trained, continuous speech recognizer based on Continuous-density hidden Markov models, by estimating a multivariate non-linear regression between the acoustic space of a new speaker and the original training space of the recognizer. Results obtained on a 10,000 words vocabulary task, applying 2-layer MLP and mixtures of experts, are shown.

Finally, in the third instance, results obtained with a particular neural speech recognizer of isolated words in a real car environment is discussed. Also in this case, adaptive amplitudes allow for better performance, mostly due to improved generalization capabilities that increase robustness of the model to noise in the input signal.

Section 3.4 provides analytical insight into the algorithms, investigating their behavior and why they perform better than standard BP alone. A discussion is developed that shows that the algorithms can also be seen as (but not limited to) a particular double-step gradient-descent procedure, as gradient-driven adaptive learning rate schemes, or as weight-grouping techniques which appear consistent with the Bayesian framework.

3.2 Derivation of Gradient-based Algorithms for Learning Amplitudes

Let us introduce our notation. A feed-forward network with L layers is considered. Individual layers are denoted by $\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_L$, where $\mathcal{L}_0$ is the input layer (which is not counted, since its units act only as placeholders) and $\mathcal{L}_L$ is the output layer. We write $w_{i,j,l}$ to indicate the weight of the connection holding between the j-th unit in layer $\mathcal{L}_{l-1}$ and the i-th unit in layer $\mathcal{L}_l$. The actual input to the latter unit is denoted by $a_{i,l}$, and the corresponding output is $o_{i,l}$. With this notation we can write

$$a_{i,l} = \sum_{j \in \mathcal{L}_{l-1}} w_{i,j,l} o_{j,l-1} \tag{3.1}$$

where, for notational convenience, the sum over $j \in \mathcal{L}_l$ is meant to be extended to all the *indexes* of units belonging to layer $\mathcal{L}_l$. Whenever $l = 0$ or $l = L$ these are also the components of the input or output vectors, respectively. An activation function is associated with each unit of the network, the nature of which may be unit-specific, encapsulating he relationship between input and output in the following way:

$$o_{i,l} = f_{i,l}(a_{i,l}). \tag{3.2}$$

An assumption is made that the activation function associated with the i-th unit in layer $\mathcal{L}_l$ can be either in the form

$$f_{i,l}(a_{i,l}) = \lambda_{i,l} \tilde{f}_{i,l}(a_{i,l}) + \sigma_{i,l} \tag{3.3}$$

which could be the case of sigmoids or Gaussians with learnable amplitude $\lambda_{i,l}$ and an offset (shift) $\sigma_{i,l}$ along the ordinate axis ($\sigma_{i,l}$ in turn can be a constant or, more generally, a function of $\lambda_{i,l}$); or in the form

$$f_{i,l}(a_{i,l}) = \tilde{f}_{i,l}(a_{i,l}) \tag{3.4}$$

that could be the case of linear activation, for instance. Anyway, in the following, the symbol $\tilde{f}_{i,l}(a_{i,l})$ will refer to the analytical expression of a function of $a_{i,l}$ which does not explicitly depend on $\lambda_{i,l}$.

Although the notation introduced herein may appear a bit cumbersome, it is useful in order to state the inductive hypothesis required to formulate general expressions for the learning rules that are introduced in the next Sections. Different learning rules are obtained for each one of three major Cases: (*i*) a unique, adaptive λ is shared among all the units for which a trainable amplitude is required; (*ii*) a different value λ_l of the amplitude is defined for each layer $\mathcal{L}_l$, and possibly shared among the units belonging to the same layer; (*iii*) different $\lambda_{i,l}$ are defined for each unit of the network.

3.2.1 Case 1: single λ over the whole network

As a first instance, we investigate the case where $\lambda_{i,l} = \lambda$ and $\sigma_{i,l} = \sigma$ for all the units for which equation (3.3) holds. Let us consider the training set $\mathcal{T} = (\mathbf{x}, \mathbf{y})$, which specifies the relationship between training input vectors and the corresponding desired (target) output vectors. An on-line learning algorithm is developed in order to minimize the squared error between target output and actual network output, on a sample by sample basis. The criterion function to be minimized at any time a new pattern from the training set is fed into the network is thus in the form:

$$C = \frac{1}{2} \sum_{i \in \mathcal{L}_L} (y_i - o_{i,L})^2 \tag{3.5}$$

and the stochastic gradient descent (Bengio, 1996) prescribes to update the current value of λ, obtaining a new value λ' for the amplitude, according to

$$\lambda' = \lambda + \Delta\lambda \tag{3.6}$$

where $\Delta\lambda$ is the amount of change and is given by:

$$\Delta\lambda = -\eta \frac{\partial C}{\partial \lambda}. \tag{3.7}$$

In equation (3.7) the constant η is the *learning rate*. It is not necessarily the same used to learn the connection weights, if a gradient-descent technique as the BP algorithm is used (in our experience, a much greater value for η is usually more effective). In order to compute equation (3.7) we write:

$$\begin{aligned} \frac{\partial C}{\partial \lambda} &= \sum_{i \in \mathcal{L}_L} \frac{\partial C}{\partial o_{i,L}} \frac{\partial o_{i,L}}{\partial \lambda} \qquad (3.8) \\ &= -\sum_{i \in \mathcal{L}_L} (y_i - o_{i,L}) \frac{\partial o_{i,L}}{\partial \lambda} \end{aligned}$$

and equation (3.7) can be rewritten as

$$\Delta\lambda = \eta \sum_{i \in \mathcal{L}_L} (y_i - o_{i,L}) \frac{\partial o_{i,L}}{\partial \lambda}. \tag{3.9}$$

From now on, efforts are concentrated on the calculation of a formulation for $\frac{\partial o_{i,L}}{\partial \lambda}$ which is required to be compact, computationally feasible and, especially, not depending on the specific architecture of the network at hand, e.g. it must hold true for an arbitrary number of hidden layers. To start with, note that two distinct cases may occur:

Case 1.1: equation (3.3) holds for unit i of layer $\mathcal{L}_L$, that is to say $f_{i,L}(a_{i,L}) = \lambda \tilde{f}_{i,L}(a_{i,L}) + \sigma$. Calculating the derivative of $o_{i,L}$ with respect to λ yields:

$$\frac{\partial o_{i,L}}{\partial \lambda} = \tilde{f}_{i,L}(a_{i,L}) + \lambda \frac{\partial \tilde{f}_{i,L}(a_{i,L})}{\partial \lambda} + \frac{\partial \sigma}{\partial \lambda} \tag{3.10}$$

that can be expanded by writing

$$\begin{aligned} \frac{\partial \tilde{f}_{i,L}(a_{i,L})}{\partial \lambda} &= \frac{\partial \tilde{f}_{i,L}(a_{i,L})}{\partial a_{i,L}} \frac{\partial a_{i,L}}{\partial \lambda} \\ &= \tilde{f}'_{i,L}(a_{i,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}}{\partial \lambda} \end{aligned} \tag{3.11}$$

and substituting equation (3.11) into equation (3.10), obtaining:

$$\frac{\partial o_{i,L}}{\partial \lambda} = \tilde{f}_{i,L}(a_{i,L}) + \lambda \tilde{f}'_{i,L}(a_{i,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}}{\partial \lambda} + \frac{\partial \sigma}{\partial \lambda}. \tag{3.12}$$

Case 1.2: equation (3.4) holds true, i.e. $f_{i,L}(a_{i,L}) = \tilde{f}_{i,L}(a_{i,L})$ (activation function does not depend on λ). In this case we can write:

$$\begin{aligned} \frac{\partial o_{i,L}}{\partial \lambda} &= \frac{\partial \tilde{f}_{i,L}(a_{i,L})}{\partial a_{i,L}} \frac{\partial a_{i,L}}{\partial \lambda} \\ &= \tilde{f}'_{i,L}(a_{i,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}}{\partial \lambda}. \end{aligned} \tag{3.13}$$

Although the expressions obtained for $\frac{\partial o_{i,L}}{\partial \lambda}$ in Case 1.1 and Case 1.2 are different, a common formulation is advisable. In passing, note that in both Cases a kind of recursive scheme is emerging, where derivatives of the outputs in a given layer are computed in terms of derivatives of outputs in the previous layer. Let us define the *auxiliary functions*

$$g_{k,l}(a_{k,l}) = \begin{cases} \tilde{f}_{k,l}(a_{k,l}) + \frac{\partial \sigma}{\partial \lambda} & \text{if } f_{k,l}(a_{k,l}) = \lambda \tilde{f}_{k,l}(a_{k,l}) + \sigma \\ 0 & \text{if } f_{k,l}(a_{k,l}) = \tilde{f}_{k,l}(a_{k,l}) \end{cases} \tag{3.14}$$

and

$$h_{k,l}(a_{k,l}) = \begin{cases} \lambda \tilde{f}'_{k,l}(a_{k,l}) & \text{if } f_{k,l}(a_{k,l}) = \lambda \tilde{f}_{k,l}(a_{k,l}) + \sigma \\ \tilde{f}'_{k,l}(a_{k,l}) & \text{if } f_{k,l}(a_{k,l}) = \tilde{f}_{k,l}(a_{k,l}). \end{cases} \tag{3.15}$$

Using the auxiliary functions it is possible to rewrite equations (3.12) and (3.13) in the following, common form:

$$\frac{\partial o_{i,L}}{\partial \lambda} = g_{i,L}(a_{i,L}) + h_{i,L}(a_{i,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}}{\partial \lambda}. \tag{3.16}$$

We finally need to introduce the notion of *expansion* function of unit k of layer $\mathcal{L}_l$, for $l = 1, \ldots, L$, which is defined as:

$$x_{k,l}(a_{k,l}) = \begin{cases} g_{k,l}(a_{k,l}) & \text{if } l = 1 \\ g_{k,l}(a_{k,l}) + h_{k,l}(a_{k,l}) \sum_{n \in \mathcal{L}_{l-1}} w_{k,n,l} x_{n,l-1}(a_{n,l-1}) & \text{otherwise.} \end{cases} \tag{3.17}$$

The following theorem can now be easily proved, by induction, to hold true.

Theorem 3.1: $\frac{\partial o_{k,l}}{\partial \lambda} = x_{k,l}(a_{k,l})$ for each $l = 1, \ldots, L$ and for each $k \in \mathcal{L}_l$.

Proof: The proof is obtained by induction on the total number of layers L. If $L = 1$ it is sufficient to show that $\frac{\partial o_{k,1}}{\partial \lambda} = x_{k,1}(a_{k,1})$. Since in this case $o_{j,0} = x_j, \forall j \in \mathcal{L}_0$ (i.e the output from the units of the input layer do coincide with the components of the input vector $\mathbf{x}$), we have:

$$\frac{\partial o_{j,0}}{\partial \lambda} = 0 \tag{3.18}$$

and equation (3.16) reduces to:

$$\begin{aligned} \frac{\partial o_{k,1}}{\partial \lambda} &= g_{k,1}(a_{k,1}) \\ &= x_{k,1}(a_{k,1}) \end{aligned} \tag{3.19}$$

where the definition of the expansion function has been applied, thus completing the basis for the induction. The inductive step is accomplished by assuming that the theorem has been proven for up to $L-1$ layers, and showing that with this hypothesis it holds true also for networks with L layers. Indeed, if $\frac{\partial o_{j,L-1}}{\partial \lambda} = x_{j,L-1}(a_{j,L-1})$ equation (3.16) (for unit k in layer $\mathcal{L}_L$) can be written as:

$$\begin{aligned} \frac{\partial o_{k,L}}{\partial \lambda} &= g_{k,L}(a_{k,L}) + h_{k,L}(a_{k,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{k,j,L} x_{j,L-1}(a_{j,L-1}) \\ &= x_{k,L}(a_{k,L}) \end{aligned} \tag{3.20}$$

according to the definition of expansion function. This completes the proof. $\mathcal{Q}.\mathcal{E}.\mathcal{D}$.

As a corollary of Theorem 3.1, equation (3.9) can now be restated as:

$$\Delta\lambda = \eta \sum_{i \in \mathcal{L}_L} (y_i - o_{i,L}) x_{i,L}(a_{i,L}) \tag{3.21}$$

that is exactly the formulation we were looking for. Its implementation in the recursive form stated above is straightforward (a recursive function $x()$ can be readily implemented in any modern computer language), once the auxiliary functions $g()$ and $h()$ have been implemented in terms of the analytical form of the activations for the particular network at hand. Equation (3.21) is general and does not explicitly depend on the peculiar architecture of the network.

(Bishop, 1995), section 9.2.2, shows that the adoption of a regularizer such as simple *weight decay* (Plaut et al., 1986) that realizes a grouping of all the weights within a single class is not consistent with some properties of network mappings, namely with respect to linear transformation of input or output data. This may induce suspicion that also Case 1, which somehow groups all the connections with adaptive amplitude altogether, is subject to the same inconsistency. As a matter of fact, in general this is not true (see Section 3.4 for a detailed discussion of relationships between the present algorithm and weight grouping techniques). First of all, the present grouping concerns only the weights of connections starting from those units that actually have a trainable λ (e.g., the algorithm introduced above copes with situations where only a few units within a given layer may have adaptive amplitudes). For instance, the classic 2-layer architecture with sigmoidal hidden units and linear output (the latter having fixed amplitude) is grouped into two distinct classes: input-to-hidden weights, and hidden-to-output weights. Since biases are not affected by the algorithm and they may thus constitute a separate class (see Section 3.4), the same groups that turn out to be consistent according to the conclusions drawn in (Bishop, 1995) are possibly allowed also in this Case for that architecture.

Secondly, the conclusions outlined in (Bishop, 1995) are a consequence of the specific form of the implicit regularizer underlying simple weight decay, whereas no (explicit) regularizer is applied herein (see again Section 3.4).

Finally, for consistency with linear transformations over input or output spaces, (Bishop, 1995) underlines that the weights of the network trained on the transformed data should be equal to those of the corresponding net trained on the original (non-transformed) data, multiplied by a proper factor λ, defined as $\frac{1}{a}$ if a transformation of the kind $ax + b$ is applied to input data x; defined as c if a transformation of the form $cy + d$ is applied to each output y. The potential availability of such a λ factor within the present framework appears to be evident[1].

[1] Of course this is not a claim that running the algorithm on linearly transformed data *actually* converges to those specific values for λ, but simply that the present formalism enables taking into account the required multiplicative factor: proper values of λ allow for consistency with respect to linear transformations, differently from simple weight decay.

3.2.2 Case 2: different λ_m for each layer

Let us now consider the case where different values of the amplitude $\lambda_m, m = 1, \ldots, L$ (along with the corresponding σ_m) are allowed, i.e. λ_m is shared among the units of layer $\mathcal{L}_m$ for which equation (3.3) holds. As in the previous Case, different activation functions can be considered within a given layer. Given the cost function (3.5) over the training set $\mathcal{T} = (\mathbf{x}, \mathbf{y})$, the goal is developing an on-line rule to learn λ_m, for each m. Using gradient descent, an iterative scheme of the form

$$\lambda'_m = \lambda_m + \Delta\lambda_m \tag{3.22}$$

is pursued, where λ'_m is the new (learned) value, while $\Delta\lambda_m$ is the amount of change:

$$\Delta\lambda_m = -\eta \frac{\partial C}{\partial \lambda_m} \tag{3.23}$$

where η is the learning rate. Different learning rates are possibly applied, according to m (note that, from now on, we will not explicitly write that formulas hold for $m = 1, \ldots, L$). As in equation (3.8) we can write:

$$\frac{\partial C}{\partial \lambda_m} = -\sum_{i \in \mathcal{L}_L} (y_i - o_{i,L}) \frac{\partial o_{i,L}}{\partial \lambda_m} \tag{3.24}$$

that allows us to rewrite equation (3.23) in the following form:

$$\Delta\lambda_m = \eta \sum_{i \in \mathcal{L}_L} (y_i - o_{i,L}) \frac{\partial o_{i,L}}{\partial \lambda_m}. \tag{3.25}$$

Exactly as in Section 3.2.1, out goal is now to introduce a general, compact form for $\frac{\partial o_{i,L}}{\partial \lambda_m}$, not depending upon the architecture of the network or the kind of activation functions. An expansion function, defined in terms of auxiliary functions, is defined and used to reach such a goal. Its form is indeed very close to equation (3.17), except for the fact that it takes into account also the layer m at which the expansion has to be applied. Considering equation (3.25), two distinct cases can be distinguished:

Case 2.1: $f_{i,L}(a_{i,L}) = \lambda_L \tilde{f}_{i,L}(a_{i,L}) + \sigma_L$. If $m = L$ then the derivative of $o_{i,L}$ with respect to λ_m becomes:

$$\frac{\partial o_{i,L}}{\partial \lambda_m} = \tilde{f}_{i,L}(a_{i,L}) + \frac{\partial \sigma_L}{\partial \lambda_m} \tag{3.26}$$

whereas if $m \neq L$ the same derivative can be written as:

$$\frac{\partial o_{i,L}}{\partial \lambda_m} = \lambda_L \tilde{f}'_{i,L}(a_{i,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}}{\partial \lambda_m}. \tag{3.27}$$

Case 2.2: $f_{i,L}(a_{i,L}) = \tilde{f}_{i,L}(a_{i,L})$ (activation function does not depend on λ_m). Again, if $m = L$ we can write:

$$\frac{\partial o_{i,L}}{\partial \lambda_m} = \frac{\partial \tilde{f}_{i,L}(a_{i,L})}{\partial \lambda_m} = 0 \tag{3.28}$$

and if $m \neq L$ we have:

$$\begin{aligned} \frac{\partial o_{i,L}}{\partial \lambda_m} &= \frac{\partial \tilde{f}_{i,L}(a_{i,L})}{\partial a_{i,L}} \frac{\partial a_{i,L}}{\partial \lambda_m} \\ &= \tilde{f}'_{i,L}(a_{i,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}}{\partial \lambda_m}. \end{aligned} \tag{3.29}$$

The definition of the *auxiliary functions* follows:

$$g_{k,l,m}(a_{k,l}) = \begin{cases} \tilde{f}_{k,l}(a_{k,l}) + \frac{\partial \sigma_l}{\partial \lambda_m} & \text{if } f_{k,l}(a_{k,l}) = \lambda_l \tilde{f}_{k,l}(a_{k,l}) + \sigma_l \text{ and } m \geq l \\ 0 & \text{otherwise} \end{cases} \tag{3.30}$$

and

$$h_{k,l,m}(a_{k,l}) = \begin{cases} 0 & \text{if } l = m \\ \lambda_l \tilde{f}'_{k,l}(a_{k,l}) & \text{if } f_{k,l}(a_{k,l}) = \lambda_l \tilde{f}_{k,l}(a_{k,l}) + \sigma_l \text{ and } l \neq m \\ \tilde{f}'_{k,l}(a_{k,l}) & \text{if } f_{k,l}(a_{k,l}) = \tilde{f}_{k,l}(a_{k,l}) \text{ and } l \neq m. \end{cases} \tag{3.31}$$

It is now possible to rewrite equations (3.26), (3.27), (3.28), and (3.29) in the common form:

$$\frac{\partial o_{i,L}}{\partial \lambda_m} = g_{i,L,m}(a_{i,L}) + h_{i,L,m}(a_{i,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}}{\partial \lambda_m}. \tag{3.32}$$

The m-th *expansion* of unit k of layer $\mathcal{L}_l$, for $l = 1, \ldots, L$, is finally defined as:

$$x_{k,l,m}(a_{k,l}) = \begin{cases} g_{k,l,m}(a_{k,l}) & \text{if } l = 1 \\ g_{k,l,m}(a_{k,l}) + h_{k,l,m}(a_{k,l}) \sum_{n \in \mathcal{L}_{l-1}} w_{k,n,l} x_{n,l-1,m}(a_{n,l-1}) & \text{otherwise.} \end{cases} \tag{3.33}$$

Following the inductive scheme used in the proof of Theorem 3.1, the following theorem can be shown to hold true:.

Theorem 3.2: $\frac{\partial o_{k,l}}{\partial \lambda_m} = x_{k,l,m}(a_{k,l})$ for each $l = 1, \ldots, L$, for each $k \in \mathcal{L}_l$ and for each $m = 1, \ldots, L$.

Proof: See Appendix A.1. $\mathcal{Q.E.D}$.

An immediate, key corollary of Theorem 3.2 is that equation (3.25) can be reduced to:

$$\Delta \lambda_m = \eta \sum_{i \in \mathcal{L}_L} (y_i - o_{i,L}) x_{i,L,m}(a_{i,L}) \tag{3.34}$$

which, exactly as rule (3.21), can be readily implemented by taking advantage of the recursive form of the auxiliary and expansion functions.

3.2.3 Case 3: different $\lambda_{i,l}$ for each unit of the network

This is the most general Case, coinciding with the setup described at the beginning of Section 3.2, where each activation function for a generic unit i in any layer $\mathcal{L}_l$ may be either in form (3.3) (i.e. depending upon its own amplitude $\lambda_{i,l}$ and offset $\sigma_{i,l}$) or in the form (3.4). The aim is again the minimization of the cost function (3.5) by means of stochastic gradient descent, according to rules of the form

$$\Delta\lambda_{i,l} = -\eta \frac{\partial C}{\partial \lambda_{i,l}} \tag{3.35}$$

defined for each $l = 1, \ldots, L$ and for each $i \in \mathcal{L}_l$ for which equation (3.3) holds (a different instance of the rule, possibly with different learning rates η, is to be used for each unit). Of course, the calculations that follow do rely on the implicit assumption that the activation function is in the form $f_{i,l}(a_{i,l}) = \lambda_{i,l}\tilde{f}_{i,l}(a_{i,l})+\sigma_{i,l}$, since otherwise no amplitude would be defined and could thus be learned. The derivation of the present learning rule is much closer to the *delta rule* of the BP algorithm than in the previous Cases 1 and 2, actually relying on the backpropagation of errors from the output layer down to adjacent hidden layers.

Consider first the case where i is an output unit, that is to say $l = L$. We can write:

$$\begin{aligned} \frac{\partial C}{\partial \lambda_{i,L}} &= \frac{\partial C}{\partial o_{i,L}} \frac{\partial o_{i,L}}{\partial \lambda_{i,L}} \\ &= -(y_i - o_{i,L}) \frac{\partial o_{i,L}}{\partial \lambda_{i,L}} \end{aligned} \tag{3.36}$$

where

$$\begin{aligned} \frac{\partial o_{i,L}}{\partial \lambda_{i,L}} &= \tilde{f}_{i,L}(a_{i,L}) + \lambda_{i,L} \frac{\partial \tilde{f}'_{i,L}(a_{i,L})}{\partial \lambda_{i,L}} + \frac{\partial \sigma_{i,L}}{\partial \lambda_{i,L}} \\ &= \tilde{f}_{i,L}(a_{i,L}) + \frac{\partial \sigma_{i,L}}{\partial \lambda_{i,L}}. \end{aligned} \tag{3.37}$$

Substituting the result into equation (3.36) and the latter into equation (3.35), we obtain:

$$\Delta\lambda_{i,L} = \eta(y_i - o_{i,L})\{\tilde{f}_{i,L}(a_{i,L}) + \frac{\partial \sigma_{i,L}}{\partial \lambda_{i,L}}\} \tag{3.38}$$

which resembles the familiar delta rule for weight update in the BP algorithm, to some extent.

Let us now consider the case of units in the first hidden layer, i.e. $l = L - 1$. Taking partial derivative of the cost function with respect to $\lambda_{i,L-1}$ and repeatedly using the *chain rule* yield:

$$\begin{aligned} \frac{\partial C}{\partial \lambda_{i,L-1}} &= \frac{\partial C}{\partial o_{i,L-1}} \frac{\partial o_{i,L-1}}{\partial \lambda_{i,L-1}} \\ &= \frac{\partial C}{\partial o_{i,L-1}} \{\tilde{f}_{i,L-1}(a_{i,L-1}) + \frac{\partial \sigma_{i,L-1}}{\partial \lambda_{i,L-1}}\} \end{aligned} \tag{3.39}$$

where the same passages as those used in equation (3.37) have been applied to expand $\frac{\partial o_{i,L-1}}{\partial \lambda_{i,L-1}}$. The calculation of $\frac{\partial C}{\partial o_{i,L-1}}$ can be accomplished by writing:

$$\begin{aligned} \frac{\partial C}{\partial o_{i,L-1}} &= \sum_{j \in \mathcal{L}_L} \frac{\partial C}{\partial o_{j,L}} \frac{\partial o_{j,L}}{\partial o_{i,L-1}} \\ &= -\sum_{j \in \mathcal{L}_L} (y_j - o_{j,L}) \frac{\partial o_{j,L}}{\partial o_{i,L-1}} \\ &= -\sum_{j \in \mathcal{L}_L} (y_j - o_{j,L}) \frac{\partial o_{j,L}}{\partial a_{j,L}} \frac{\partial a_{j,L}}{\partial o_{i,L-1}} \\ &= -\sum_{j \in \mathcal{L}_L} (y_j - o_{j,L}) f'_{j,L}(a_{j,L}) w_{j,i,L}. \end{aligned} \tag{3.40}$$

Substituting equation (3.40) into equation (3.39), and the latter in turn into equation (3.35) we obtain:

$$\begin{aligned} \Delta \lambda_{i,L-1} &= \eta \{\sum_{j \in \mathcal{L}_L} w_{j,i,L}(y_j - o_{j,L}) f'_{j,L}(a_{j,L})\}\{\tilde{f}_{i,L-1}(a_{i,L-1}) + \frac{\partial \sigma_{i,L-1}}{\partial \lambda_{i,L-1}}\} \\ &= \eta \{\sum_{j \in \mathcal{L}_L} w_{j,i,L} \delta_{j,L}\}\{\tilde{f}_{i,L-1}(a_{i,L-1}) + \frac{\partial \sigma_{i,L-1}}{\partial \lambda_{i,L-1}}\} \end{aligned} \tag{3.41}$$

where we wrote $\delta_{j,L}$ to denote the quantity $(y_j - o_{j,L}) f'_{j,L}(a_{j,L})$.

Finally, for subsequent layers, i.e. $l < L - 1$ we can still write:

$$\frac{\partial C}{\partial \lambda_{i,l}} = \frac{\partial C}{\partial o_{i,l}} \{\tilde{f}_{i,l}(a_{i,l}) + \frac{\partial \sigma_{i,l}}{\partial \lambda_{i,l}}\} \tag{3.42}$$

as in equation (3.39). Let us now assume that, for $m = l+1, \ldots, L-1$ and $h \in \mathcal{L}_m$, the following equation holds:

$$\frac{\partial C}{\partial o_{h,m}} = -\sum_{k \in \mathcal{L}_{m+1}} w_{k,h,m+1} \delta_{k,m+1} \tag{3.43}$$

(equation (3.40) shows that this is true for $m = L - 1$, yielding the basis for the following induction), where $\delta_{i,l}$ is recursively defined as:

$$\delta_{i,l} = \begin{cases} (y_i - o_{i,L}) f'_{i,L}(a_{i,L}) & \text{if } l = L \\ \{\sum_{j \in \mathcal{L}_{l+1}} w_{j,i,l+1} \delta_{j,l+1}\} f'_{i,l}(a_{i,l}) & \text{otherwise.} \end{cases} \tag{3.44}$$

With this hypothesis it is possible to write:

$$\begin{aligned}\frac{\partial C}{\partial o_{i,l}} &= \sum_{j \in \mathcal{L}_{l+1}} \frac{\partial C}{\partial o_{j,l+1}} \frac{\partial o_{j,l+1}}{\partial o_{i,l}} \\ &= -\sum_{j \in \mathcal{L}_{l+1}} \{ \sum_{k \in \mathcal{L}_{l+2}} w_{k,j,l+2} \delta_{k,l+2} \} \frac{\partial o_{j,l+1}}{\partial o_{i,l}} \\ &= -\sum_{j \in \mathcal{L}_{l+1}} \{ \sum_{k \in \mathcal{L}_{l+2}} w_{k,j,l+2} \delta_{k,l+2} \} \frac{\partial o_{j,l+1}}{\partial a_{j,l+1}} \frac{\partial a_{j,l+1}}{\partial o_{i,l}} \\ &= -\sum_{j \in \mathcal{L}_{l+1}} \{ \sum_{k \in \mathcal{L}_{l+2}} w_{k,j,l+2} \delta_{k,l+2} \} f'_{j,l+1}(a_{j,l+1}) w_{j,i,l+1}. \end{aligned} \tag{3.45}$$

Substituting equation (3.45) into equation (3.42), and the latter into equation (3.35) we obtain:

$$\begin{aligned}\Delta\lambda_{i,l} &= \eta\{ \sum_{j \in \mathcal{L}_{l+1}} w_{j,i,l+1} [\sum_{k \in \mathcal{L}_{l+2}} w_{k,j,l+2} \delta_{k,l+2}] f'_{j,l+1}(a_{j,l+1}) \} \{ \tilde{f}_{i,l}(a_{i,l}) + \frac{\partial \sigma_{i,l}}{\partial \lambda_{i,l}} \} \\ &= \eta\{ \sum_{j \in \mathcal{L}_{l+1}} w_{j,i,l+1} \delta_{j,l+1} \} \{ \tilde{f}_{i,l}(a_{i,l}) + \frac{\partial \sigma_{i,l}}{\partial \lambda_{i,l}} \} \end{aligned} \tag{3.46}$$

where we used definition (3.44) for $\delta_{j,l+1}$. Given this definition and considering equation (3.46) it immediately follows (by induction) that since assumption (3.43) holds for $m = L-1$, it also holds for all subsequent layers $L-2, \ldots, 1$. This reasoning encapsulates the familiar idea of backpropagating deltas from the topmost to lower layers, one at a time, using the above equations. More precisely, all of the above discussion can be summarized by rewriting equations (3.38), (3.41) and (3.46) in the common form:

$$\Delta\lambda_{i,l} = \begin{cases} \eta(y_i - o_{i,L})\{\tilde{f}_{i,L}(a_{i,L}) + \frac{\partial \sigma_{i,L}}{\partial \lambda_{i,L}}\} & \text{if } l = L \\ \eta\{\sum_{j \in \mathcal{L}_{l+1}} w_{j,i,l+1} \delta_{j,l+1}\}\{\tilde{f}_{i,l}(a_{i,l}) + \frac{\partial \sigma_{i,l}}{\partial \lambda_{i,l}}\} & \text{otherwise.} \end{cases} \tag{3.47}$$

This mechanism allows for the computation of $\Delta\lambda_{i,l}$ for each unit at any given layer. It should be noticed that the *deltas* are exactly those computed in the BP algorithm for weight update. This means that the present algorithm can be easily implemented by using the same quantities already available within the learning procedure, at each step, if BP is used. As in BP, the process can be described as backpropagating deltas along the connections, by multiplying by the corresponding connection weights, until the desired unit is reached.

Although Case 3 may seem far more powerful than previous Cases 1 and 2, allowing for training of individual amplitudes on a unit-specific basis (that is exactly what is required in certain circumstances), the resulting algorithm has more parameters to be determined by the learning process, so that a gain in performance with respect to the other Cases (or even to the bare network, without adaptive amplitudes) is not guaranteed.

Section 3.4 will show that the behavior of the present algorithms can be interpreted as a particular weight grouping technique, and that the grouping perspective allows for a better understanding of the rationale behind the improvements that are gained over standard BP with fixed amplitudes. From this point of view, it is clear that Case 3 appears to have particular relevance whenever multiple connections start from the neuron under consideration, i.e. more than one unit is present in the next layer of the network, since otherwise a single-weight group, which is no longer a group, is obtained. Indeed, in the latter case the equivalence between the unit with trainable amplitude λ and single weight w and the corresponding unit with fixed amplitude and weight $\tilde{w} = \lambda w$ is strong (unless an *output* unit is considered), and the effect of the present algorithm is assimilable to a double-step gradient descent over the weight $\tilde{w}$: these topics are thoroughly developed in the next Sections.

3.3 Experimental Demonstrations

3.3.1 Synthetic problem: teacher and student

A simple synthetic problem is introduced first, to illustrate the behavior of the algorithms and to compare their performance with standard training techniques. The discussion provided in Section 3.4 investigates the behavior of the algorithms in deeper detail and from different perspectives, forming the basis for a better understanding of the results presented in these Sections.

The problem is a *teacher-student* task, i.e. the parameters of a teacher network with a given architecture are randomly initialized and the net is used to generate target vectors to be used for the training of a student network. A three-layer architecture was chosen for the teacher net, with 2 input units, 2 linear output units, and 5 standard sigmoids in both the first and second hidden layers. More precisely, sigmoidal activations are in the following form:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.48}$$

which has fixed amplitude, equal to 1.0, and no bias or variable smoothness.

The architecture of the student net is similar to that of the teacher, with 2 inputs, 2 linear outputs, and 2 hidden layers with sigmoidal nonlinearities as defined by (3.48), but the number of units is increased, namely 20 instead of 5 per each hidden layer. The exclusion of trainable bias from (3.48) allowed for a simpler and clearer overall training process, focusing attention only on amplitudes and connection weights.

In order to evaluate statistical significance of the results, 5 instances of three datasets (training, validation and test set) were generated. The input pairs were randomly drawn from a uniform distribution in R^2, while the corresponding target values were obtained as the outputs yielded by the teacher net over the inputs. One of the instances is shown in Figure 3.2. Experiments with the student net were then repeated starting from 5 different random initializations of the weights (uniformly distributed over the interval $(-0.05, 0.05)$), for a total of 25 experiments for each of the training techniques[2].

Since the student network has many more hidden units than the teacher net has, the former may perfectly model the mapping for an appropriate choice of the parameters (yielding zero error on training and test sets), but it is also prone to fall into several wrong local minima. In particular, a subset of the hidden units should turn out to be redundant. It will be shown in the following that an automatic topology-pruning technique may be achieved as a side-effect of Case 3, which allows for the deletion of redundant neurons without worsening the performance of the resulting network.

Evaluation of performance is expressed in terms of average Mean Squared Error (MSE). The results are summarized in Table 3.1, along with the corresponding standard deviations.

	Training Set: MSE		*Test Set: MSE*	
Student model/Learning algorithm	*Average*	*Std. dev.*	*Average*	*Std. dev.*
Linear transformation with SVD	5.641	2.550	6.509	2.908
MLP with fixed amplitude, 50 epochs of BP	0.190	0.085	0.231	0.104
MLP with fixed amplitude, 100 epochs of BP	0.134	0.062	0.241	0.111
MLP with fixed amplitude, adaptive learning rate	0.148	0.070	0.207	0.116
MLP with fixed amplitude, Conjugate Gradient	0.170	0.076	0.206	0.092
MLP with single trainable amplitude	0.074	0.042	0.121	0.055
MLP with layer-by-layer trainable amplitude	0.068	0.036	0.107	0.049
MLP with neuron-by-neuron trainable amplitude	0.061	0.033	0.088	0.042

Table 3.1: *Mean Squared Error (MSE) obtained over the training and the test set in the teacher-student problem, for each one of the student models and training algorithms. Average and unbiased standard deviation over a series of 25 randomly-initialized experiments are reported.*

A simple linear regression model computed via *Singular Value Decomposition* (SVD) (Deprettere, 1988) was first attempted, in order to have a comparison

[2] Since the nature of the activation functions and the layered architectures were common between the student and the teacher net, no data normalization was assumed to be necessary in order for the student to learn the mapping defined by the teacher. All results presented in this Section are thus obtained using non-normalized inputs and outputs. For the sake of completeness, several experiments discussed in the following were also repeated applying zero-mean, unit-variance data normalization, without affecting to any significant extent the analysis and the conclusions drawn throughout the Section.

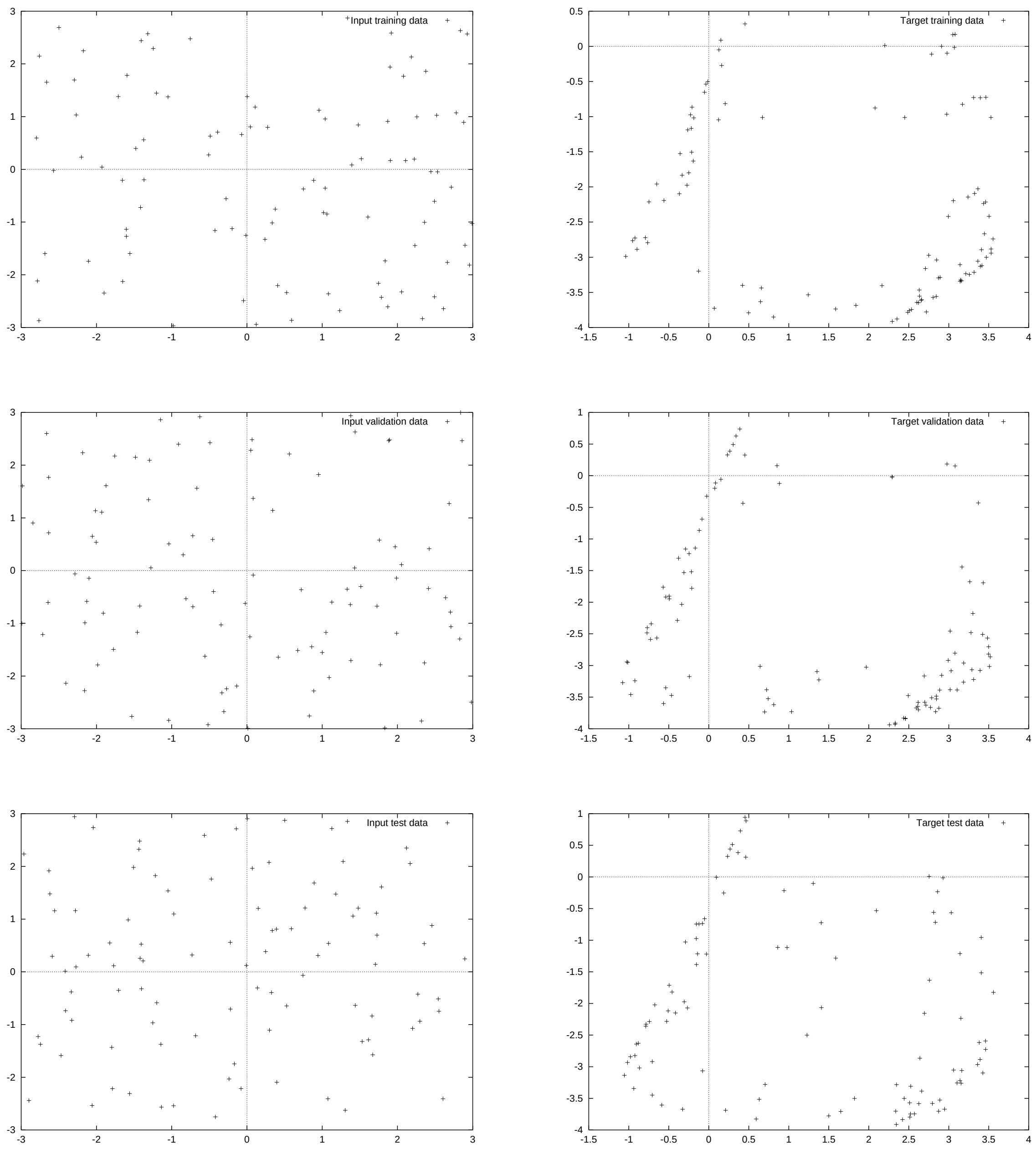

Figure 3.2: *One of the 5 datasets used in the teacher-student problem. The top-most scatterplots show the 2-dimensional input patterns (on the left) drawn from a uniform distribution, and the corresponding teacher-generated target outputs (on the right). Similarly, the scatterplots in the middle row show the validation set, and the bottom-most plots show the test set.*

baseline[3]. As shown below, all the following connectionist approaches improved over this linear model, underlining the heavily non-linear nature of the problem.

The student MLP was then initialized with random weights, and trained with BP. 50 training epochs were used, with a learning rate equal to 0.1. Here and in what follows, proper choices for the learning rate, the number of training epochs, and other algorithm-specific parameters (e.g. the adapting factor when an adaptive learning rate was used) were made by evaluating the behavior of the trained network over the validation sets.

Before repeating the simulations with other algorithms, let us consider a given weight w in the network, a given amplitude λ, and let us consider the quantity $\tilde{w} = \lambda w$. The latter is a weight in the corresponding net with fixed amplitudes that incorporates the λ's directly into the connection weights (the correspondence between the two nets is further analyzed in Section 3.4). Given this definition we can write:

$$\frac{\partial C}{\partial w} = \frac{\partial \tilde{w}}{\partial w}\frac{\partial C}{\partial \tilde{w}} = \lambda \frac{\partial C}{\partial \tilde{w}} \tag{3.49}$$

and, defining $W(\lambda)$ to be the set of all weights in the network that are subject to a given λ (i.e., all the weights exiting from units with amplitude λ):

$$\frac{\partial C}{\partial \lambda} = \sum_{w \in W(\lambda)} \frac{\partial \tilde{w}}{\partial \lambda}\frac{\partial C}{\partial \tilde{w}} = \sum_{w \in W(\lambda)} w \frac{\partial C}{\partial \tilde{w}} \tag{3.50}$$

that shows that applying one of the proposed schemes to train the amplitude(s) along with standard BP for weight updating implies two gradient descent steps (with respect to each $\tilde{w}$) at each iteration[4], and the proposed algorithms may head toward a minimum of the criterion function at double speed. This fact may be suspected of being the only reason why possible improvements in network performance are gained by the scheme described in this paper. A direct comparison with standard BP applied for a doubled number of training epochs is thus required. To achieve this, the student MLP was then trained by 100 epochs of BP. In so doing overfitting occurred and performance on test set was worsened on average. This fact underlines that the improvement obtained using trainable amplitudes is not, in general, a simple consequence of falling down into the same local minimum at double speed. In all the following experiments with gradient-descent training algorithms, 50 training epochs were adopted.

The algorithms proposed herein can also be seen as a particular scheme of BP with *adaptive learning rate* (ALR) (Jacobs, 1988; Hertz et al., 1991), where ALR updating is "modulated" by λ. In fact, the updating rule for weight w can be written as follows:

[3]Since SVD does not rely on any parameter initialization, the corresponding experimental results are averaged over 5 experiments only (one for each dataset), instead of 25.

[4]This is particularly evident if Case 3 is applied and a single connection is assumed to exit from a given unit, since equation (3.50) reduces to $\frac{\partial C}{\partial \lambda} = w\frac{\partial C}{\partial \tilde{w}}$.

$$\begin{aligned}\Delta w &= -\eta\frac{\partial C}{\partial w} \\ &= -\eta\frac{\partial\tilde{w}}{\partial w}\frac{\partial C}{\partial\tilde{w}} \\ &= -\eta\lambda\frac{\partial C}{\partial\tilde{w}}\end{aligned} \tag{3.51}$$

which can be thought of as an instance of standard BP (over $\tilde{w}$) with ALR $\eta\lambda$. If a single λ is considered over the whole net (Case 1 of Section 3.2), this corresponds to having an individual trainable learning rate. In Case 2 an ALR per layer is used. Finally, Case 3 corresponds to the adoption of neuron-specific trainable learning rates. This perspective, as further investigated in Section 3.4, differs from previous ALR schemes (see for instance (Jacobs, 1988; Hertz et al., 1991)), since it realizes a learning rate-updating process that is modulated by a gradient-derived factor, namely λ.

The algorithms presented do not realize *just* a standard, sped-up version of BP with self-correction of "wrong" initial choices for the learning rate. Although preliminary experiments that were accomplished and evaluated on the validation set led to what appeared to be a "right" value for the learning rate, and the dramatic reduction of MSE on test set gained by the standard MLP over the linear model seems to confirm the choice was indeed good, an ALR analogous to that described in (Hertz et al., 1991), section 6.2, was then applied along with standard BP for comparison purposes. Adaptation of the learning rate η was carried out by adding a quantity $\Delta\eta$ to the current value of η at the end of each training epoch, according to the following scheme:

$$\Delta\eta = \begin{cases} +b\eta & \text{if } \Delta C < 0 \\ -b\eta & \text{if } \Delta C > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.52}$$

where C is the value of the cost function averaged over all the patterns of the training set, and b is an *adapting factor* which controls the amount of change, determined by cross-validation. In the present setup, a value of 0.2 for b was chosen. As expected, the introduction of such an ALR allowed for a slight improvement, the latter being significant only to a limited extent.

A comparison with a training scheme that does not rely on any learning rate at all, namely the *conjugate gradient* algorithm (Bishop, 1995), was then accomplished to further ensure independence from the specific choice of the learning rate. In addition, conjugate gradient allows for an additional, important benchmark for the proposed techniques. The computation of the search direction in the weight space was accomplished using the *Polak-Ribiere* formula (Polak and Ribiere, 1969), which is generally found to perform better than other formulas (Bishop, 1995). The average MSE that was obtained on test set slightly improves the result scored by gradient-descent techniques described so far. This gain is likely to be due to the fact that, in general, conjugate-gradient and standard BP do not necessarily converge toward the same local minima of the criterion function. Before taking into consideration adaptive amplitudes, note that all the training techniques applied so far in the fixed-amplitude scenario yield test results that do not significantly differ each other.

Let us now evaluate the behavior of the amplitude-learning algorithms. Using 50 epochs of BP along with a single trainable amplitude outperformed the standard training techniques. Using the experimental results obtained with the standard MLP with fixed amplitude and those obtained with Case 1 of the algorithm to compute sample averages, deviations, and estimate pooled variance, a Student's t test (Box et al., 1978) enables us to discard the null hypothesis ("the proposed algorithm does not improve over BP in the present setup") at a level of significance lower than 0.005 %. Figure 3.3 shows the evolution of the amplitude, during one of the training sessions, as a function of the number of epochs.

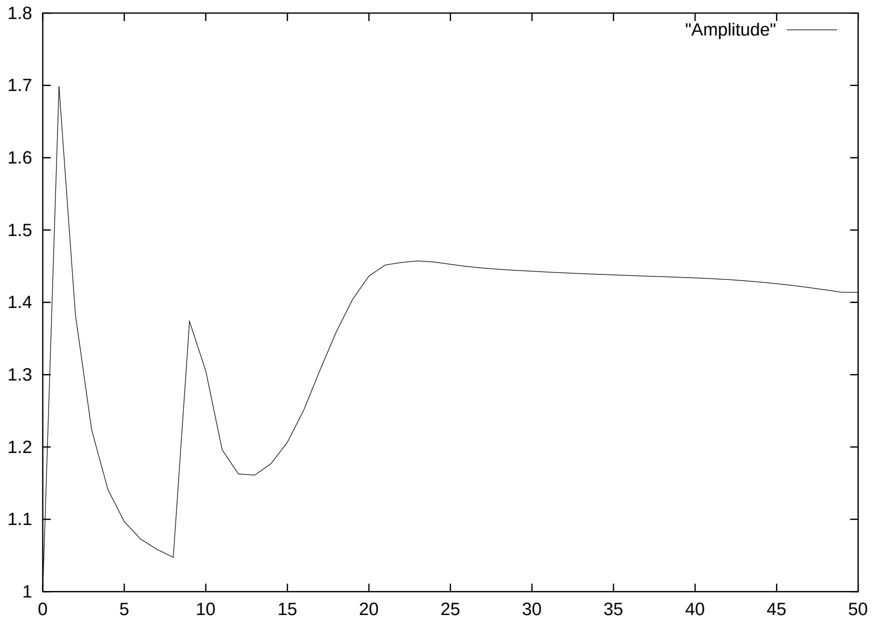

Figure 3.3: *Evolution of the single amplitude for the student network during one of the training sessions (the abscissa represents the number of training epochs).*

Considering then the layer-by-layer version (Case 2) of the algorithm, a further improvement in learning and generalization abilities can be observed. At the end of the 25 training sessions, the absolute values of the amplitudes in the two hidden layers ranged from -1.310 to 1.773 (starting up with the usual value of 1.0), All amplitudes in the first hidden layer were positive. The sign of the amplitudes in the second hidden layer turned out to be negative in 10 runs out of 25, i.e. switching to inverted sigmoids such as that shown in Figure 3.1 occurred. It is likely that even a larger improvement over Case 1 can be gained whenever activation function of different nature are employed in different layers: see Section 3.3.2 for an example.

Finally, Case 3 (unit-by-unit amplitude) yielded the best scores on both training and test sets. To reach such a result, random initialization of individual amplitudes was necessary, as commonly done for the weights: indeed, experiments accomplished on the validation set showed that uniform initialization to 1.0 did not allow for any improvement with respect to previous Cases. Although this should not be intended as a strict rule, breaking ties by starting the algorithm with slight differences in initial values may improve the learning process, as in standard BP. The amplitudes that emerged from the training process ranged in the interval $(-10.292, 10.130)$, in a non-uniform manner, thus inducing also inverted sigmoids, like in Case 2. Comparing the performance of the three instances of the proposed algorithm (Case 1, 2, and 3) it may appear that they correspond to increasingly effective training schemes. This conclusion is not true, in general, since behavior of the algorithms strictly depends on the specific learning problem at hand. For instance, Section 3.3.2 shows an example in which Case 3 worsens the performance of the other Cases.

As given advance notice of, an interesting side-effect of Case 3 can be observed by taking into consideration the values of individual amplitudes at the end of training. In fact, a number of them reached a final value very close to zero, i.e. very small if compared with the average size of the other amplitudes. It turns out that the contribution from the corresponding units to the overall behavior of the trained network is limited. Such units emerge from training as "redundant" in order to fit the data, and the algorithm attempts to turn them off. In this experimental setup this phenomenon is meaningful, since the student networks were deliberately chosen with more hidden units than the teacher nets have. Neurons with an amplitude below a given threshold can thus be *pruned*, without significantly affecting the performance of the resulting, simplified net. The latter is expected to yield about the same MSE on the training set, and to generalize at least as well as the original architecture. In the present experiments, a threshold was quickly determined by cross-validation. At the end of each experiment an average of 11 hidden units out of 40 was pruned, without affecting performance of the networks: the same average MSE was scored, a result that points out that contribution from the pruned units was definitely not influential. The algorithm was not capable to prune all the redundant neurons originally forced into the architecture of the student net, but the observed behavior can be considered satisfactory. Although in this example none of the amplitudes was exactly zero at the end of training, the next Section reports a case in which some neurons are completely turned off. Self-pruning of the latter is immediate, not requiring the introduction of any threshold.

Figure 3.4 offers comparative plots of the learning curves (on training set) and generalization curves (on validation set), in terms of MSE, as a function of the number of training epochs in one of the 25 experimental runs. The curves for BP with fixed amplitude are shown in the top graphic, while Case 3 of trainable amplitudes is shown in the bottom graphic, respectively. Training of Case 3 was protracted to 100 epochs to provide full comparison. It is worth noting that in Case 3 both training and validation errors, definitely lower than in the BP case,

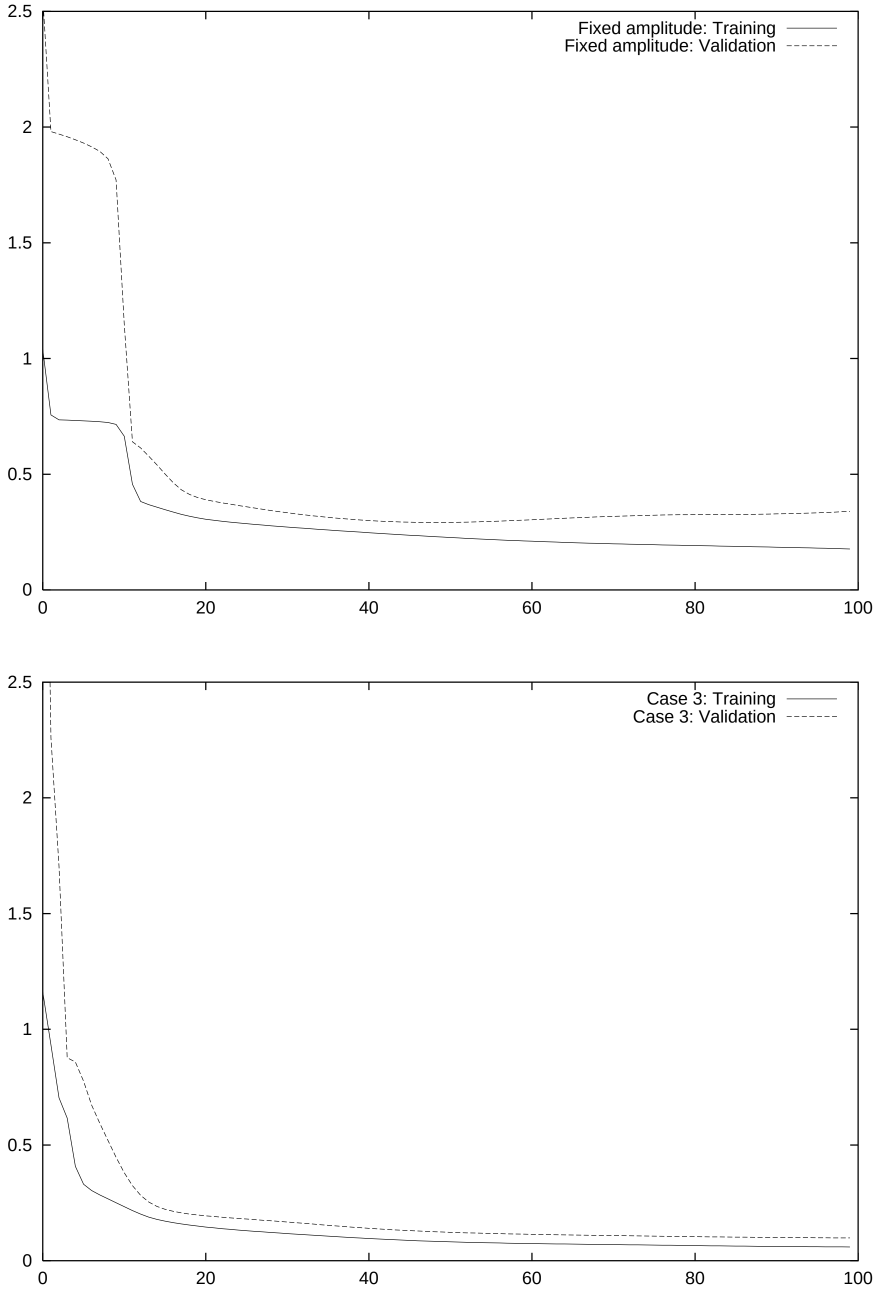

Figure 3.4: *Learning curve (MSE on the training set) and generalization curve (MSE on the validation set) for the teacher-student problem: standard BP with fixed amplitude (top) versus Case 3 of trainable amplitude (bottom). The abscissa represents the number of training epochs.*

continue to decrease up to the end of the training process, i.e. overfitting does not occur during the first 100 epochs. This implies that a much deeper minimum is approached, corresponding to a set of parameters for the student net that allows for a much closer approximation ("mimic") of the teacher net. The pruning effect plays an active role in this perspective, since the student architecture is simplified and more similar to the teacher.

The teacher-student problem considered so far involved nets with fixed architecture. Let us now turn our attention to the topic of architectural complexity, in particular in order to investigate the possible relationship between the behavior of the proposed algorithms and the network size, given a certain learning problem. This investigation may turn out to be useful in understanding *when* the different Cases of the algorithms are indeed mostly effective. Consider, for instance, Case 3 applied to a two-layer topology with a hidden layer of nonlinearities and a linear output layer. If a single output unit is present, all hidden-to-output weights are grouped into as many singletons as the number of connections (one for each nonlinearity in the hidden layer), each consisting of an individual weight. In practice there is no actual grouping, and the benefit taken from the algorithm with respect to BP may be expected to be marginal, being possibly limited to the double-step gradient descent and to the saturation-avoidance effects only. As the number of output units is increased, groups grow in cardinality and the algorithm appears more promising. On the contrary, Case 1 imposes the same λ over all the network units: a common but effective value for λ may turn out to be hardly learnable when the number of nonlinear units exceeds a certain threshold.

Additional experiments were thus performed, varying the complexity of both the teacher and the student nets. Two-layer architectures, with sigmoidal hidden units and linear outputs, were used. The other parameters (initial range of weights, learning rate, etc.) were inherited from the experiments discussed above. For the present topology, Cases 1 and 2 collapse into the same training scheme. Five teacher networks were created, having an increasing number of outputs, namely 1, 5, 10, 15 and 20. The number of hidden units was also increased accordingly, being set to 5, 10, 15, 20 and 25, respectively. Training, validation and test sets were randomly generated as above by application of the teacher nets to a uniform two-dimensional input distribution. For each of the given output dimensionalities, five student networks (with 10, 20, 30, 40 and 50 hidden units, respectively) were trained for 50 epochs and evaluated, for a total of 25 experiments.

The behavior of Cases 1/2, as well as of Case 3, with respect to bare BP was first evaluated as a function of the number of output units. Figure 3.5 shows the percentage of relative improvement over BP (i.e., the zero-level along the ordinate corresponds to BP), in terms of %MSE on the corresponding test sets (averaged over the five student nets with increasing number of hidden units). Cases 1/2 provide a relative improvement that shows limited fluctuations, and that is particularly relevant in the mid-sized output spaces (5 and 10 output units). Case 3, as expected from the above discussion, moderately worsens BP in the 1-output instance (no

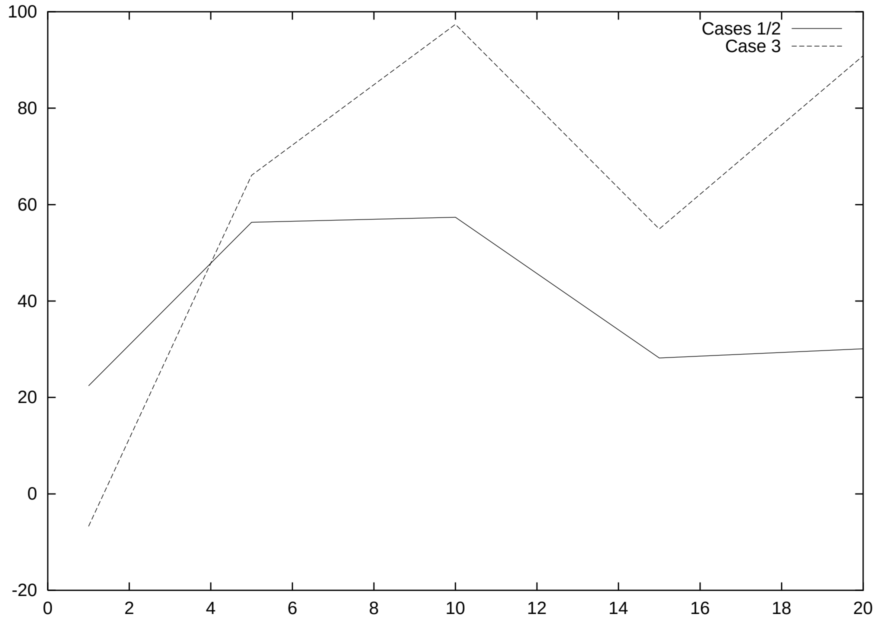

Figure 3.5: *Teacher-student problem: average relative variation (expressed in terms of %MSE, ordinate axis) provided by the proposed algorithms with respect to the bare BP, as a function of the number of output units (along the abscissa).*

actual grouping occurs), but dramatically improves as dimensionality grows (with a local exception for 15 outputs), outperforming Cases 1/2 when applied to the most complex architectures.

Complementary, Figure 3.6 shows the percentage of relative improvement over BP, in terms of % MSE on the corresponding test sets averaged over the five student nets with increasing number of outputs, as a function of the number of hidden units. When applied to the smallest architectures (10 hidden units) Cases 1/2 and Case 3 substantially yield comparable improvements. As complexity grows, Cases 1/2 slowly get less effective, while Case 3 quickly reaches a high *plateau*.

These experiments emphasize that, in the present scenario, Cases 1/2 turn out to be particularly suited for small and mid-sized architectures, due to the fact that grouping all weights in the net (Case 1) or all weights in a layer (Case 2) altogether in an effective manner becomes harder as the complexity of the model, i.e. the cardinality of each group, grows. On the contrary, splitting into unit-by-unit groups (Case 3) is mostly advantageous for complex models, but is uneffective or even worse than BP if a single output unit is introduced in a 1-hidden layer network. Next Section shows an example where Case 3 allows for good performance when applied to a 1-output net with 2 hidden layers of nonlinearities.

In summary, the following conclusions can be drawn from observation of the results reported so far:

1. in this teacher-student problem, trainable amplitudes allow for improvement in

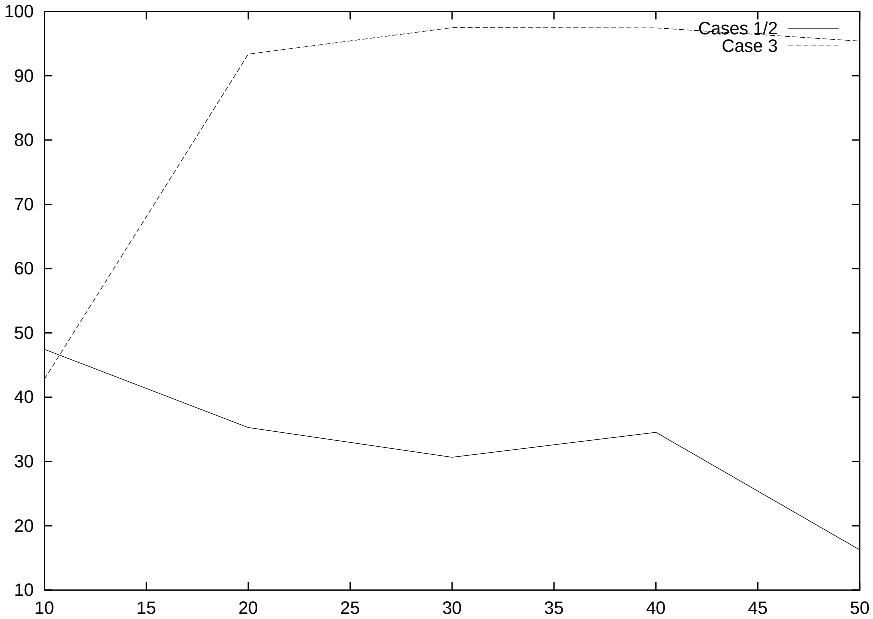

Figure 3.6: *Teacher-student problem: average relative variation (in terms of %MSE, ordinate axis) provided by the proposed algorithms with respect to the bare BP, as a function of the number of hidden units of the student network (along the abscissa).*

learning and generalization abilities, the latter being not only due to a double-speed BP effect or to a simple adaptive adjustment of the (maybe wrongly initialized) learning rate.

2. Introduction of trainable amplitudes affects the training process, so that *different trajectories* in the weights space are produced by the overall optimization scheme with respect to standard BP.

3. Whenever the same local minimum as in BP is reached, convergence is faster (double-step gradient descent). In other words, the present algorithms in conjunction with *early stopping* (Bishop, 1995) may turn out to be more efficient than standard BP. This aspect may be crucial whenever real-world huge datasets are used.

4. When, on the contrary, the same minimum as in BP is not reached, there emerges to be a statistically significant tendency to converge toward deeper minima without overfitting, if the same number of training epochs as in BP is applied (e.g. the "optimal" number of epochs for BP to obtain the best generalization on a validation set). In particular:

   (a) Since such minima allow for better generalization too, they are expected to coincide with better solutions, i.e. closer to the "right" one (the teacher net itself, in the present scenario). The self-pruning mechanism takes particular relevance in this respect.

(b) Since the phenomenon occurs also in Cases 1 and 2, explanation of this tendency may be searched for in the weight-grouping perspective. At each training iteration, BP provides for "local" (weight-by-weight) steps toward a local minimum, while the trainable amplitudes force jumps in the weight space that are globally relevant to a whole group (the whole net, in Case 1). After a few major jumps during the early training, the amplitudes tend to stabilize (see again Figure 3.3) and BP can eventually locally fine-drive individual weights to that minimum. A qualitative view of this process may be described as follows: since introduction of trainable λ's increases the dimensionality of the search space, the effect of the algorithms is to allow for instantaneous jumps into a hyper-(weight)space, from a local region of the original space, back to another, along paths over a modified error surface, while BP proceeds toward the nearby minimum into each one of these local regions.

(c) The weight-grouping and the pruning mechanisms underly the tendency to drive the process toward simpler, regularized solutions which are expected to provide for better generalization.

(d) Another reason why better minima are approached is due to saturation avoidance: a saturated sigmoid in BP actually reduces the dimensionality of the search space, i.e. a sub-space of the whole weight space can be explored to find a minimum. The saturation-avoidance effect prevents such phenomenon to occur. Sudden growths in amplitude as those reported in Figure 3.3 may turn out to be necessary during training in order to de-saturate certain hidden units.

Section 3.4 provides further discussion on these topics.

3.3.2 Three examples from speech processing tasks

Speaker identification

In *Speaker Identification* (Furui, 1981; Farrel et al., 1994) the aim is to identify a person, belonging to a given group (in the present case a closed-set identification task is faced, i.e. all the members of the group are known in advance), using her/his voice as a cue. To put this in the setup of statistical pattern recognition, each speaker can be seen as a class, with corresponding population constituted by the data set representing the acoustic features of the voice. An observation $\mathbf{x}$ is an acoustic vector obtained by means of short time spectral analysis of the speech signal. The first 8 *Mel Frequency Scaled Cepstral Coefficients* (MFSCCs) (Davis and Mermelstein, 1980) are used in the reported experiments.

The database, collected at ITC-irst, is composed of the Italian digits (from 0 to 9) uttered by 10 speakers. The spectral analysis was carried out using time windows of 30ms and with an analysis step of 10ms. For each speaker, 30 signals were collected and the data were divided into three equally sized sets, to be used for training, validation and test, respectively. Each of these 8-dimensional sets consisted of about 1,100 observations, on average, for each speaker.

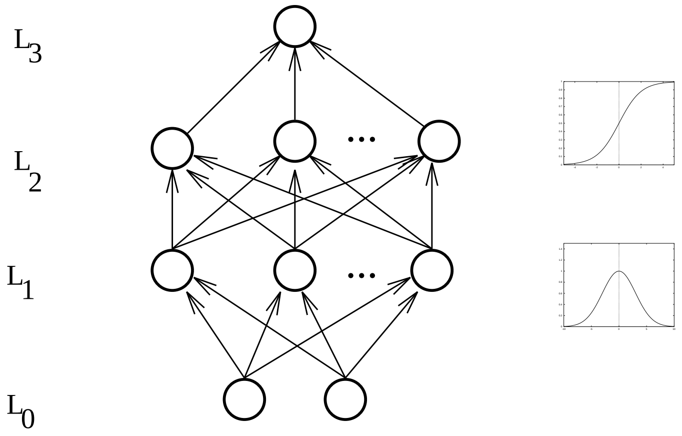

Figure 3.7: *Architecture of the feedforward network used as a posterior class probability estimator in the speaker identification task.*

Although a statistical technique is introduced below for comparison purposes, in this paper the goal is not to show that neural nets are optimal solutions to the problem, but to evaluate the behavior of the proposed training algorithms on a multilayer neural classifier with different analytical form for the activation functions.

The architecture (Figure 3.7) is a three-layer feedforward network. Activations in the first hidden layer ($\mathcal{L}_1$) are exponential in the form:

$$f_{i,1}(a_{i,1}) = \lambda_{i,1} \exp\{-\frac{1}{2}(\frac{a_{i,1} - \mu_{i,1}}{\theta})^2\} \tag{3.53}$$

where $\mu_{i,1}$ is a unit-specific trainable *center*, and θ is a smoothness which determines the variance of the exponential function. To use the terminology of generalized RBF (Poggio and Girosi, 1990; Girosi et al., 1995), equation (3.53) defines a Gaussian kernel $K_i(\mathbf{x}; \mu_i)$ over the transformed, variable metrics given by the weights $w_{i,j,1}$ of the first layer, the local basis function for unit i in $\mathcal{L}_1$ being $K_i(\mathbf{x}; \mu_i) = f_{i,1}(\sum_{j \in \mathcal{L}_0} w_{i,j,1} x_j)$, where $\mathbf{x} = (x_1, \ldots, x_8)$ is the input pattern.

Sigmoidal activation functions are used in the second hidden layer ($\mathcal{L}_2$):

$$f_{i,2}(a_{i,2}) = \frac{\lambda_{i,2}}{1 + e^{-\frac{a_{i,2} - \beta_{i,2}}{\theta}}} \tag{3.54}$$

where $\beta_{i,2}$ denotes a unit-specific adaptive bias and θ is the variable smoothness determining the shape (slope) of the sigmoid. Note that no offset $\sigma_{i,l}$ was used in

$\mathcal{L}_1$ or $\mathcal{L}_2$: this means that $\frac{\partial \sigma_{i,l}}{\partial \lambda_{i,l}} = 0$ and the corresponding term can be dropped from the equations derived in Section 3.2. The output layer ($\mathcal{L}_3$) is built of linear activations, namely:

$$f_{i,3}(a_{i,3}) = a_{i,3}. \tag{3.55}$$

The overall architecture can be thought of as a particular case of a nested RBF-MLP network. For each class (speaker) i involved in the task, an individual network was trained as an estimator of the posterior class probability given the acoustic observation. In practice, the i-th network was trained to yield an output equal to 1.0 when fed with the patterns sampled from i-th speaker, and 0.0 otherwise. All the acoustic observations of the training set were thus used as positive or negative examples for all the 10 networks, but the contribution of negative samples on the error function was multiplied by 0.1, in order to compensate for the unbalanced cardinality of the populations (of negative versus positive patterns). During the test, all the frames of each test signal were fed through all of the 10 networks, producing scores that were summed over the whole signal on a network by network basis, and the signal was assigned to the speaker with the best overall score (summation of scores over the sequence of acoustic observations is a common and effective practice in speaker identification (Falavigna, 1995)).

In order to establish a comparison baseline for the present classification task, relying on classical statistical pattern recognition techniques, a linear discriminant analysis (LDA) approach (Duda and Hart, 1973) was applied. A linear discriminant function of the form

$$g_i(\mathbf{x}) = \mathbf{w}_i'\mathbf{x} + b_i \tag{3.56}$$

was estimated using the *Widrow-Hoff* algorithm (Duda and Hart, 1973), for each class $i = 1. \ldots, 10$. In equation (3.56), $\mathbf{w}_i'$ denotes the transpose of the i-th column vector of the linear coefficients of the discriminant, and b_i is an offset term. The corresponding decision rule assigns observation $\mathbf{x}$ to class j if $g_j(\mathbf{x}) \geq g_i(\mathbf{x})$ for each i.

In speaker identification tasks, evaluation of the performance of the resulting classifier is accomplished in terms of error rate (which is an estimate of the Bayes' error (Duda and Hart, 1973)), i.e. the fraction of misclassifications over the whole test set, and of the Equal Error Rate (EER) (Li and Porter, 1988; Trentin, 1995; Trentin et al., 1995). The concept of EER can be summarized as follows: suppose that the *reject option* (Fukunaga, 1990) is used in a classification task, i.e. an observation is not assigned to any of the classes if none of the estimates of the posterior probability of the classes given the observation is over a given *rejection threshold* (the reject option is a requirement in most applicative scenarios). Once a rejection threshold has been fixed, a *false rejection* is said to occur whenever an observation is actually rejected but, at the same time, the classifier would have classified it correctly. It is possible to fix a threshold for which the probability of false rejection and the *Bayes' error* coincide, and equal a certain value ϵ. The latter is defined as the EER.

Table 3.2 shows the error rate and the EER obtained on the test set with the approaches introduced above. The results yielded by LDA are not surprising, due to the strong overlapping of classes in the acoustic feature space, as well as to the limited amount of training data used herein.

Classifier Type	*Error rate*	*EER*
Linear Discriminant	40%	18%
NN with fixed amplitude, 50 BP training epochs	8%	6%
NN with fixed amplitude, 100 BP training epochs	8%	6%
NN with fixed amplitude and adaptive learning rate	7%	5%
NN with fixed amplitude and conjugate gradient	9%	6%
NN with single trainable amplitude	4%	3%
NN with layer-by-layer trainable amplitude	2%	2%
NN with neuron-by-neuron trainable amplitude	1%	1%

Table 3.2: *Speaker identification problem. Error rate and Equal Error Rate (EER) obtained on the test set with the different proposed approaches (NN = Neural Network).*

The other rows of the Table report the results obtained with the neural networks. If not otherwise stated, 50 epochs of training were applied. The standard BP algorithm was used to learn the weights, the biases for equation (3.54) and the centers for equation (3.53). Learning was accomplished in an on-line manner (stochastic gradient descent), updating the values after presentation of each pattern of the training set. All the experiments started with an initial value of 1.0 for the amplitude $\lambda_{i,l}$ of the activations. As expected, all the connectionist classifiers outperformed the LDA approach.

The networks with fixed amplitudes were first evaluated. 100 training epochs were then applied, without affecting the performance on test set to any extent (a slight but definitely not significant decrease in terms of MSE on training set was observed). For the same comparative reasons already pointed out in Section 3.3.1, training of the model with an ALR (equation (3.51)), as well as with conjugate gradient and the Polak-Ribiere formula, were applied. As might be expected, adoption of the ALR allowed again a little improvement, while the conjugate gradient even worsened the results.

A single trainable λ was then introduced. All the other parameters (e.g. learning rate, smoothness, etc.) were left unchanged (this holds true for all the approaches that follow). In spite of the different nature of the nonlinearities in $\mathcal{L}_1$ and $\mathcal{L}_2$, the proposed algorithm (a direct application of rule (3.21)) allowed for a lowering of the error rate and of the EER. It is interesting to observe the evolution of λ during the training for each of the 10 class-dependent networks involved in the experiments. Figure 3.8 shows the corresponding curves. The final values of λ for the 10 nets ranged from 4.05 to 6.60.

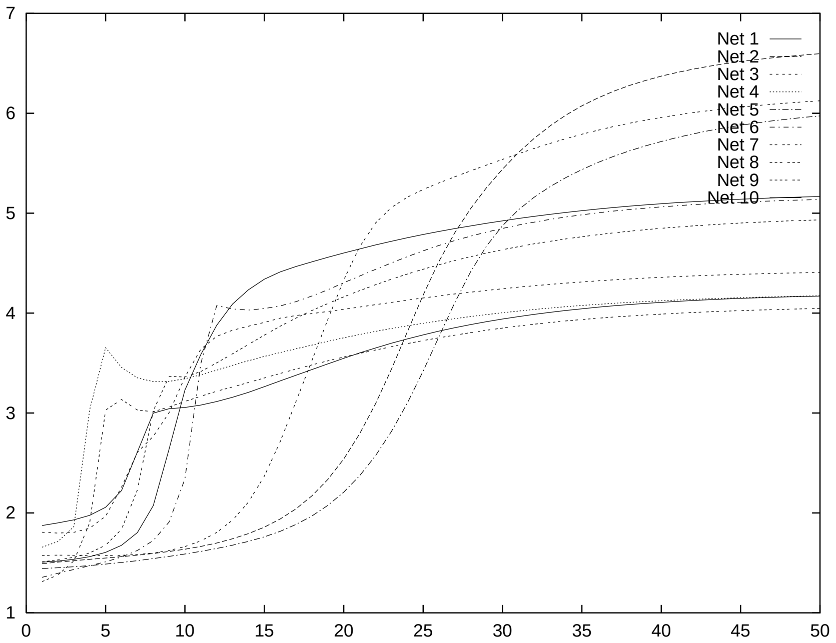

Figure 3.8: *Evolution of the widths for the 10 networks during the training (the abscissa represents the number of training epochs).*

The seventh row of Table 3.2 reports the results obtained with the network having different λ_l at each layer (Case 2, Section 3.2.2). This layered approach

Case 2 of the algorithm is well suited for the present architecture, allowing different amplitudes for Gaussian and sigmoidal activations. The experiments confirmed the expectations. The final values of λ_1 (Gaussian activations) for the 10 networks ranged from 6.17 to 9.57, while λ_2 (sigmoidal activations) ranged from 2.91 to 4.55.

Finally, individual amplitudes $\lambda_{i,l}$ for each neuron (Case 3, Section 3.2.3) were considered. The last row of Table 3.2 shows the performance of the resulting networks, i.e. an 87.5% relative error rate reduction with respect to the neural model with fixed amplitudes. It is worth reporting that, at the end of the training, some of the amplitudes associated with sigmoidal units resulted null, implying that some specific hidden units were redundant for some of the networks, allowing for their immediate pruning. Figure 3.9 shows graphs of the curves of error rate and false rejection rate, as functions of the rejection threshold, for the four major approaches (standard BP with fixed amplitude, single trainable amplitude, layer-by-layer trainable amplitudes, and neuron-by-neuron trainable amplitudes). The point in each graph where the curves intersect determines the EER (along the ordinate axis) and the corresponding rejection threshold (along the abscissa).

Figure 3.10 represents the learning curves in the same four cases, i.e. the average

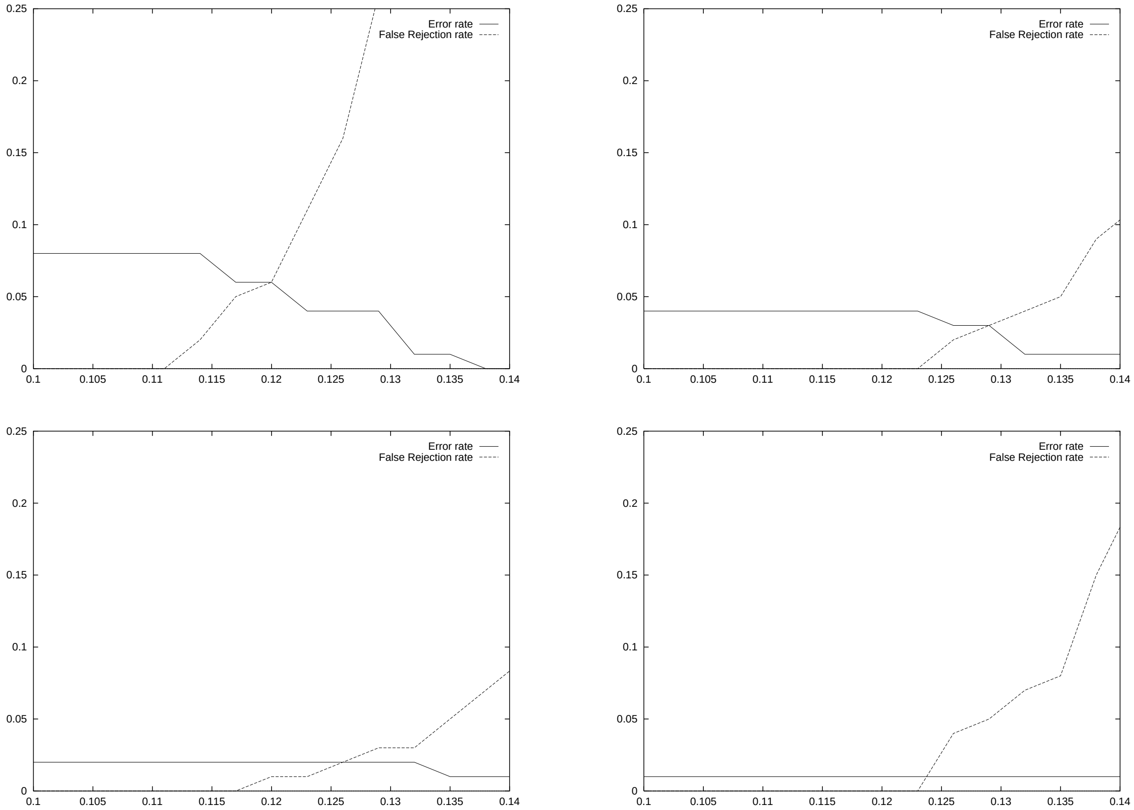

Figure 3.9: *Error rate and False Rejection rate as functions of the rejection threshold for the proposed connectionist approaches (from the left-top to the right-bottom: NN with fixed amplitude; single trainable amplitude; layer-by-layer trainable amplitudes; neuron-by-neuron trainable amplitudes). The point where the curves intersect determines the Equal Error Rate (along the ordinate axis) and the corresponding rejection threshold (along the abscissa).*

squared error on the training set as a function of the number of training epochs[5].

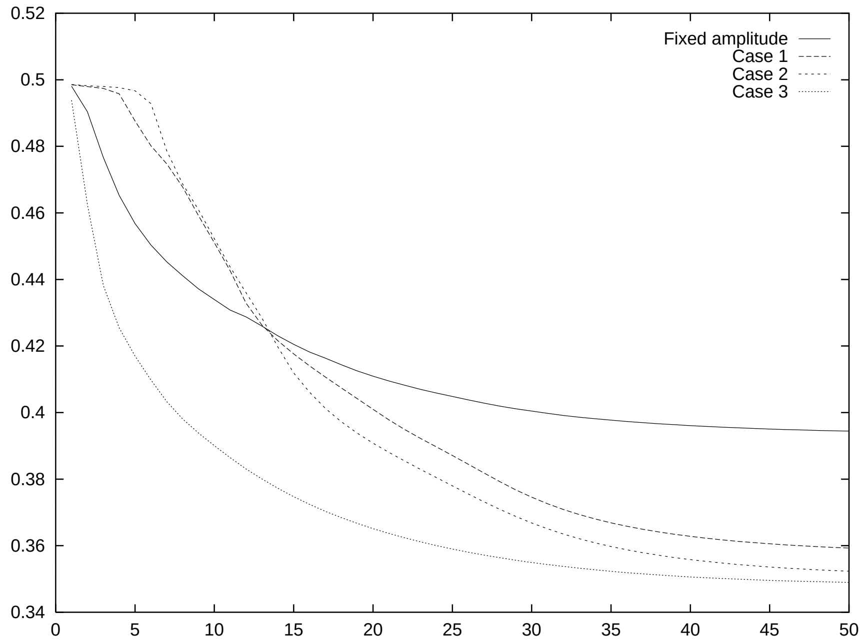

Figure 3.10: *Learning curves on the training set for the 4 connectionist approaches (the abscissa represents the number of training epochs).*

Speaker normalization

Speaker normalization (Huang, 1992; Trentin and Giuliani, 1997) can be formulated as a regression problem between the multidimensional acoustic spaces corresponding to different speakers. The acoustic space of a speaker is represented by means of a set of feature vectors, obtained by acoustic analysis of the speech signals acquired from the speaker herself. The aim is to estimate a vector function, namely *spectral mapping*, able to model the relationship between the acoustic patterns of a *new speaker* and the corresponding patterns of a *reference speaker*, whose speech data were used to train a speaker-dependent (SD) speech recognition system (Furui and Sondhi, 1991). Application of the spectral mapping on the acoustic data of the new speaker, before sending them to the SD recognizer, should allow the latter to perform acceptably, from an applicative point of view.

In the present case the transformation is realized with neural networks. The networks are trained on the time-aligned pairs of feature vectors (Furlanello et al., 1997; Trentin and Giuliani, 2001) of the speakers involved. The trained nets are

[5] As already pointed out, no relevant changes in MSE occurred when further 50 epochs of training were applied to the network with fixed amplitude, and the corresponding curve, which was substantially flat in the range between epochs 50 and 100, is consequently not shown in the graphics.

subsequently used, at the recognition step, as an acoustic front-end for the speech recognizer. The results are compared either with the baseline system performance, i.e., word error rate obtained with the speaker-dependent recognition system without normalization at all, or applying a linear regression model for normalization.

The feedforward model that was adopted is a 2-layer MLP with 48 hidden units. Training was accomplished via on-line BP, including a momentum term and an ALR, on a limited amount of acoustic material from each speaker (10 utterances). Sigmoidal activation functions were used in the hidden layer, with linear output units. More precisely, the chosen sigmoid for hidden unit i of layer l was in the form

$$f_{i,l}(a_{i,l}) = \frac{\lambda_{i,l}}{1 + e^{-\frac{a_{i,l}-\beta_{i,l}}{\theta}}} + \sigma_{i,l} \tag{3.57}$$

where $\beta_{i,l}$ is a trainable unit-dependent bias, which allows the sigmoid to be shifted along the abscissa, θ is the smoothness, and the meaning of the remaining symbols is the same as that introduced in Section 3.2.

A mixture of 8 such MLPs (or *experts*) was used to realize a multiple-model regression for speaker normalization (refer to (Furlanello et al., 1997) for details on combining multiple regression models for speaker normalization). Credit assignment to individual experts was accomplished by partitioning the feature space into 8 phone-class dependent subregions, and assuming a Normal distribution of acoustic patterns within each class. Maximum-likelihood estimates of the parameters of the Gaussian densities were carried out on the training data. Bayesian posterior probability of a certain class given an acoustic observation was used to establish the weight (impact) of the observation on the overall cost function the minimization of which was pursued by the network associated with the class itself.

Speaker dependent versions of a speech recognizer based on Continuous-density hidden Markov models (CD-HMM) (Rabiner, 1989) were trained using the Baum-Welch algorithm (Rabiner, 1989) over the APASCI speech corpus (Angelini et al., 1994). A set of 34 context independent acoustic-phonetic speech units was adopted. Emission probability densities were modeled with mixtures of 16 Gaussian components, resulting in about 1,500 Gaussians over the whole system.

In order to train the normalization modules and to test the approach, novel speech material from 4 test speakers was collected. The recognition task was the dictation of fragments (chosen at random) of newspaper articles taken from the financial Italian journal *Il Sole 24 Ore*. It is a continuous speech task with large dictionary size (10,000 words). For each speaker, a test set of 30 utterances and an adaptation (training) set of 10 utterances were acquired, on different days, in an office environment. As in the case of speaker identification, 8 MFSCCs were used as acoustic features, along with the log-energy of the speech signal and their first and second order time-derivatives. The normalization modules were applied just on MFSCCs (performing a mapping between 8-dimensional spaces), while derivatives were computed from the transformed parameters and directly fed into the CD-HMMs.

Table 3.3 reports the results obtained on a test database (out of training sample) composed of 13 pairs (*new speaker*, *target speaker*), from the APASCI corpus. Results are expressed in term of Word Error Rate (WER), which is defined as $WER = 100(Ins + Del + Sub)/\mathcal{N}$, where Ins is the number of extraneous words inserted by the recognition system into the recognized text, Del denotes the number of omitted words, Sub represents the number of words substituted with wrong ones, and $\mathcal{N}$ is the overall number of words in the uttered text.

Regression model	*Average WER*
Baseline (SD recognizers without normalization)	35.0%
Linear multiple regression	20.7%
Mixture of 8 MLPs with fixed amplitudes	17.4%
Mixture of 8 MLPs with unique trainable λ	15.2%
Mixture of 8 MLPs with neuron-specific trainable λ_i	16.5%

Table 3.3: *Speaker normalization problem. Average WERs over the 13 pairs (new speaker, reference speaker), using the corresponding SD recognition systems, obtained with the proposed normalization modules.*

The average WER obtained with the SD recognition systems without any speaker normalization module is unfeasible for applicative purposes, as expected. Performance was improved with the adoption of a linear speaker normalization scheme, relying on multiple linear regression (estimated using standard SVD-based linear algebra techniques).

Let us consider the effect of speaker normalization accomplished via mixtures of neural networks. In all three cases, a WER reduction is gained with respect to the linear regression approach. Since the MLP architecture has only one layer of non-linear units, Cases 1 and 2 of Section 3.2 collapse into a single framework, that allows for a 12.64% relative WER reduction with respect to bare BP. In the speech community, such an improvement is considered indeed satisfactory. Case 3 of the algorithm compares favorably with the fixed-amplitude mixture of MLPs, but worsens the performance yielded by the unique-λ algorithm. From a theoretical point of view, a network with unit-specific trainable λ_is should always be able to fit the training data *at least* as well as the corresponding net with a single, adaptive λ (this could be trivially achieved by putting each λ_is equal to λ). Apart from the possible presence of newly introduced local minima of the cost functional, due to the augmented dimensionality of its domain, an attempt to explain the apparent contradiction might focus on the tradeoff between the number of parameters involved in the training process and the generalization ability of the learning machine. This concept is also related to the well-known *bias-variance dilemma* (Geman et al., 1992) to some extent. By introducing an adaptive amplitude for each unit of the network, we increase the number of parameters to be determined from the data by the learning procedure. Consequently, a reduction of learning and generalization abilities may occur, once the training set has been fixed.

3.3.3 Application to ASR: isolated-word recognition in a noisy environment

The experiment described in the present section starts by considering a particular ANN/HMM hybrid system (hybrid systems are presented in detail in Chapter 5), modifying the scope of the ANN within the paradigm, and applying Case 1 of the adaptive amplitudes to the connectionist model in order to increase its robustness. The hybrid under consideration, introduced in (Zavaliagkos et al., 1994), will be reviewed in Section 4.1.5. An ANN called Segmental Neural Network (SNN) is used therein for *rescoring* the N-best hypothesis of a HMM. The network computes scores on whole *segments* (sub-sequences) of frames, corresponding to phonemes, according to the segmentation provided by the underlying HMM. Segmental information is expected to reduce noise sensitivity. Since segments are made up of a variable number of frames, a discrete cosine transform (DCT) is applied to each segment, retaining as many parameters as it is necessary to fill the fixed-sized input layer of the ANN.

In the present framework the SNN is enhanced to provide robustness against noise by improving its generalization ability via application of Case 1 of the above algorithms. In addition, it is used as a stand-alone speech recognizer itself, instead of as a mere rescoring tool for a HMM. We call such a model *Segmental Neural Network with Trainable Amplitudes* (SNN-TA) (Trentin and Matassoni, 1999; Trentin and Matassoni, 2000). To summarize, differences between the standard SNN and SNN-TA are the following: (a) SNN yields scores for individual phonemes over acoustic segments provided by the HMM. SNN-TA provides a posterior probability estimation for each "word" of the dictionary to be recognized over the whole input acoustic sequence; (b) in (Zavaliagkos et al., 1994) SNNs are trained on a relative entropy criterion. Instead, backpropagation of squared errors between target and actual outputs is used herein, along with the gradient-descent algorithm to train amplitudes; (c) the SNN-TA is said to be "regularized", since a soft "self-regularization" technique that increases noise-tolerance is induced by the amplitude training algorithm. Figure 3.11 shows a schematic representation of the architecture of the SNN-TA.

There are basically three ways in which the SNN-TA can be used for ASR, either as a stand-alone paradigm or combined with a HMM into a hybrid system:

1. Recognition of isolated words: the whole input sequence, corresponding to an individual word in the dictionary, is fed into the SNN-TA via DCT. An output unit is introduced in the architecture for each word involved in the task. This can be seen as a pure connectionist model, or as a hybrid length-normalization (DCT)/ANN hybrid model.

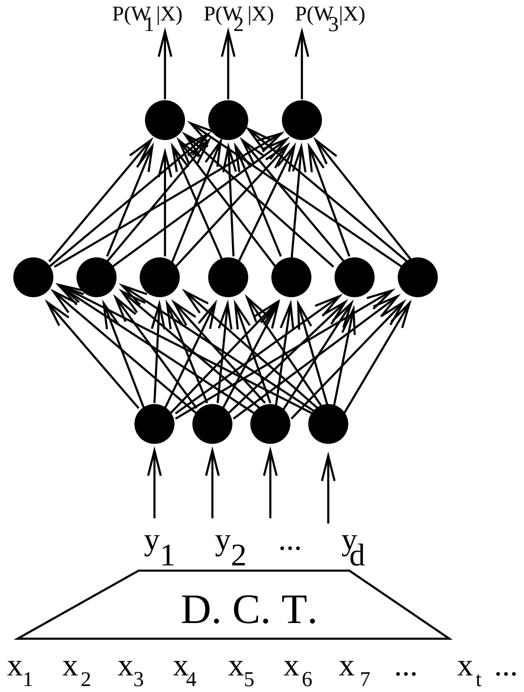

Figure 3.11: *Architecture of a SNN-TA. Amplitude of non-linear activation functions is learned from the data.*

2. Recognition of connected speech: the SNN-TA is trained over whole words, as in the previous case. A HMM specialized in acoustic frame *segmentation* is used along with the Viterbi algorithm (Rabiner, 1989) to segment the test utterances (built up of connected words from the given dictionary), and each segment is fed into the SNN-TA and recognized (or rejected, if the word probability estimates rest below a given threshold).

3. Continuous speech recognition: the *level-building* algorithm (Rabiner, 1989) is used to dynamically provide simultaneous segmentation and probability estimation relying on the SNN-TA, where the latter is trained over sub-word units, e.g. phonemes. During the level building procedure, basically all possible segmentation hypothesis (and corresponding scores provided by the SNN-TA over such segments) are evaluated to search for the optimal path.

In the present section we concentrate our efforts on an isolated words ASR task with noise. In the next chapter we point out that learning long-term time dependences within input sequences (e.g. in ASR) with conventional ANNs is difficult. This is a rationale behind the search for novel paradigms, aimed at combining ANNs and HMMs, that will lead us to the development (Chapter 5) of a new, robust ANN/HMM model for ASR.

In order to evaluate the robustness of the SNN-TA and to compare it with a HMM-based ASR system, a two-step experimental setup was designed. In the first step (training) the ANN and the HMM were trained on *clean* speech material from the *SPK* database (detail on SPK are given in Chapter 5), collected in laboratory conditions at ITC-irst. SPK consists of 100 speakers (male and female), each repeating the Italian digits for a total of 5000 isolated signals. Eight *Mel Frequency Scaled Cepstral Coefficients* (MFSCC) and the signal log-energy were extracted as acoustic features. The combined technique 2 described above (for connected speech) was applied, i.e. a specialized HMM was first used to segment each utterance into three segments, namely silence, actual speech, and silence again, respectively. Only the central segment was then fed into the SNN-TA.

In the second step (test), performance of the recognizers was evaluated on a noisy corpus, collected under the EC VODIS-II/SpeechDatCar projects (Matassoni et al., 1999) in a real car environment. In the present experimental setup, 150 speakers (male and female) were used. Recording sessions took place under several noisy car conditions. Each speaker uttered isolated Italian digits during two distinct sessions, for a total of 1435 utterances. Signals were acquired using a far (about 40cm) microphone placed in front of the driver, behind the sunvisor. The average Signal to Noise Ratio was about 7dB.

The ITC-irst CDHMM (De Mori, 1998) was used to provide a state-of-the-art comparison baseline for the SNN-TA. Training of CDHMMs was accomplished using the Baum-Welch algorithm (Rabiner, 1989), while the Viterbi algorithm was used for decoding. A set of 10 word units was modeled with left-to-right HMMs having 5 to 8 states. Emission probability distributions were modeled using mixtures of 16 Gaussian components per each HMM state, with diagonal covariance matrices, for a total of about 1,000 Gaussian components. In addition to the HMM trained over the 9-dimensional feature space defined above, a CDHMM trained over a 27-dimensional space obtained including also the first- and second-order time-derivatives of the MFSCCs and log-energy was used. Introduction of such dynamic features is intended as a compensation for the capability of the DCT to cover variations of the speech signal over time within each acoustic segment fed into the SNN-TA.

The architecture of the SNN-TA was based on a 2-layer Perceptron (MLP), with one hidden layer of 100 sigmoidal units (determined via cross-validation), and one output layer of 10 linear units (one for each digit to be recognized). The fixed-size input layer consisted of 9×10 units, since 9 was the dimensionality of the feature space (8 MFSCCs and log-energy), and the first 10 coefficients from the DCT (applied over each acoustic observation sequence) were retained for each of the features. The chosen sigmoid $f_i(a_i)$ for hidden unit i having input a_i was in the form $f_i(a_i) = \frac{\lambda}{1+e^{-\frac{a_i-\beta_i}{\theta}}} + \sigma$, where β_i is a trainable unit-dependent bias, and θ is the smoothness. The latter was empirically estimated by cross-validation: a value of 15.0 was chosen, corresponding to quite smooth activations. Using smooth sigmoids turned out to be useful in order to avoid the well-known phenomenon of *saturation*. The scalar λ controls the amplitude of the sigmoid. Finally, σ is an offset that shifts the values of the activation function along the ordinate axis, compensating for the actual value of λ. *Early stopping* relying on 20 training iterations over the whole training set was applied, including a momentum rate and an adaptive learning rate, to improve generalization properties.

Recognizer Type	*WER (%)*	*ERR (%)*
HMM	40.8%	23.8%
HMM with time-derivatives	26.3%	17.2%
DCT/MLP	14.4%	7.8%
SNN-TA	13.4%	7.0%

Table 3.4: *Word Error Rate (WER) and Equal Error Rate (EER) obtained on the recognition task with the different proposed approaches. DCT/MLP denotes the proposed connectionist model, but without application of the algorithm to train amplitudes.*

Table 3.4 shows the results obtained on the recognition task under consideration. The acronym DCT/MLP refers to the SNN-TA without application of the algorithm to train amplitudes. Results are expressed in terms of *Word Error Rate* (WER) and of *Equal Error Rate* (EER) (Trentin et al., 1995). Figure 3.12 represents the WER curves and the false rejection rate curves in the major cases of the HMM fed with time-derivatives of the acoustic features and of the SNN-TA, respectively. Results show that the connectionist models outperform the HMMs (a significant gain is provided by inclusion of dynamic features in the models), in terms of both WER and EER. In particular, while the CDHMMs do not allow for acceptable performance on the task, the EER for the SNN-TA may take practical relevance in a user perspective. It is also worth noticing that a significant 6.3% relative WER reduction over the bare DCT/MLP is provided by the SNN-TA, along with a 10.2% relative EER reduction. These results emphasize that the simple connectionist models appear to be definitely promising (and much more stable) in comparison with the much more complex HMMs in this kind of recognition tasks (i.e., with real, basically unpredictable noise). This behavior is a consequence of three facts: (1) taking into account whole acoustic segments provides information that is less sensitive to noise, in particular to instantaneous fluctuations, than modeling "local" acoustic frames; (2) ANNs trained with proper techniques (early stopping, momentum rate, adaptive learning rates, cross-validation) do generalize much better than HMMs over test acoustic conditions that significantly differ from training conditions; (3) finally, the proposed technique based on trainable amplitudes further improves generalization capabilities of the SNN, as expected.

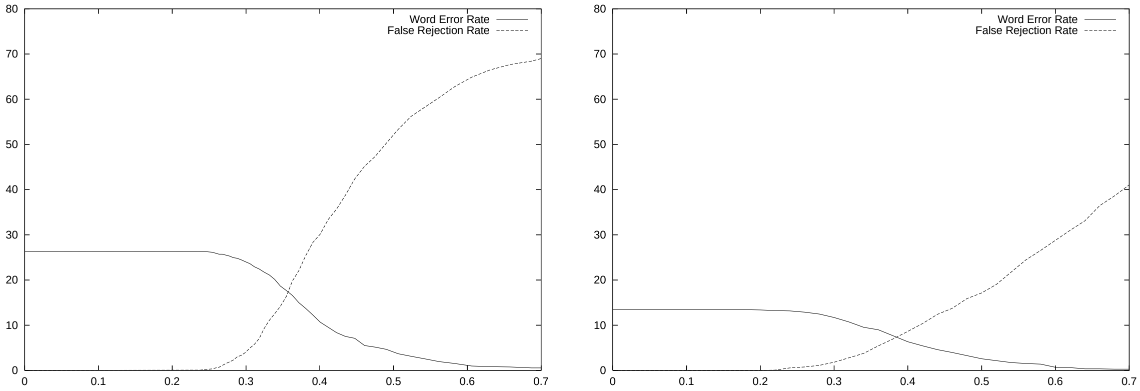

Figure 3.12: *Word Error Rate (%) curves and False Rejection Rate (%) curves, as functions of the rejection threshold, in the HMM case including time-derivatives of acoustic features (left), and in the SNN-TA case (right), respectively. In each graphic, the ordinate of the intersection between the curves defines the Equal Error Rate.*

3.4 Discussion

Surprisingly enough, little attention was paid so far to the problem of "learning" aspects concerning the activations in neural networks. Previous attempts mainly focused on the shape of sigmoidal functions, under well determined application-specific constraints.

This paper focused on learnable amplitudes of activation functions with arbitrary analytical form, a concept that turned out to be as simple as it is effective. Algorithms based on gradient-descent were developed for three main Cases (Section 3.2). The algorithms, the final form of which is indeed compact and easy to implement, rely on the notion of *auxiliary* and *expansion* functions, and were inductively proved (first two Cases); or on a back-propagation scheme that recalls the BP algorithm (third Case). The learning rules, being instances of an iterative gradient-descent technique, can be straightly enhanced by introducing a *momentum term* (Hertz et al., 1991).

Section 3.1 gave qualitative arguments in support of benefits that should be made available from the adoption of an automatic adaptation scheme for the amplitude of activation functions. At least four circumstances were listed under which, in literature, learning and generalization capabilities are improved by empirically tuning the value of amplitudes. It was also observed that modifying the original range of values of the nonlinearities turns out to be necessary sometimes, e.g. whenever the range of nonlinear outputs is not known a priori. The introduction of an algorithm for *learning* these amplitudes in accordance with the specific nature of the data at hand was then expected to further improve performance. The expectation was systematically confirmed through the experiments, accomplished on a demonstrative function approximation problem as well as on multivariate, highly non-trivial classification and regression tasks on real-world data. In particular, the automatic recovery from saturation of sigmoids at training time (by "inflating" the sigmoid until de-saturation occurs, and then possibly reducing the amplitude again) is a first, important rationale behind the improvement offered by the present algorithms over standard BP.

Nonetheless, a more analytical insight on why and how such improvements were obtained is required at this point, given that amplitudes *appear* to be absorbable by weights. In fact, as pointed out in Section 3.3.1, if a network solution to a given learning task is found using adaptive amplitudes then, for any connection having weight w and starting from a certain unit with amplitude λ, a corresponding connection having weight $\tilde{w} = w\lambda$ can be introduced into an equivalent standard network with fixed amplitudes and the same architecture. The latter net computes exactly the same mapping as the former. As a matter of fact, the claim of equivalence holds true only if linear output units (or, anyway, fixed-amplitude outputs) are present in the model; adaptive amplitude of output nonlinearities cannot be emulated by fixed-amplitude output nonlinearities, unless an extra layer of weights with linear outputs is added at the top of the network, which would nullify any reasonable attempt to establish a strong equivalence between the two models at training time. This peculiar but relevant difference should be kept well in mind in the following.

As anticipated in Section 3.1, the equivalence (whenever linear outputs are used) is out of discussion after the training is completed, and going back to standard feedforward nets with weights $\tilde{w}$ defined as above is exactly what is done in practice after having used the proposed algorithms, for instance to load the parameters of the trained model into the registers of a standard neural hardware to be exported in the applicative domain. But a shift of emphasis toward the training is necessary, since during training the equivalence does not hold true any longer: the algorithms to learn λ along with BP to learn w define a *different* learning scheme (i.e., different trajectories in the parameter space) with respect to the bare BP to learn $\tilde{w}$.

In Section 3.3.1 it was observed (equations (3.49) and (3.50)) that applying BP in conjunction with the present algorithms on the network with adaptive amplitudes λ and weights w implies, at each iteration, a double-step gradient descent on the hypothetical corresponding network with fixed amplitudes and weights $\tilde{w}$. This is basically a double-speed BP algorithm, and this provides us with a first hint of the rationale behind the tendency for increased convergence rate exhibited by the presented algorithms, but this may also lead to suspect that this is the main cause of improvement. The whole experimental corpus presented does prove the suspicion is not founded (even though an evident increase in convergence speed is an emerging benefit, especially in Case 3, as shown by the learning curves reported in Section 3.3), since the models with trainable amplitude tend to approach deeper minima of the cost function, along with increased generalization abilities on the test set, when compared with corresponding architectures with fixed amplitudes and BP accomplished for a doubled number of training epochs. A first consideration in this respect can be done by considering equations (3.49) and (3.50) again: given the learning rate η, the effect of an adaptive λ is indeed that of a double-step BP, but the quantity $\frac{\partial C}{\partial \tilde{w}}$ turns out to be "modulated" by novel learning rates $\eta\lambda$ (equation (3.49)) and ηw (equation (3.50)) which, in turn, can be thought of as ALRs as a direct consequence of the parallel training scheme for λ and w.

The fact that the present algorithms implicitly induce such an ALR within the training procedure was stated explicitly in equation (3.51). This formulation can be considered a different and otherwise interesting perspective on the proposed training scheme: temporary dropping the concept of trainable amplitude, the latter can thus be seen as a novel, effective algorithm that adapts the learning rate according to gradient-descent of the criterion function. Cases 1, 2 and 3 described in Section 3.2 are equivalent to having a single ALR over the whole network, a distinct ALR at each layer, or unit-specific ALRs, respectively. Benefits deriving from the adoption of ALRs are well-known from literature (see (Jacobs, 1988; Hertz et al., 1991) for relevant examples), so it is not surprising to observe improvements such as those reported in Section 3.3 in the present setup. Again, in the above experiments a direct comparison with a variant of the ALR technique described in (Hertz et al., 1991) was carried out, and the present algorithms yielded improved performance, allowing us to draw the following conclusions:

i. The effect of the algorithms described herein is not to (or, anyway, is not limited to) "correct" or compensate for possible wrong initial choices of the fixed learning rate η used in BP (since, otherwise, also the ALR technique used for the comparison would have possibly played the same compensative role).

ii. If these algorithms are thought of as a particular scheme of gradient-driven ALR, then the scheme is effective, performing better than that cited above.

Further support for this is brought in here by the experimentally favorable comparison with conjugate gradient training, which does not depend on any learning rate at all: this further suggests the opportunity of a shift in emphasis to other directions than the ALR viewpoint, to provide more insight on the behavior of the algorithms. In Section 3.2 it was observed that using adaptive amplitudes might also be interpreted as a particular "soft" *weight grouping* technique: all the connections that are subject to a given λ are somewhat "grouped" during training, and the λ factor tends to scale the values of the weights associated to connections belonging to a given group. In addition to two basic classes, namely that of weights which are not subject to any adaptive amplitude (e.g. connections starting from linear units) and that of other free parameters that are not properly weights (e.g. biases of sigmoids), grouping into one more class, containing all the connections present in the topology of the network that are subject to variable amplitude, is obtained whenever Case 1 of the algorithms is used. Case 2 yields one more group per each layer containing units with trainable amplitude and, finally, Case 3 adds a group per neuron with adaptive amplitude, consisting of all the connections that start from the neuron itself. An idea about the way the grouping is accomplished can be obtained by considering again equation (3.50), in which gradient-descent with respect to the amplitude λ prescribes to augment it or to reduce it, according to average disagreement vs. agreement in sign among the weights in $W(\lambda)$ and partial derivatives of C with respect to the weights. From a certain point of view, the most evident difference with respect to other weight grouping techniques (such as hard weight sharing) comes from groups being built up of connections coming from specific units, rather than unit-independent, sparse ones. As a matter of fact, this topic requires a much deeper insight. A brief review of main weight grouping concepts is therefore needed at this point.

Weight grouping was originally introduced in its simpler form, known as *weight sharing*, by (Rumelhart et al., 1986a; LeCun, 1989; LeCun et al., 1990). The basic idea is that of reducing the complexity of a model, namely the number of free parameters, by imposing that subsets of the connections (not limited to connections coming from specific units) do share a common value for the corresponding weights. In so doing, reduction of overfitting risk and improvement in generalization ability are expected. The shared weight is trained via standard BP and the approach was proven to work under many circumstances, but it requires *a priori* knowledge of specific properties of the problem at hand, to determine exactly which connections can reasonably share their weight value. *Soft weight sharing* (SWS) (Nowlan and Hinton, 1992) overcomes this major limitation, removing the requirement of prior hard grouping and allowing for adaptive grouping of weights within Gaussian clusters. The parameters of the clusters, i.e. their mean and variance vectors, can be adapted during training. The Bayesian framework described by MacKay (MacKay, 1992b) provides us with a probabilistic interpretation of weight grouping schemes corresponding to different choices of regularizers introduced in the learning algorithms,

The idea underlying SWS is an extension of weight decay (Plaut et al., 1986), a well-known *regularization* (Bishop, 1995) technique that provides a control over the complexity of a model by attempting to keep the weights as small as possible in order to avoid overfitting, implicitly adding a "penalty" term to the criterion function. This (implicit) regularizer has the form $\alpha \sum_i \frac{1}{2} w_i^2$, where the sum is extended to all the parameters of the network and α is a given regularizing positive constant. Instead of considering the sum of squared weights, it is reasonable to hypothesize that weights are drawn from a Normal distribution with zero mean and variance $\frac{1}{\alpha}$ (Nowlan and Hinton, 1992), thus redefining the penalty term as the negative log-likelihood of the Normal probability density function (pdf). The basic idea is first extended by taking into consideration, instead of a single Normal, a simple mixture of two Gaussian components both having zero mean, one narrow and peaked around zero, the other broad and flat (a uniform distribution in the limit case). The former takes responsibility for the smaller and less *relevant* weights (which may eventually drop to zero to prevent affecting the generalization capabilities of the network), while the latter takes responsibility for larger weights, which are expected to have significance in modeling the data. The opportunity of sharing similar values within groups of connections, as in weight-sharing, is reintroduced by extending the mixture to n Gaussian components with independent adaptive parameters, where the weights spontaneously tend to form clusters as training proceeds.

The present algorithms to learn amplitudes actually realize, in a broad sense, a kind of weight grouping, where a specific λ factor modulates or scales the weights of the connections that are subject to its influence, but they do not imply any *de facto* sharing of specific values among different connections, neither imposed *a priori* (weight sharing) nor adaptively tuned (SWS): the "effective" value $\tilde{w}$ is finally obtained by multiplying the shared amplitude λ by the non-shared individual weight w, and no regularization or sharing is imposed on w here. Indeed, connection weights subject to a certain amplitude λ may turn out, at the end of training, to differ heavily and even to have opposite signs.

The present algorithms themselves cannot be thoroughly considered as regularization techniques, e.g. no penalty term (regularizer) is introduced in the criterion function - although the side effect of self-pruning that can be found in Case 3 may be viewed as an implicit regularizing effect at the architectural level, since it simplifies the model by removing redundant units, possibly allowing for a better generalization.

Differently from SWS, in this paper no *a priori* (although reasonable, under many circumstances) assumptions on the form of parameter distribution in weight space, in the line of "weights are drawn from a mixture of n Gaussian pdfs" or similar, are taken. Furthermore, focus is on units, rather than on connections; or, otherwise said, on connections that come out of a specific unit, or layer, or over the whole net: no sparse groupings are allowed. Of course, the latter may be considered a limitation from the grouping point of view, but it is just an obvious consequence of the particular framework investigated herein.

The discussion concerning SWS developed so far, where weight grouping emerges as a consequence of the choice of a given regularizer, suggests taking into consideration the fundamental work by MacKay on Bayesian framework for learning in neural networks (MacKay, 1992a; MacKay, 1992b; Bishop, 1995). The framework allows, among other things, comparing different regularizers on an objective basis, each regularizer determining a particular weight grouping scheme. Looking at the algorithms presented in this paper in light of conclusions drawn by MacKay turns out to be insightful, because of two main reasons. First, some behaviors described here actually operate architectural simplifications of the model, seemingly involving an implicit self-regularizing criterion. Second, overall, a similarity appears to hold between grouping of weights in classes characterized by ranges of values distributed according to different variances (or "amplitudes"), as suggested in (MacKay, 1992b), and grouping into classes of connections the value of which is scaled by a certain amplitude λ of certain activation functions. The similarity between the present definition of "adaptive amplitude" and that of variance of prior probability distribution of a class of connections is obviously only conceptual and not quantitative (unless a formal proof of a quantitative relationship is given, which is beyond the scope of this paper). The conclusions drawn in (MacKay, 1992b; Bishop, 1995) that are relevant to the present discussion are now summarized. Next, analogies and differences between those conclusions and the scenario presented herein will be outlined.

Investigating different regularization techniques, (MacKay, 1992b) and (Bishop, 1995) show that the classic regularizer $\alpha \sum_i \frac{1}{2} w_i^2$ is scarcely supported in the Bayesian framework, it has low *evidence*, while evidence is high for the regularizer $\sum_{c=1}^{K} \alpha_c \sum_{j \in c} \frac{1}{2} w_j^2$, where K classes of weights are introduced, each one with its own regularization term and constant α_c. As in the former case the hypothesis of weights drawn from a Normal distribution with zero mean and variance $\frac{1}{\alpha}$ is met, in the latter case a distinct, zero-mean Normal distribution with variance $\frac{1}{\alpha_c}$ for each class c is assumed. Considering a 2-layer MLP with sigmoidal hidden units and linear outputs, for instance, evaluation of Bayesian evidence shows that a good partition of the connections into classes is obtained via the following 3-class grouping: (1) hidden weights, i.e. weights associated to connections from the input to the hidden layer; (2) hidden biases, namely biases of the sigmoids in the hidden layer; (3) output weights and output biases. This partition is pairwise aligned with the intuitive idea that parameter values for hidden units cannot be scaled within the same range that is used for output units, etc. Extension to different architectures is rather straightforward

Since Bayesian evidence is a function of the dataset at hand, in the following we are limited to qualitative arguments. Given that the present algorithms do not explicitly affect biases[6] and that they are possibly limited to the amplitude of non-linear activation functions, then it can be immediately seen that Case 2 (as well as Case 1, in the 2-layer architecture) actually induces the same partitioning in the architecture at hand (once biases of sigmoids are trained separately from weights, i.e. with their own "knobs setting", as is common and usually necessary practice), since connections that start from the input units (which act as placeholders, without performing any transformation over the input features) are not subject to any amplitude at all.

Case 3 represents a specialization of Cases 1 and 2: Case 3, and Case 2 in more-than-2-layer nets, do actually induce a more-than-3-classes partition of the parameters, which is a refinement of the grouping induced by Case 1, i.e. each class yielded by Case 1 can be obtained by joining separate classes induced by Cases 2 or 3. Indeed, (MacKay, 1994) introduces a high-evidence multi-class partitioning by separately grouping weights of connections starting from different input units. This grouping is achieved by further backpropagating Case 3 to input units, considering them as standard linear neurons instead of mere placeholders. Case 3 performed

[6]In fact, in standard BP biases can be trained by introducing an extra *diagonalization* unit with unity output, connected to all the units that require an adaptive bias, and the values of the corresponding weights actually play the role of biases. If the amplitude of the diagonalization unit were made adaptive, Case 3 could be used to scale biases, thus realizing explicit distinct groupings for biases.

better than the others in the teacher-student problem and in the speaker identification task, while slightly lowered the performance in the speaker normalization problem. This reveals different degrees of consistency of Case 3 with respect to different datasets; only an explicit evaluation of Bayesian evidence would allow us to draw conclusions in this respect.

In summary, whenever interpreting the schemes proposed herein to learn amplitudes as a particular weight grouping technique, ideally determined by the introduction of as many regularization constants as the effective number of adaptive amplitudes taken into account (one, one per layer, or one per unit), there is no contradiction with the results yielded by Bayesian evidence. It should be clear that the low evidence exhibited by the regularization term $\alpha \sum_i \frac{1}{2} w_i^2$ is a consequence of the choice of a common Gaussian prior for *all* the weights in the network, and this is definitely not the case herein. It could be hypothesized that if the present algorithms would have violated the conclusions derived within the Bayesian framework to some extent, it would have been very difficult to justify the improvement in generalization ability that was found in the experimental series discussed in Section 3.3, even though there is no lack of historical examples - e.g. weight-decay and SWS - where, in spite of limited Bayesian evidence, widely experienced good practical behavior on real world data was obtained. The latter aspect is crucial in a perspective of evaluation of the present algorithms in a hypothetical Bayesian framework: although the fact that the low evidence induced by the choice of the regularizer $\sum_i \frac{1}{2} w_i^2$ may suggest that, for instance in more-than-2-layer nets, the adoption of a single λ (Case 1) could possibly be inconsistent from the Bayesian viewpoint, nonetheless its experimentally confirmed practical benefit and its simplicity would remain, in our opinion, relevant supporting arguments.

Note that, according to (MacKay, 1992b), if a network with weights $\tilde{w}_j = \lambda w_j$ is considered, then for the c-th weight group the equation

$$2\alpha_c \sum_{i \in c} \tilde{w}_i^2 = \gamma_c \tag{3.58}$$

should be satisfied by choosing the "optimal" (more probable) regularizing constant α_c according to the effective number γ_c of well-measured parameters for class c given the data at hand. Assuming that, as in the present algorithms, c-th group of weights is characterized by a specific value λ_c of amplitude, equation (3.58) can be rewritten as:

$$2\alpha_c \sum_{i \in c} (\lambda_c w_i)^2 = \gamma_c \tag{3.59}$$

and, finally:

$$2\alpha_c \lambda_c^2 \sum_{i \in c} w_i^2 = \gamma_c. \tag{3.60}$$

If Case 3 of the algorithms is considered, a class of weights exists for each non-linear unit with adaptive amplitude in the network (plus, possibly, classes for biases or other function-dependent parameters, and for linear activations), each one with its own amplitude λ_c; if, given the training data, c-th unit may be assumed to be redundant, then γ_c turns out to be zero, and equation (3.60) is certainly satisfied for $\lambda_c = 0$, the latter being the value yielded by the automatic pruning effect of redundant units. Roughly speaking, the pruning effect ensures respect of conditions on the number of well-measured parameters for each class given the data, and it appears thus justified within the Bayesian framework. In passing, let us observe that the pruning mechanism is somewhat complemental to the unit-pruning approach introduced by (Mozer and Smolensky, 1989): in fact, the latter multiplies the input (instead of the output) of i-th unit by a factor λ_i, which allows controlling the *relevance* of the unit itself, the relevance being approximated by using partial derivatives of the criterion function with respect to λ_i.

A version of the algorithms suitable for partially recurrent networks (Hertz et al., 1991; Haykin, 1994) was also developed, and successfully applied in speaker normalization with mixtures of recurrent nets for a speaker-independent recognition system (Trentin and Giuliani, 2001). In general, the present algorithms may be easily extended to generic recurrent neural nets, whenever BPTT (see Chapter 2) is used: once the net has been unfolded in time, λ's are trained as usual, exactly like connection weights via BP, over the unfolded architecture and, eventually, collapsed into their "average" values.

Chapter 4

Hybrid ANN/HMM Models for Automatic Speech Recognition

As pointed out in Chapter 1, ANNs substantially failed as a *general* solution to the ASR problem, due to their strong limitations in modeling long-term time dependencies. As a consequence, a variety of different hybrid ANN/HMM architectures, along with novel training algorithms, have been proposed in literature. This chapter reviews a number of significant hybrid models for ASR, putting together approaches and techniques from a highly specialistic and non-homogeneous literature. Efforts concentrate on describing and referencing architectures and algorithms, their advantages and limitations, as well as on categorizing them into broad classes. Early attempts to emulate HMMs by ANNs are first described. Then we focus on ANNs to estimate posterior probabilities of the states of a HMM and on "global" optimization, where a single, overall training criterion is defined over the HMM and the ANNs. Connectionist Vector Quantization for discrete HMMs, and other more recent approaches are also reviewed. It is seen that, in addition to their theoretical interest, hybrid systems have been allowing for tangible improvements in recognition performance over the standard HMMs in difficult and significant benchmark tasks. In spite of that, there is still a lack for a general framework capable of combining the two paradigms within a homogeneous, globally-optimized framework suitable for robust speech recognition in noisy environments.

4.1 State of the Art of Hybrid ASR Systems

Whereas standard HMMs have a consolidated and homogeneous theoretical framework, hybrid ANN/HMM systems are a recent research field with no unified formulation. A variety of different architectural and algorithmic solutions have been proposed in the literature. The following Sections are organized according to four major categories:

1. *Early attempts.* Early approaches (between the late Eighties and the beginning of the Nineties) relied on ANN architectures that attempted to emulate HMMs (Bridle, 1990; Niles and Silverman, 1990). Some of them incorporated the Dynamic Programming (DP) algorithm (Bengio, 1996) within the ANN itself (Haffner et al., 1991; Levin et al., 1992). These approaches strengthened the idea that ANNs could be effectively used for ASR, but the straight emulation of standard HMMs did not allow to overcome limitations of the latter (such limitations were summarized in Section 1.2).

2. *ANNs to estimate the HMM state-posterior probabilities.* Some ANN/HMM hybrids assume that the output of an ANN is sent to a HMM for ASR (Levin, 1990; Franzini et al., 1990; Haffner et al., 1991; Tebelskis et al., 1991). The architectures proposed by Bourlard, Morgan *et al.* rely on a probabilistic interpretation of the ANN outputs (Bourlard and Morgan, 1994a; Morgan et al., 1995). Each output unit of the ANN is trained to perform a non-parametric estimate of the posterior probability of a CDHMM state given the acoustic observations (and, possibly, given the previous state). This represents a fundamental class of hybrid models, which had a strong influence over a number of following approaches. Points of strength of this paradigm are the relative ease of implementation, due to a training scheme that rely on alternating between BP and a standard Viterbi alignment, as well as a discriminative training criterion that may allow for improved recognition performance. Weakness of the approach comes from the unavailability of an analytically motivated global optimization scheme defined at the whole-system level, and also from the requirement of huge ANNs architectures (possibly with millions of connection weights) to be trained for the most complicated ASR problems, e.g. SI continuous speech recognition with large vocabularies.

3. *Global optimization.* The ANN and the HMM are often trained separately, but techniques have also been proposed in which a simultaneous training is accomplished. In (Bengio, 1996; Bengio et al., 1992b), the ANN is used to transform a vector of acoustic features into more effective observations for CDHMMs, according to a "global optimization" of the parameters of the combined system. Apart from the feature extraction aspect, what is more relevant in this case is the introduction of an analytically motivated training scheme aimed at the optimization of a global criterion function, the latter being defined at the whole-system (i.e., recognition) level. Other researchers proposed different globally-optimized hybrids in recent years. A brief comparison of hybrid architectures relying on global discriminative training algorithms can be found in (Johansen, 1996).

4. *Networks as vector quantizers for discrete HMM.* As mentioned in Section 1.2, discrete HMMs assume that a finite alphabet of input symbols has to be modeled. Since in ASR the assumption does not hold, a *quantization* in

the acoustic feature space is required. Instead of using standard clustering algorithms, unsupervised ANNs can be effectively used as vector quantizers. This class of hybrids is characterized by distinct training steps for the ANN and for the HMM. Lack of a combined, global optimization scheme is compensated by the reduced complexity of the overall machine (mainly due to the ease of use of discrete HMMs versus CDHMMs) and by good recognition performance, the latter being close to that yielded by CDHMMs in some small-scale ASR tasks. Examples can be found in (Kimber et al., 1990; Iwamida et al., 1991; Rigoll, 1994).

5. *Other approaches.* Along with the architectures introduced so forth, during the Nineties several researchers proposed hybrid systems relying on particular combination techniques between ANNs and HMMs. Such techniques do not properly belong to any of the previous categories, and often focused on specific tasks, e.g. *rescoring* or *word-spotting.* In (Bengio, 1996; Haffner et al., 1991; Tebelskis et al., 1991), the ANN outputs are interpreted as "scores" which are used within a DP algorithm to perform alignment and segmentation. As an alternative, the ANN can be used for re-scoring the N-best hypotheses produced with an HMM (Zavaliagkos et al., 1993). Although a uniform treatment of these approaches does not appear feasible, Section 4.1.5 provides a summary of major architectures, algorithms and experimental results reported from literature.

4.1.1 Early attempts (Viterbi Net, Alpha Net)

In 1987 Lippmann and Gold (Lippmann and Gold, 1987) proposed a recurrent neural network architecture, implemented in VLSI, able to mimic the decoding behavior of the continuous-density Viterbi algorithm, which was for the recognition of isolated words. It was called the Viterbi net. Although the recognition performance did not represent any improvement with respect to what could be achieved with standard HMMs, this connectionist architecture is remarkable for historical reasons.

The structure of the Viterbi net has as many input units as the dimensionality of the acoustic vectors. Acoustic observations are fed into the network in sequence, one at a time. Inputs are propagated forward through a single, fully-connected layer, and summed up before being passed to each one of a set of state-units representing the states of a corresponding left-to-right HMM. A Viterbi network is built for each word model of the HMM. The state-units have a threshold-logic activation and a fixed delay on the output, and are laterally connected (each of them to the following one) in a way that resembles the topology of the left-to-right HMM. In addition to the summed inputs, each state-unit also receives a feedback input from an associated subnetwork, which is able to select the maximum between the output from the state-unit itself and the output from the adjacent state-unit on the left.

There is no actual training procedure for the Viterbi net, and this is one of its major limitations. It is initialized using the parameters of the corresponding HMM, obtained using the Baum-Welch algorithm. After the initialization, when fed with a sequence of input vectors, the recurrent dynamics of the net result in a parallel version of Viterbi decoding, producing as the final output the logarithm of the likelihood of the input sequence (and most likely state path) given the model.

Experimental results on isolated word recognition tasks, performed using 12 *MEL* cepstral coefficients (Davis and Mermelstein, 1980) and 13 differential MEL cepstra as features, were satisfactory and comparable with those obtained with state-of-the-art HMMs.

In 1990 John S. Bridle (Bridle, 1990) proposed a connectionist architecture which was able to behave like a HMM for ASR. The idea underlying his approach was to look at the forward and backward computation of the probabilities in HMMs, and to give them an interpretation in terms of a neural network. The model was called *Alpha Net*, because of the way its architecture and dynamics were calibrated to resemble the forward computation of the alphas in the Baum-Welch algorithm.

The Alpha net is a recurrent neural network. As for the Viterbi net, its parameters are the same as those of the corresponding HMM, but a complete forward estimation of the likelihood of the observations given the models is accomplished, instead of a single best-path search as occurs with the Viterbi algorithm. Furthermore, a learning procedure, derived from the backward step of the Baum-Welch algorithm, is available. This procedure is based on a discriminant cost function (maximum mutual information) and backpropagation of its partial derivatives. A recurrent architecture is built for each acoustic unit (word) to be included in the acoustic model. The neurons are organized in order to represent the states of the HMM. For instance, in the most common case of left-to-right HMMs, each unit (neuron) is connected with a recurrent connection to itself, and with a forward connection to the unit representing the following adjacent state in the HMM. The weights of these connections are equal to the state transition probabilities between the corresponding pairs of states. The likelihoods of the emission probabilities are fed into the recurrent loops from another, distinct part of the network, and are multiplied instead of summed. The units are linear, with a unit delay, thus resulting in the computation of the product of the joint probability of transition and emission for that state at each time step. In so doing, the overall behavior is consistent with the probabilistic framework of the HMM. The separate network that is responsible for the computation of these likelihoods is supposed to rely, for example, on multipliers and exponentials in order to approximate as closely as possible the likelihoods generated by the Gaussian or mixture of Gaussians associated to the states of the corresponding HMM. The final output of such an architecture is the actual likelihood of the input acoustic sequence given the model, summed over all possible paths within the network.

4.1.2 ANNs to estimate the state posterior probabilities

Bourlard *et al.* (Bourlard and Wellekens, 1990; Morgan and Bourlard, 1990; Bourlard and Morgan, 1993; Bourlard and Morgan, 1994a) proposed ANN/HMM hybrids for continuous ASR in which a MLP was trained to estimate the posterior probabilities of HMM states, with the ultimate objective of maximizing the posterior probability of a given (left-to-right) Markov model M_i given an acoustic observation sequence X. Posterior probabilities can be written as:

$$\begin{aligned} Pr(M_i \mid X) &= \sum_{q_1^L} Pr(q_1^L, M_i \mid X) \\ &= \sum_{q_1^L} Pr(q_1^L \mid X) Pr(M_i \mid q_1^L, X) \\ &= \sum_{q_1^L} Pr(q_1^L \mid X) Pr(M_i \mid q_1^L) \end{aligned} \tag{4.1}$$

where the model M_i is supposed to have Q states $S_1, ..., S_Q$, and the acoustic observation sequence $X = (\mathbf{x}_1, ..., \mathbf{x}_L)$ is assumed to be of length L. In Equation (4.1) the sums are extended over all possible sequences q_1^L of states.

The quantity $Pr(M_i \mid q_1^L)$ does not depend on the acoustics (observation sequence X), but only on higher-level choices made in the definition of the models and can thus be computed separately.

Repeatedly applying the properties of joint probabilities, Equation (4.1) can be rewritten as:

$$\begin{aligned} Pr(M_i \mid X) &= \sum_{q_1^L} Pr(q_1 \mid X) Pr(q_2 \mid X, q_1) \ldots \\ &\quad \ldots Pr(q_L \mid X, q_1, ..., q_{L-1}) Pr(M_i \mid q_1^L) \\ &= \sum_{q_1^L} \{ \prod_{\ell=1}^{L} Pr(q_\ell \mid X, q_1^{\ell-1}) \} Pr(M_i \mid q_1^L). \end{aligned} \tag{4.2}$$

Attempts to determine analytical developments for the present formulation, similar to those adopted in the Forward-Backward algorithm for maximization of the likelihood of the observations given the model, are not practicable. This is due to the constraint $\sum_i Pr(M_i \mid X) = 1$ that must be satisfied within the present *Maximum A Posteriori* discriminant framework. Bourlard and Morgan's idea was then to use feedforward neural networks as estimators of the posterior probabilities of states given the observations and the previous state sequence. In so doing, advantages are taken from ANNs discriminant training and from their capability to estimate Bayesian posterior probabilities when trained by BP on a Mean Squared Error (MSE) criterion. The basic architecture is schematically represented in Figure 4.1.

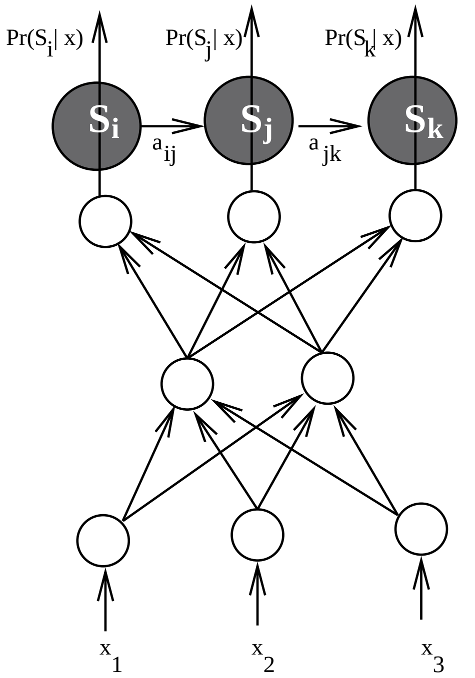

Figure 4.1: *Basic hybrid architecture where a 2-layer feedforward ANN estimates the posterior probabilities of states* S_i, S_j, S_k *of a left-to-right HMM given a hypothetic acoustic observation* $x = (x_1, x_2, x_3)$.

Actually, approximate versions of Equation (4.2) are used, e.g. by taking:

$$Pr(M_i \mid X) \approx \sum_{q_1^L} \{ \prod_{\ell=1}^{L} Pr(q_\ell \mid \mathbf{x}_{\ell-k}, ..., \mathbf{x}_{\ell+k}, q_{\ell-1}) \} Pr(M_i \mid q_1^L) \quad (4.3)$$

that is to say, the network is trained to estimate the state posterior, called *conditional transition probability*, $Pr(q_\ell \mid \mathbf{x}_{\ell-k}, \ldots, \mathbf{x}_{\ell+k}, q_{\ell-1})$ given a fixed number $2k+1$ of acoustic vectors $\mathbf{x}_{\ell-k}, \ldots, \mathbf{x}_{\ell+k}$ (a *window* or *context* of size k centered in the current acoustic observation $\mathbf{x}_\ell$) and the previous state. This is accomplished using the BP algorithm on a MLP, which has an output unit for each state, that represents the estimate of the state posterior probability. Other attempts were made to use the MLP to estimate the posteriors in different ways, for example as a function of the current observation and of the previous state only, or as a function also of previous MLP outputs (previous states). In any case, the speech recognition performance of the resulting overall system was surprisingly poor in many experiments (Bourlard and Morgan, 1994b). This was attributed to a mismatch between the priors estimated from relative frequencies over the training data, and priors implicitly constrained by the topology of the models. A step backward was then made by moving the system back to likelihoods. This issue was pursued by using a somewhat standard version of the HMM, in conjunction with neural networks. The latter were trained to perform exactly the same probability estimation as in Equation (4.3), but with their outputs divided by the *a priori* probabilities of the corresponding states, in order to reduce probabilities to scaled likelihoods, normalized by the unconditional likelihood of each observation (using Bayes' theorem). Priors can be computed apart, from the training data or from statistical considerations on the constraints given by the specific task. This solution was effective and allowed the system to reach the recognition performance of state-of-the-art HMM recognizers on large vocabulary continuous speech tasks, but at the expense of the original theoretical framework.

One central point raised with this approach concerns the training procedure. Indeed, networks are trained by BP, which would require knowledge of target values for the outputs in order to compute the gradient of the cost function. With the exception of toy tasks, no supervised labeling of acoustic frames in terms of HMM states is actually available (labeling by hand in real-sized databases is not feasible). Bourlard *et al.* suggested an iterative training procedure that starts up with an initial segmentation of the acoustic observations, performs training of the networks according to that segmentation, then uses the Viterbi algorithm, in conjunction with the newly trained networks as estimators of the state-posterior probabilities, in order to produce a new and more reliable segmentation of the data, that in turn is used to train again the networks, and so on in an iterative fashion. The initial segmentation may be obtained using a standard HMM, or by dividing the observation sequence into equally-sized segments; each segment is associated to the corresponding state in the correct HMM state-sequence. An analogous training scheme was proposed also in (Franzini et al., 1990).

The approach by Bourlard *et al.* was successfully adopted in (Renals et al., 1994) for the DARPA Resource Management (RM) SI, continuous speech task, with a vocabulary size of 991. A HMM with 2 or 3 states left-to-right phone models was used, with parameter tying of the pdfs associated to the states of a given model.

The performance of the hybrid system was compared with that of Decipher, a HMM-based recognizer where emission probabilities are modeled with mixtures of Gaussian components. Decipher was also used to bootstrap the hybrid, i.e. to obtain the initial segmentation of acoustic observations used for the first training step of the MLP.

Experimental results obtained averaging on the three datasets used for testing, using 12 MEL plus energy and the corresponding first-order time-derivatives and a word-pair grammar (that reduced the perplexity to 60), yielded an average error of 8.7% for the context-independent hybrid, a 7.0% for the context-dependent HMM and a 5.5% for the interpolation of the two models. Moreover, results on one of the test sets (February91) showed considerably better performance of the context-independent hybrid (5.8%) with respect to the context-independent HMM (11.0%).

(Singer and Lippmann, 1992) used Radial Basis Function (RBF) (Powell, 1987) networks instead of MLPs as Bayesian probability estimators; the resulting hybrid was used in an isolated word recognition task.

Recently, (Hennebert et al., 1997) have proposed a reinforcement of the theoretical framework originally formulated by Bourlard, Morgan et al., by generalizing the local connectionist estimates of posterior probabilities to global posterior of models, formulating also a novel training algorithm within that framework.

An extension to the hybrid paradigm based on connectionist estimation of state posterior probabilities was proposed by (Franco et al., 1994), where the basic context-independent HMM scheme was replaced by a model capable to take into account acoustic contexts. The introduction of context-dependency is accomplished using Bayes' theorem to obtain the following factorization:

$$p(X \mid q_j, c_k) = \frac{Pr(q_j \mid X, c_k)p(X \mid c_k)}{Pr(q_j \mid c_k)}, \tag{4.4}$$

where X is an acoustic observation, q_j is a state of the HMM, and c_k is a certain context ($k = 1, \ldots, K$).

In Equation (4.4) the quantity $Pr(q_j \mid X, c_k)$ is estimated by an MLP. Bourlard *et al.* used a single MLP to estimate all state posteriors $Pr(q_j \mid X)$,$q_j \in S$; (Franco et al., 1994) adopts K MLPs if K is the number of contexts to be modeled. The k-th MLP is trained as an estimator for $Pr(q_j \mid X, c_k)$, for all $q_j \in S$, applying BP on the feature vectors embedded within k-th context in the training sequences.

The quantity $p(X \mid c_k)$ is expanded using Bayes' theorem again, writing a further factorization of the form:

$$p(X \mid c_k) = \frac{Pr(c_k \mid X)p(X)}{Pr(c_k)} \tag{4.5}$$

where $Pr(c_k \mid X)$ is estimated by an MLP in Bourlard's way. The other quantities involved in Equation (4.5) are directly computable from statistical evaluations (analysis of relative frequencies) on the training data, otherwise they come out to be irrelevant in Viterbi search for the optimal path.

(Franco et al., 1994) used 2-layer, context-dependent MLPs with sigmoidal activations. The connection weights between input and hidden layer were shared among all the networks associated to a given phone class, independently of the context $c_k, k = 1, \ldots, K$, and only the weights of connections between hidden and output layers were specialized on the specific context. In so doing, the complexity of the system (number of free parameters) was considerably reduced, thus limiting the effect of reduction of available training data for individual networks as the number K of contexts increased. A novel training procedure was introduced for this architecture.

Experimental results, carried out on a continuous speech, SI task (DARPA RM) using generalized biphone phonetic contexts, showed a 28.0% relative WER reduction with respect to the context-independent hybrid. When adopting a word-pair grammar and averaging over three different test sets, performance improved from 8.8% WER to 6.3% WER, whereas a standard context-dependent HMM scored a 7.0% WER.

Connectionist estimate of posterior probabilities is also adopted in (Franco and Digalakis, 1995), where the estimation is accomplished using a MLP which takes in input a time-window of previous acoustic observations, in addition to the current acoustic vector, so as to take the correlations among adjacent observations into account.

Robinson *et al.* (Robinson, 1994; Hochberg et al., 1994; Hochberg et al., 1995) extended Bourlard's approach with the introduction of a recurrent network instead of a static MLP to estimate state-posteriors. Their system, called ABBOT, successfully participated to November 1993 ARPA Wall Street Journal Test. ABBOT is a continuous speech, SI recognition system for large vocabularies (more than $10,000$ words). In (Hochberg et al., 1994) the basic system was improved by providing:

1. *Combination of neural models.*
   Different recurrent nets are trained on different acoustic features, namely MEL and Perceptual Linear Prediction (PLP) (Hermansky, 1990) coefficients. Furthermore, "backward recurrent" networks (i.e. models that start processing from the end of an input sequence back to the beginning) are introduced in addition to the conventional "forward recurrent" models, thus resulting in 4 parallel probability estimators. These models are then combined, either in a linear fashion (averaging on the corresponding probability estimates), or averaging in the log-domain (the latter choice allowing for a further gain in performance).

2. *Introduction of a "posterior-directed path pruning".*
   Pruning of the search space is accomplished relying, for each new input frame, on the connectionist probability estimates of individual phones. Paths containing phones the posterior probability of which is below a given threshold, are immediately pruned.

Further improvements in performance for the ABBOT system are presented in (Hochberg et al., 1995), which discusses more sophisticated combination strategies for the recurrent models and improved techniques for phone-duration modeling.

Another approach which is conceptually analogous to that by Bourlard *et al.* is presented in (Yu et al., 1994), which does not use a HMM explicitly, but a Viterbi alignment strategy is rather applied on state scores computed as the output values of the second hidden layer of a feed-forward ANN.

A variant on hybrid systems where the ANNs are used as posterior probability estimators is described in (Chen et al., 1996). The output of the networks is interpreted as a discriminant function (for instance, applying Bayes' theorem and assuming that all the states of the HMM share the same prior probability). In this perspective, the approach of (Chen et al., 1996) consists in training MLPs as "class recognizers", relying on the discriminative training capabilities of feedforward nets, combining then the MLPs into a sequence (Sequential Multilayer Perceptron: SMLP) with the aim of modeling and recognizing isolated words, within a DP scheme.

The system models the words in the vocabulary with such a sequence of MLPs, using a left-to-right chain of as many nets as the number of states in the corresponding underlying HMM. Each net has an output unit for each of the classes (words) to be recognized. The input sequence of acoustic features is fed through the networks, producing sequences of scores for each class. A decoding engine based on Viterbi algorithm is run on the scores obtained this way.

The training is in two steps:

1. Each MLP is roughly trained by BP on pre-labeled sequences (e.g. the label for a given acoustic observation may represent the corresponding phoneme).

2. Fine-tuning of MLPs parameters is accomplished using an optimization scheme based on Generalized Probabilistic Descent (GPD) (Katagiri et al., 1991; Chang and Juang, 1993) and BP of partial derivatives throughout the layers of the net.

Another variant to Bourlard's approach can be found in (Reichl and Ruske, 1995). In that paper RBF networks, instead of MLPs, are used to estimate state-posterior probabilities. The training takes place again in two steps:

1. RBF models are trained by BP over a training set which was previously labeled by a standard HMM.

2. A global discriminative training of the hybrid, based on GPD with the Minimum Classification Error criterion (Juang and Katagiri, 1992), is accomplished.

Also in (Yan et al., 1997) ANNs are used to estimate state posteriors, but a novelty is proposed in the training scheme. Instead of considering only the strict values "0" and "1" as targets for output units, according to Viterbi segmentation of training sequences, the probability targets in the continuous range $(0, 1)$ generated within the forward-backward mechanism are considered. The training technique was evaluated on a recognition task of continuous digits collected over the telephone channel, where the novel hybrid scored a 4.9% WER, whereas the hybrid with standard training yielded a 6.0% WER, and a standard CDHMM scored a 5.0% WER.

Table 4.1 summarizes the main hybrid models of the present category, giving references, brief description and experimental results, in order to allow for a concise comparison.

4.1.3 The "global optimization" approach

The hybrid model described in (Bengio et al., 1992b; Bengio, 1993; Bengio, 1996) is based on the simultaneous estimation of all the parameters for both the HMM and the ANN according to the joint optimization of a single criterion, defined at the sequence level (e.g., word or sentence). The approach was originally inspired from the Alpha net (see Section 4.1.1).

Model	Brief description	Performance
Reference model (Bourlard and Morgan, 1993; Bourlard and Morgan, 1994a; Renals et al., 1994)	A MLP is trained with BP over a window of acoustic frames to estimate the posterior probabilities of CDHMM states given an acoustic observation sequence. A discriminative Maximum a Posteriori criterion is considered.	DARPA Resource Management (RM) "February91" test set, SI, continuous speech task, vocabulary size of 991: 5.8% WER (standard CDHMM: 11.0% WER).
Replacing MLP with RNN (Robinson, 1994; Hochberg et al., 1994; Hochberg et al., 1995)	Extension of the approach relying on RNNs, instead of static MLPs, to estimate posterior probabilities.	The system, called ABBOT, successfully participated to November 1993 ARPA Wall Street Journal Test. In 1994 a 8.8% WER on Nov93 *hub 2* test set, using a trigram LM, was reported.
Explicit context modeling (Franco et al., 1994)	Acoustic context is explicitly considered: the criterion is the likelihood of observations given the HMM state *and* the context. The latter is computed from the posterior probability of states given the observations and the context. MLPs (one for each context) are trained to estimate such probability, using a variant of BP with sharing of input-to-hidden weights among the nets.	DARPA RM, SI, continuous speech task, averaging over three Test Sets: 6.3% WER (context-independent hybrid: 8.8% WER; standard CDHMM: 7.0% WER).
Continuous target outputs (Yan et al., 1997)	Variant in the training scheme: instead of the strict targets "0" and "1", probability targets in the continuous range $(0,1)$ (generated within the Forward-Backward mechanism) are considered.	Recognition task of connected digits collected over the telephone channel: 4.9% WER (hybrid with Bourlard et al. training scheme: 6.0% WER; standard CDHMM: 5.0% WER).

Table 4.1: *Summary of main hybrid architectures based on a connectionist estimate of posterior probability of HMM states (Section 4.1.2). First column identifies the model, giving the bibliographic reference. Second column provides a brief description. Last column reports experimental results from literature.*

The ANN is trained as a *feature extractor* for a CDHMM, with the goal to transform input acoustic representations into compact, but significant, low-dimensional representations that are more suitable to be modeled by the emission probabilities of the HMM than standard acoustic parameters. In so doing, an increased recognition rate is expected.

The ANN maps a raw input sequence $\mathbf{u}_1^L$ onto a low-dimensional observation sequence $\mathbf{x}_1^L$. An assumption is made that the optimization criterion C used to train the HMM is a continuous and differentiable function of the observations $\mathbf{x}_t, t = 1, \ldots, L$, so that a gradient descent optimization method can be applied. The partial derivatives $\dfrac{\partial C}{\partial x_t}$ (here x_t denotes a generic component of t-th observation $\mathbf{x}_t$) can then be computed and used (along with the *chain rule*) to train the weights of the ANN. The training scheme is sketched in Figure 4.2.

Interestingly, the computation of $\dfrac{\partial C}{\partial x_t}$ is very simple, since the Forward-Backward algorithm implicitly computes the derivative of the HMM likelihood $Pr(\mathbf{x}_1^L|Y)$ with respect to the emission probabilities $b_{i,t} = Pr(\mathbf{x}_t|q_t = S_i)$ [1].

Beering in mind the definitions and the notation introduced in Section 1.2 and referring to (Rabiner, 1989) for details, it can be stated that in the forward pass the Baum-Welch algorithm computes:

$$\alpha_{i,t} = b_{i,t} \sum_j a_{ij} \alpha_{j,t-1}, \tag{4.6}$$

where $\alpha_{i,t} = Pr(q_t = S_i, \mathbf{x}_1^t)$. It can be shown (Rabiner, 1989) that the likelihood $Pr(\mathbf{x}_1^L)$ can be obtained from the α's at the last time step, for the set $\mathcal{F}$ of "final" states (Bengio, 1996) of the HMM, as follows:

$$Pr(\mathbf{x}_1^L) = \sum_{i \in \mathcal{F}} \alpha_{i,L}. \tag{4.7}$$

The backward pass is initialized as follows:

$$\beta_{i,L} = 1 \text{ if } i \in \mathcal{F}, \ 0 \text{ otherwise} \tag{4.8}$$

where $\beta_{i,t} = Pr(\mathbf{x}_{t+1}^L|q_t = S_i)$. These are computed recursively as:

$$\beta_{i,t} = \sum_j a_{ji} \beta_{j,t+1} b_{j,t+1}. \tag{4.9}$$

Note that from Equations (4.7) and (4.8) $\frac{\partial Pr(\mathbf{x}_1^L)}{\partial \alpha_{i,L}} = \beta_{i,L}$. From Equations (4.9) and (4.6) and proceeding by induction (applying also the chain rule), it is possible to write:

[1] For notational convenience, we write $b_{i,t}$ instead of $b_i(\mathbf{x}_t)$ to denote emission probability of state S_i at time t, keeping in mind that the observation sequence $\mathbf{x}_1^L$ is under consideration.

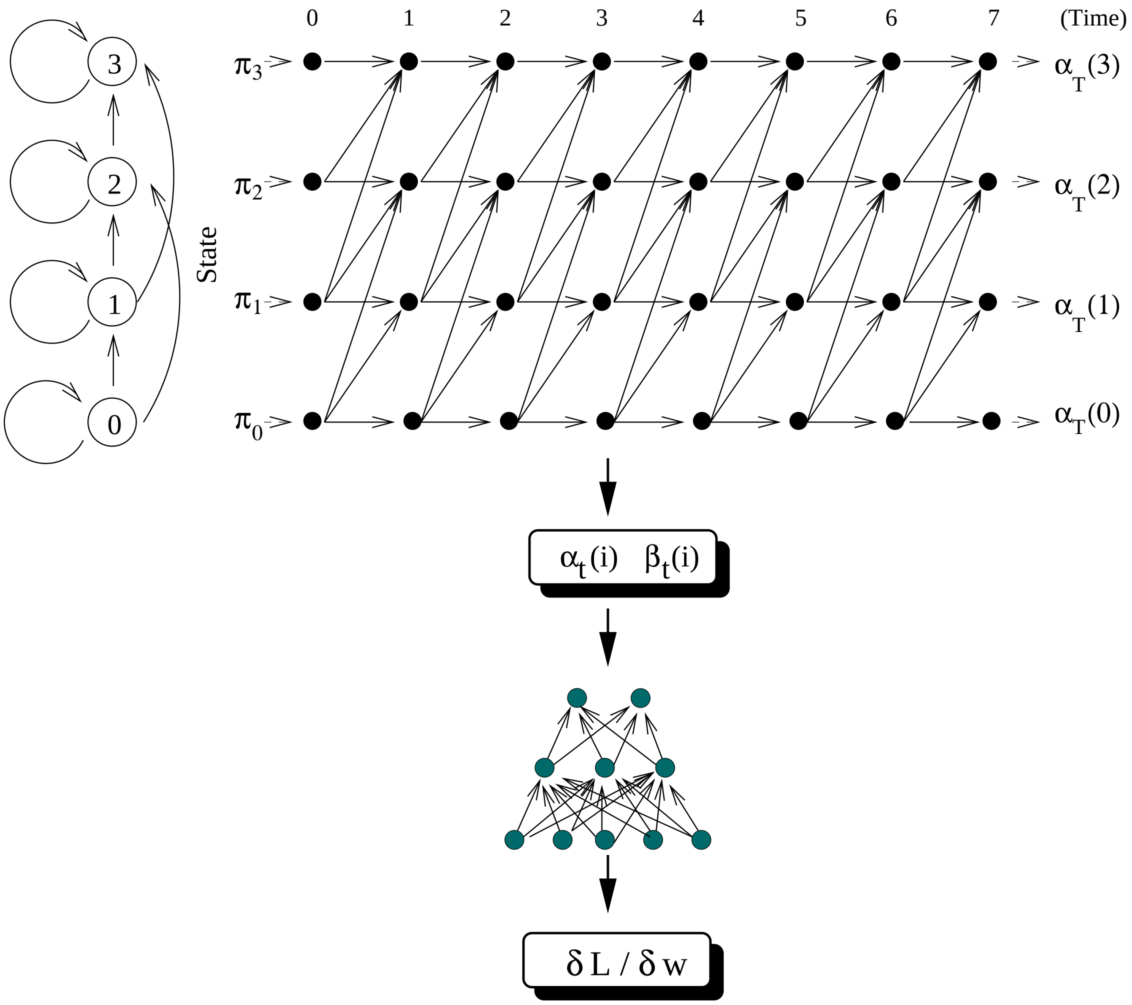

Figure 4.2: *Training scheme of Bengio's "global optimization" approach. A gradient-ascent learning rule for weight w of the ANN (which transforms raw input acoustic representations into low-dimensional, significant features) is obtained by taking partial derivatives of the likelihood L of the acoustic observations given the model. Quantities involved in the calculations, namely α's and β's at each time $t = 1, \ldots, T$ and for each HMM state i (a 4-state left-to-right HMM is shown), are* backpropagated *from the usual* Forward-Backward *algorithm applied to the standard* trellis *structure shown in the upper part of the figure.*

$$\frac{\partial Pr(\mathbf{x}_1^L)}{\partial \alpha_{i,t}} = \beta_{i,t}. \tag{4.10}$$

Derivatives with respect to the emission probabilities can now be obtained from Equation (4.6):

$$\frac{\partial Pr(\mathbf{x}_1^L)}{\partial b_{i,t}} = \frac{\alpha_{i,t}\beta_{i,t}}{b_{i,t}}. \tag{4.11}$$

When considering the case in which the training criterion C is the likelihood of the observations, and using a HMM constrained by the correct word sequence Y, the gradient of the criterion C with respect to the emission probabilities can be immediately obtained from Equation (4.11). To compute the gradient with respect to the ANN outputs and thus, via chain rule, with respect to the ANN weights, the computation of the partial derivatives of the emission density function with respect to the observations is needed. (Bengio et al., 1992b; Bengio, 1996) give an example in the case of Gaussians or Gaussian mixtures emission pdfs.

It should be noticed that in the above scheme an infinite value of the likelihood can be obtained whenever the ANN produces a constant output, since in this case the Gaussian mixtures can converge to zero variance, and the emission probabilities increase towards infinity, as well as the likelihood $Pr(\mathbf{x}_1^L)$. This troublesome behavior is the same that occurs when training standard Gaussian mixtures to maximize the likelihood when a Gaussian concentrates on one point. The problem did not occur in the experiments discussed in (Bengio et al., 1992b; Bengio, 1996). However, the problem can be definitely tackled by adopting discriminant training criteria, such as Maximum Mutual Information or Maximum A Posteriori (Bengio, 1996).

The basic model described above was further developed with the combination of multiple, specialized ANNs, trained on specific acoustic classes (e.g. plosives) with a class-dependent set of acoustic features. A *gathing* network that combined the outputs of specialized ANNs was initialized to perform principal component analysis. The whole HMM/multiple ANNs hybrid was then fine-trained with the global optimization technique.

Experimental results showed that training the ANN jointly with the HMM in the proposed hybrid architecture improved recognition performance over the standard CDHMM (Bengio et al., 1992b; Bengio, 1996), lowering the error rate on a plosive recognition task from 19% to 14%.

In (Johansen and Johnsen, 1994; Johansen, 1994) the global optimization technique was used to train an MLP as an extractor of "adjoint" input features for a HMM relying on the Maximal Mutual Information criterion.

Recently, (Riis and Krogh, 1997) proposed a more general framework for the global optimization approach, which extends the application of ANNs from feature extraction to estimation of HMM parameters, obtaining encouraging experimental results in recognition of five broad phonetic classes when compared to a discrete HMM and a discriminative conditional ML training criterion was adopted.

As a final consideration, it should be noticed that also the approaches where a joint optimization of the HMM and of the ANN is performed according to the GPD method (and that are described throughout the other Sections of this Survey) are, in general, instances of "globally optimized" architectures.

4.1.4 Networks as vector quantizers for discrete HMM

Starting from the late Eighties, a number of researchers began to apply ANNs to the problem of generating codebooks for discrete HMMs (see for instance (Kimber et al., 1990; Iwamida et al., 1991)). Most of those works relied on Kohonen's Learning Vector Quantization (LVQ) (Kohonen, 1986; McDermott and Katagiri, 1991) as an effective neural alternative to standard clustering algorithms.

A good example of the principles and motivations underlying the present approach can be found in (Rigoll, 1994). In that work a new neural architecture was proposed to perform Vector Quantization (VQ) on the acoustic features for a discrete HMM. The novelty of the approach concerned the training of the ANN. The latter was a feedforward net trained with an unsupervised algorithm based on the Maximal Mutual Information (MMI) criterion, having defined the Mutual Information (MI) as:

$$MI(Y, W) = H(Y) - H(Y \mid W) \tag{4.12}$$

where $H(\cdot)$ denotes entropy, Y is the sequence of labels (*codewords*) produced by the vector quantizer on the input feature stream, and W is the corresponding words string.

The proposed network topology is basically a 1-layer MLP with activations:

$$z_j = \| \mathbf{w}_j - \mathbf{x} \| = \sum_{i=1}^{d} (w_{ji} - x_i)^2 \tag{4.13}$$

which somehow expresses a "distance" between input vector $\mathbf{x} = (x_1, \ldots, x_d)$ and the j-th "codeword" (or prototype) of the codebook, represented by the connection weights for j-th output unit (see also Competitive Neural Networks (Hertz et al., 1991) used for clustering).

The training algorithm iteratively modifies the weights of the winner unit in order to increase the quantity (4.12). (Rigoll, 1994) extends also the procedure to the cases of multilayer nets and multiple input features.

Experimental results obtained by initializing the weights of the ANN with the *k-means* (or *isodata*) (Duda and Hart, 1973) clustering algorithm, in a SD task of 18 isolated Japanese phonemes from the ATR speech database, using a codebook size of 100, showed a 25.0% relative error reduction over the VQ based on k-means, averaging over 5 different speakers and using the same discrete HMM. Further improvements to the system were presented in (Rigoll et al., 1995; Neukirchen and Rigoll, 1997) and in (Rottland et al., 1997), where evolutions of ANN architecture and training were discussed, along with experimental results that compare well with those obtained with a CDHMM-based system.

(Jang and Un, 1996) presents a hybrid system for SI isolated words recognition. The system is constituted by a connectionist fuzzy-vector quantizer (FVQ), the output of which is fed into a discrete HMM. The proposed FVQ is a modular combination of TDNNs, where the nets are trained by BP on a training set labeled by hand. The TDNNs are used as phoneme classifiers. Two additional nets are trained to discriminate between vowels and consonants.

Instead of feeding the outputs of the networks into the HMM, the latter takes in input the internal representations (activations of the second hidden layer) of the acoustic features that the TDNNs develop. The internal representation is thought of as the (nonlinear) projection of the feature space over a transformed space, replacing the standard vector quantization process. The HMM is separately trained on the transformed space, using standard HMM-training algorithms.

In order to improve performance of the system and to allow training of the system with few data, a smoothing technique (Rabiner and Juang, 1986) is used. Emission probabilities are smoothed relying on a "smoothing matrix" computed from the average values of the activation vectors of the second hidden layer of the TDNNs.

A 44.9% relative WER reduction with respect to the discrete HMM with a floor-smoothing technique was gained in a SI (5 males for training, 2 males for test), isolated Korean words (75 words dictionary) recognition task, when 12 LPC MEL-cepstral coefficients were used.

Strictly related to the idea of using ANNs as vector quantizers for discrete HMMs is the concept of neural *labelers* (Ma and Compernolle, 1990; Cerf et al., 1994). In its basic version, this system is based on an MLP (or TDNN), fed with input acoustic features, with an output unit per each phonetic class. The actual output value can be 0 or 1, according to the membership of the input observation to the class or not. The output from the MLP is passed on to a standard discrete HMM that uses it as a class label within a Viterbi decoding paradigm. Training is supervised, relying on BP over a phonetically balanced training set that is pre-labeled by means of a standard discrete HMM.

The system turns out to be useful mainly for small vocabularies, where a model for each word is used (instead of phoneme models). The main advantage over approaches based on the estimation of state posterior probabilities is that no implicit requirement is made of reaching the global minimum of the error functional to keep consistency with the theoretical framework.

The basic model was then extended (Cerf et al., 1994) with the introduction of *N-Top*, that is to say, the N best (highest) MLP outputs are considered (not only the winner label) and passed on to the HMM. This is of particular interest in the regions of the feature space in proximity of separation surfaces between pairs of adjacent acoustic classes, where misclassification is more alike. Other improvements concerned the use of parallel MLPs trained on different acoustic features (e.g. bare coefficients, their first order derivatives, second order derivatives, energy) and the

Model	Brief description	Performance
VQ based on Maximal Mutual Information (MMI) (Rigoll, 1994; Rigoll et al., 1995; Neukirchen and Rigoll, 1997)	A feedforward net (1-layer or multi-layer Perceptron) is trained to perform VQ for a discrete HMM with a novel unsupervised algorithm based on the MMI criterion. Output activation functions somehow express a "distance" between acoustic input vector and codewords of the discretized codebook of input symbols.	ATR speech database, SD task of 18 isolated Japanese phonemes, codebook size of 100, averaging over 5 different speakers: 25.0% relative error reduction over the VQ based on k-means (using the same discrete HMM).
Connectionist fuzzy-vec-tor quantizer (FVQ) (Jang and Un, 1996)	A FVQ is used. It is a modular combination of TDNNs, where the nets are trained as phoneme classifiers by BP on a training set labeled by hand. TDNNs internal representations (activations of the second hidden layer) of the acoustic features are used as inputs for a discrete HMM, the latter being separately trained using standard algorithms.	SI (5 males for training, 2 males for test), isolated Korean words (75 words dictionary) recognition task: 44.9% relative WER reduction with respect to the standard discrete HMM.
Neural *labelers* (Ma and Compernolle, 1990; Cerf et al., 1994)	An MLP (or TDNN) with an output unit per each phonetic class is trained to yield output values equal to 0 or 1, according to the membership of the input observation to the class or not. The output from the MLP is passed on to a standard discrete HMM that uses it as a class label within a Viterbi decoding paradigm. Training is supervised, relying on BP over a pre-labeled training set.	SI, isolated Flemish digits task (over the telephone line); using MLP as labelers performed better than VQ based on standard clustering techniques, and yielded recognition rates comparable to those obtained with a hybrid ANN/HMM system with connectionist probability estimation.

Table 4.2: *Summary of main hybrid architectures based on a connectionist Vector Quantization of acoustic feature space for discrete HMMs (Section 4.1.4).*

introduction of a *Fuzzy Vector Quantization* scheme.

The system was evaluated on a SI, isolated Flemish digits task over the telephone line. Results reported in (Cerf et al., 1994) are difficult to interpret, since they miss a direct comparison with a standard CDHMM; nonetheless, using MLP as labelers performed better than vector quantization based on standard clustering techniques, and yielded recognition rates comparable to those obtained with a hybrid ANN/HMM system with connectionist probability estimation.

Main hybrid architectures based on a connectionist Vector Quantization of acoustic feature space for discrete HMMs are summarized in Table 4.2.

4.1.5 Other approaches

One important topic is related to the concept of connectionist *rescoring*. The idea, proposed in (Zavaliagkos et al., 1993), is developed in (Zavaliagkos et al., 1994), where a connectionist approach is applied to the problem of rescoring the hypothesis generated by a HMM which uses an N-best strategy. In this case the network does not compute scores on individual acoustic frames, but on whole *segments* (subsequences) of frames, corresponding to phonemes. In this way, correlations between frames that are close in time - and that correspond to the same phoneme - are exploited, thus overcoming the usual limitations following the assumption of independence made in standard HMMs.

The connectionist model used for this purpose is called Segmental Neural Network (SNN) (Austin et al., 1992). It takes a given acoustic segment in input, and produces an estimation of the posterior probability of a certain frame given that segment. The overall score, which is actually used in the rescoring process of a whole sentence is obtained by multiplying the scores of the segments in which the sentence has been segmented by the HMM. The score obtained this way is then linearly combined with that computed by the HMM, yielding a total score for the sentence given a specific segmentation. The weights for the linear combination are empirically determined by cross-validation on a development set.

Since the input layer of the networks has fixed size, while the segments are made up of a variable number of frames, two alternative techniques of "normalization" of the length of the input window are proposed. The former is a semilinear temporal sampling, which drops some of the original frames if the segment is longer than the input dimensionality, or replicates some of them if the segment is shorter. The alternative is a discrete cosine transform (DCT) applied to the whole segment, retaining as many parameters as it is necessary to fill the input layer.

Training of the system is accomplished using a segmentation performed by the HMM, possibly considering the first N-best hypothesis of segmentation - where segmentations from the 2nd best down to the Nth best are used as negative examples. Networks are trained on a relative entropy criterion; the outputs are normalized in order to fit the probabilistic framework.

Different neural architectures were tried (1-layer Perceptron, MLP, HyperBF)

(Zavaliagkos et al., 1994) without significant fluctuations in performance. They were also combined altogether to obtain a more robust rescoring process.

The system was evaluated on the DARPA RM, SI, continuous speech task, with a $1,000$ words vocabulary and a bigrams grammar (perplexity 60), using also duration modeling, one left context (at the segment level) and a regularization technique to improve performance. The rescoring approach obtained combining HMM, SNN based on MLP and HyperBF, improved the WER of the bare HMM from 3.4% to 2.7% on the test set RM Feb91, and from 6.0% to 5.5% averaging on the two test sets RM Sep92.

Another application of ANNs for rescoring of a standard HMM with N-best strategy can be found in (Moudenc et al., 1996).

Connectionist *wordspotting* was discussed, for instance, in (Morgan et al., 1991; Li and Naylor, 1992; Lippmann and Singer, 1993).

Somewhat similar to connectionist rescoring of HMM hypothesis is the technique by (Mari et al., 1994). In this case, a standard HMM with N-best is used for connected speech recognition with confusable words (e.g. acoustically similar words). The neural nets are applied after the HMM has generated the N-best word-sequence hypothesis, to "correct" (or confirm) those individual words that belong to specific "confusable" classes. For each such a class, an MLP is trained to produce output scores for all the words of the given class. The MLP is fed with an input vector of acoustic features (MEL Frequency Scaled Cepstral Coefficients) extracted in an *ad hoc* manner, according to a prior knowledge concerning which position within an acoustic segment is more likely to allow for better discrimination among words of a certain class. Training of the MLPs is accomplished using BP. The proposed technique allowed for a significant gain in performance for some classes of confusable letters (e.g. "B", "D", and "V"), from 491 spellings of names collected over the telephone channel.

The approach proposed by (Dugast et al., 1994) is not truly a hybrid, since it uses a standard HMM (trained with Viterbi algorithm on the ML criterion), which is used in parallel with connectionist models by combining the estimates of the emission probabilities (likelihoods) provided by the HMM with the normalized scores obtained with the ANNs. The linear combination scheme is the following:

$$\log P = \mu \log P_{HMM} + (1 - \mu) Q_{net} \tag{4.14}$$

where $\log P$ is used as the combined emission log-likelihood estimate for a given state of the HMM, P_{HMM} is the usual emission likelihood yielded by the standard HMM, Q_{net} is the properly normalized output of the ANNs (a score assigned to the same state of the HMM) and μ is a tuned weighting linear coefficient in the range $(0.0, 1.0)$.

The combined value obtained from Equation (4.14) is used within a Viterbi decoding strategy.

The ANNs are a hierarchical mixture of TDNNs, with a gather (root of the one-level tree which defines the hierarchy) which is trained, along with the experts of the mixture (the leaves of the hierarchy), to assign each acoustic frame to the proper TDNN. Each TDNN expert is specialized on subsets of phonemes, in order to reduce the complexity of training on whole corpora of continuous-speech data.

The networks are trained with the usual MSE criterion, instead of ML, and the approach is expected to improve the baseline Markovian recognizer thanks to the combination of the different training criteria.

In fact, the hierarchical TDNNs mixture, trained and tested on the DARPA RM SD corpus significantly worsened the performance of the HMM (test was carried out by averaging on three different reference speakers), while the results obtained with the TDNN/HMM combination (1.7% average WER) were slightly better then those provided by the HMM alone (2.0% average WER), that is a 15% relative WER reduction at the expense of only 10% increase in the number of free parameters to be estimated during training.

In (Gurgen et al., 1994) the classical framework of a HMM-based recognizer where ANNs are used as preprocessors (feature extractors, VQs, etc.), or probability estimators, is inverted. The HMM is used as a preprocessor for an ANN, the latter being the classifier. The HMM maps sequences of acoustic features into subsequences (the normalized means of the Gaussian emission probabilities of the states associated to the observation sequence according to Viterbi alignment) of fixed length (the number of states within the corresponding left-to-right model). The subsequence obtained this way is then passed in input to the ANN which, in turn, carries out the classification.

In (Hutter, 1995) MLPs are trained by BP to estimate the class-dependent probability density functions associated to the states of a SCHMM.

A different approach is discussed in (Chung and Un, 1996b). In the latter, ANNs are used for *weighting* the emission probabilities (likelihoods) associated to the states of a CDHMM for continuous speech recognition. The system relies on a CDHMM with mixtures of Gaussian components. The HMM is trained and used for decoding of speech signals using conventional algorithms (e.g. Viterbi), but an MLP (with an input window of 3 left contexts and 3 right contexts, i.e. a basic TDNN) is associated to each state of the HMM in order to estimate a state "weight" (or credit), given the current acoustic observations. Performance improvement is expected from the combination of HMM capabilities to classify long sequences, with discriminative training properties of ANNs.

In practice, (Chung and Un, 1996b) used a pair of MLPs for each phoneme, one for the acoustic features (LPC Cepstral coefficients) and the other for the corresponding time derivatives; each net was "split" over the three adjacent states of 3-state, left-to-right HMMs that modeled phonemes.

The training algorithm is a three-step procedure:

1. Training of the CDHMM, using a conventional algorithm.

2. Training of the MLPs as phoneme classifiers, using BP.

3. Re-training of the MLPs with an iterative scheme in which the models obtained at steps 1 and 2 above are used (the re-trained MLPs are then used during the successive iterations) to classify the training data, evaluating a differentiable, discriminative criterion function aimed at minimizing the error rate (GPD). The partial derivatives of the criterion with respect to the weights of the MLPs are then backpropagated within the networks to obtain new weight values, and the next iteration is started.

The system was experimentally evaluated over a continuous speech, SI (16 speakers for training, 10 for testing) task, with vocabulary of 102 Korean words. The best results were obtained using mixtures of 4 Gaussian components: WER on the test set was reduced from 15.9% (HMM) to 15.1% (HMM with non-retrained MLPs), and finally down to a significant 11.1% (HMM with retrained MLPs).

In (Chung and Un, 1996a) another approach is presented, which extends the idea of Neural Prediction Model (NPM) (Iso and Watanabe, 1990) by integrating non-linear neural predictors within a hybrid framework. Each predictor is a 2-layer MLP, with a hidden layer of sigmoids and a linear output layer, trained by BP. The MLP is fed with acoustic features, and produces in output the prediction for the acoustic frame at time t, given a time window of the acoustic observations which immediately precede and follow the t-th frame. Such predictors are associated to the states of a CDHMM, where mixtures of Gaussian components are used to model the distributions of the prediction errors.

The major advantages of this system are the capability of the networks to take into account the temporal correlations between adjacent acoustic observations and the global optimization of the system. The latter relies on a GPD-based algorithm that simultaneously trains the HMM and the ANNs (the latter ones acting as feature extractors for the HMM).

Two training techniques are proposed, based on ML and on a discriminative criterion (minimum classification error), respectively. Experiments accomplished on a database consisting of 102 Korean words, in a continuous speech, SI (16 speakers for training and 27 speakers for testing) task, using 12 LPC plus energy and their first order time-derivatives, showed a 22.8% WER for the hybrid architecture trained with ML criterion, and a 11.1% WER when the discriminative training criterion was used, while the standard HMM resulted in a 16.0% WER.

In (Pican et al., 1996; Pican et al., 1997) a novel neural model, called OWE (Orthogonal Weight Estimator), is used in order to face the problem of context dependence. The OWE is applied after a step of standard context-dependent HMM (or, alternatively, to feed a second-order HMM with transformed input frames) in recognition tasks from the TIMIT database.

Recently, (Moon and Hwang, 1997) have proposed a novel approach based on the joint optimization of models and feature space in order to improve robustness to noise in a standard HMM. The approach is based on the principle of *inversion* of a neural network (Linden and Kindermann, 1989), although no strict integration between HMM and ANN is adopted.

The inversion principle refers to the duality which holds between the weights of an ANN and the values of the inputs in the computation of the gradient of the criterion (error) function. Indeed, analogously to the calculation of the delta rule (Pao, 1989) for weight updating in the BP algorithm (Rumelhart et al., 1986a), it is possible to obtain an updating rule for the current d-dimensional input $\mathbf{x} = (x_1, \ldots, x_d)$ of an ANN by writing:

$$x'_i = x_i - \eta \frac{\partial C}{\partial x_i} \tag{4.15}$$

for $i = 1, \ldots, d$, where $\mathbf{x}' = (x'_1, \ldots, x'_d)$ is the updated feature vector, i.e. a transformed input which is made more suitable to the ANN (the parameters of the latter are kept fixed) in order to extremize the training criterion, and C is the criterion itself (e.g. MSE). The effective calculation of the partial derivatives in Equation (4.15) is accomplished by repeatedly applying the chain rule and backpropagating them throughout the layers of the network, like in the standard BP case. (Moon and Hwang, 1997) apply the same approach for inversion of a HMM, writing:

$$x'_{it} = x_{it} + \eta \frac{\partial C_{HMM}}{\partial x_{it}} \tag{4.16}$$

where x_{it} is i-th coefficient of t-th acoustic frame in the input sequence $\mathbf{x}$, x'_{it} is the same coefficient after the transformation has been applied, and C_{HMM} is the criterion to be optimized to increase performance of the HMM. For instance, if a ML training criterion is used, then $C_{HMM} = \log Pr(\mathbf{x} \mid \lambda) = \log \alpha_{N,L}$ (log-likelihood), where λ is the vector of parameters of the HMM and $\alpha_{N,L}$ is the usual *forward probability* term (Rabiner, 1989) used in the forward-backward algorithm, for the final state S_N, assuming an input sequence of length L. It should be noticed that in the ML case a maximization is to be carried out, instead of a minimization of a cost (error) function as in ANNs, and this motivates the presence of the sign "+" in Equation (4.16).

In addition to the application of the gradient method, (Moon and Hwang, 1997) proposes also an inversion technique directly based on the Baum-Welch algorithm, as well as a combined scheme where the HMM inversion is combined with a MINIMAX-based (Merhav and Lee, 1993) model parameter adaptation (Lee et al., 1991). Parameter adaptation can be accomplished in a batch manner, as well as alternating it with an inversion step, in an iterative fashion. Experimental results on isolated digits with noise of the TI database with 16 speakers, using CDHMMs with 7 states (a model for each digit) and mixtures of 4 Gaussian components defined over the 12-dimensional LPC space, showed that the combined scheme yields an improvement over the standard HMM, increasing the recognition rate from 25.73% to 54.90% with Signal-to-Noise Ratio (SNR) of 5 dB, and from 76.63% to 89.27% with SNR of 20 dB.

Model	Brief description	Performance
Connectionist *rescoring* (Zavaliagkos et al., 1993; Zavaliagkos et al., 1994)	A *Segmental Neural Network* computes scores on whole *segments* of frames, corresponding to phonemes, for re-scoring the segmentation hypothesis yielded by a HMM which uses an N-best strategy.	DARPA RM, SI, continuous speech task (vocabulary size of $1,000$): 2.7% WER on Feb91 test set (standard CDHMM; 3.4% WER); 5.5% WER averaging on Sep92 test sets (standard CDHMM; 6.0% WER).
Rescoring of "confusable words" (Mari et al., 1994)	MLPs are applied after the HMM has generated the N-best word-sequence hypothesis, to "correct" (or confirm) those individual words that belong to specific "confusable" classes.	A Significant gain in performance for some classes of confusable letters (e.g. "B", "D", and "V"), from 491 spellings of names collected over the telephone channel.
Parallel ANN/HMM state-probability estimates (Dugast et al., 1994)	HMM estimates of the emission probabilities are linearly combined with scores obtained with a hierarchical mixture of TDNNs.	DARPA RM SD task, averaging on three different reference speakers: 1.7% WER (standard HMM alone: 2.0% WER).
Connectionist *weighting* of state-emission probabilities (Chung and Un, 1996b)	ANNs are associated with HMM states to estimate a state "weight" (or credit), given the current acoustic observations. Training combines BP and GPD.	Continuous speech, SI (16 speakers for training, 10 for testing) task, vocabulary of 102 Korean words: 11.1% WER (standard CDHMM: 15.9% WER).
Hybrid HMM/Neural Prediction Model (NPM)(Chung and Un, 1996a)	MLPs are fed with a time window of acoustic observations centered in the t-th frame and predict the acoustic frame at time t. The pdfs associated with the states of a CDHMM are used to model the distributions of the prediction errors.	Continuous speech, SI (16 speakers for training and 27 speakers for testing) task, 102 Korean words: 11.1% WER (standard HMM: 16.0% WER).
HMM *inversion* (Moon and Hwang, 1997)	Joint optimization of models and feature space via "model inversion": an updating rule for the current input vector is developed, and the transformed input is expected to be more suitable to the model.	TI database, isolated digits task with noise, 16 speakers: increase in recognition rate from 76.63% (standard CDHMM) to 89.27% (applying inversion) with SNR of 20 dB.

Table 4.3: *Summary of main "other approaches" discussed in Section 4.1.5.*

(Menendez-Pidal et al., 1994) and (Menendez-Pidal et al., 1995) use Time-Space Neural Network (TSNN), a variant of TDNN, for probability estimation of phonemes given the acoustic observations. Different techniques for the integration within a HMM are presented, featuring the adoption of fuzzy-probabilistic information. in a continuous speech, SD task.

Finally, other approaches are those by (Gemello et al., 1997), where a different kind of acoustic units (named Stationary-Transitional units) is successfully used within a hybrid with connectionist estimates of the emission probabilities, by (Dupont et al., 1997), and by (Bengio and Frasconi, 1996; Bengio and Bengio, 1996); the latter introduces a novel paradigm, namely the *Input-Output HMM*, which extends the concept of HMM by allowing for generation of sequences of *output values* in front of sequences in input.

Table 4.3 summarizes the main approaches discussed throughout the present Section.

4.2 Future Issues: toward a novel ANN/HMM architecture

Hybrid speech recognizers allowed for significant gain in performance with respect to standard HMMs in a variety of situations. In addition to the scientific interest related to their theoretical framework, hybrid systems appear to be a flexible and efficient alternative paradigm, to be taken into full consideration by the speech community in the next years to face open ASR problems.

This chapter surveyed different combination techniques that have been proposed in literature, highlighting a number of opportunities to take advantage from both HMMs and ANNs. One future issue concerns merging some of these techniques together, e.g. globally-optimized connectionist feature extraction (as in Bengio's approach) for an hybrid system where ANNs perform probability estimation (as in Bourlard and Morgan's paradigm). ANNs properties can be exploited by combining neural models at different levels within the ASR system, increasing robustness of the latter, its acoustic modeling capabilities and supporting integration with the LM. Increase in robustness to acoustic variability is expected as a consequence of the generalization ability of ANNs, as well as from selective (adaptive) extraction of relevant acoustic features. Accuracy in acoustic modeling is expected due to the universal approximation property of ANNS (Cybenko, 1989), potentially suitable to model any form of the emission pdfs associated with HMM states. A different perspective (Bourlard and Morgan, 1993) is to consider ANNs as optimal non-parametric estimators of Bayesian classifiers. Finally, integration with the LM is sought whenever connectionist approaches to language modeling (Morgan and Scofield, 1991), in addition to acoustic modeling, are used: under these circumstances, a scenario where a global BP learning scheme is defined within a homogeneous (neural) combined system can be hypothesized. Some issues that are strictly related to the open problems introduced so far follow.

An important topic in state-of-the-art ASR systems is *model adaptation* (Zavaliagkos et al., 1995; Neto et al., 1995), e.g. *speaker normalization* (Huang, 1992; Trentin and Giuliani, 1997) or *channel adaptation* (Das et al., 1994; Takahashi and Sagayama, 1994). Speaker normalization deals with the problem of recognizing speech signal acquired from a new speaker, not included in the training population used to estimate parameters for the ASR system, whose voice may significantly differ from those used for training. Differences in vocal tracts dramatically affect the recognition performance. Similarly, in channel adaptation the diversity in acoustic conditions is introduced by a novel acquisition channel, e.g. the telephone line. Related problems, which have to deal with system robustness, concern the electronic transducer, namely the microphone(s) used to capture the voice signal: the microphone used for testing may differ from that used to collect the training data, or it may no longer be close-mouth but placed at a certain distance from the talker(s). A *microphone array* (Flanagan et al., 1991; Che et al., 1994) is sometimes needed to allow hands-free recognition of people moving within a room, and so on. Model adaptation techniques usually aim at building an acoustic front-end to a pre-trained HMM-based recognizer, or to tune the parameters of the latter, in order to increase recognition performance whenever acoustic conditions are changed with respect to those which held during the training, using a limited amount of acoustic material collected in the new conditions and avoiding a full re-training of the whole system. The nature of ANNs, e.g. their adaptivity and generalization ability, should be exploited to yield effective approaches to the problem of model adaptation. In particular, *on-line* learning (Bengio, 1996) given a global training criterion defined at the recognizer level (e.g., ML or minimum WER) appears promising in this respect.

Another hot issue concerns ASR in noisy environments, such as a room with many people talking or loud background noises (e.g. noisy electrical devices), a car or the telephone line. Many noise-reduction approaches, based also on ANNs, have been investigated by the speech community, such as extraction of robust acoustic features or blind separation of sources (Bell and Sejnowski, 1995). Future hybrid systems should allow for adaptive and robust connectionist reduction, or filtering, of noise. As mentioned above, a globally optimized robust feature extraction process is sought.

A final remark to VLSI implementation of ASR systems is due: one major limitation of state-of-the-art ASR systems based on standard HMMs is the difficulty to implement them in hardware (at least when non-trivial tasks are considered). ANNs may require much less parameters than HMMs and are easily implemented in VLSI. This could be a considerable advantage of many hybrid systems, since availability of compact and efficient hardware devices would increase applicability and diffusion of ASR to a large extent.

In summary, hybrid ANN/HMM systems with connectionist estimation of probabilistic quantities for the underlying HMM (i.e., Bourlard-and-Morgan-like) appear definitely promising, since the role of the ANN should in principle allow to tackle the limitations of standard HMMs that were pointed out in Section 1.2. Unfortunately, such models rely on a kind of heuristic training scheme (the iterative Viterbi/BP procedure is not theoretically motivated). In addition, generalization properties of the resulting learning machine are not taken into explicit account, and no experimental evidence of the robustness of ANN/HMM paradigms in noisy environments is available yet. On the contrary, the "global optimization" framework by Bengio (after Bridle) is mathematically motivated and sound, but the particular role of ANNs within the overall architecture (i.e., feature extractors) do not allow for solving the above problems, still being the underlying recognizer a bare CDHMM. Our goal is combining these distinct perspectives, developing a novel, globally optimized hybrid with connectionist probability estimation for a HMM (Chapter 5). In addition, since a novel training algorithm was proposed in Chapter 3 capable to improve generalization in feedforward and recurrent ANNs, further efforts will be concentrated on combining such an algorithm with the new hybrid in order to increase its robustness to noise (Chapter 6).

Chapter 5

Training Algorithms for a Novel ANN/HMM Hybrid

This chapter introduces a novel algorithm based on a gradient-ascent technique for global training of a hybrid ANN/HMM system, in which the ANN is used as a non-parametric estimator of emission probabilities associated with individual states of the HMM. The approach is related to the major hybrid systems proposed by Bourlard & Morgan and by Bengio, respectively, with the aim to combine the benefits from both within a unifying framework and to overcome their limitations.

5.1 Toward a Novel Paradigm for ASR

In Chapter 4 we reviewed several hybrid architectures. Although, basically, all of them do represent significant paradigms to some extent, not all of the directions which they point out may be considered definitely promising at the state of the art. Indeed, relevance of the *Viterbi net* and of the *Alpha net* (Section 4.1.1) is only historical. Connectionist vector quantization modules for discrete HMMs (Section 4.1.4) are not allowed - due to their particular role - to fix the gain that hold, in general, between the discrete underlying models and the more powerful CDHMMs. Other approaches (Section 4.1.5), such as connectionist rescoring, neural labeling, or the HMM-inversion, eventually rely on a standard CDHMM, including the usual limitations of the latter (as summarized in Section 1.2). It is our conviction that - as explained at the end of Section 4.2 - two major approaches emerge, under different perspectives, because of their potential: Bourlard and Morgan's connectionist posterior probability estimations (*et similia*, according to Table 4.1), and Bengio's globally optimized neural feature extractor. The former features an effective and outstanding architecture, but lacks of a theoretically-motivated training algorithm. The latter, on the contrary, relies on a steady mathematical framework for the global optimization of its parameters, but the role of the ANN, joint with a completely standard CDHMM, is inherently limited. This chapter proposes novel ANN/HMM architectures and training techniques, aimed at combining the advantages of both these schemes, tackling their intrinsic drawbacks.

The present hybrid relies on ANN non-parametric estimation of the emission probabilities of a HMM, with a gradient-ascent global training technique aimed at maximizing the likelihood of acoustic observations given the model. The ANN has an output unit for each of the HMM states, which provides an estimate of the corresponding emission probability given the current acoustic observation in the feature space.

Analytical developments are first outlined in Section 5.2, where a ML on-line learning rule for connection weights of the ANN is obtained in the linear domain. Implementation issues, e.g. batch training and learning rules in the logarithmic domain are discussed in Section 5.3. Once the generic training algorithm is given, a critical point has to be considered (Section 5.4): how can we ensure that the ANN outputs actually estimate likelihoods? Major solutions to the problem are attempted by: (i) imposing a probabilistic constraint on ANN weights; (ii) factorizing the emission probabilities via Bayes' Theorem, i.e. performing connectionist *state-posterior* probability estimation; (iii) maximizing a *Maximum A Posteriori* training criterion, instead of the bare ML. An experimental demonstration is carried out in Section 5.5, where a reduced-scale task (speaker dependent, isolated digits recognition) is used to analyze the behavior of the algorithms and to compare them with a standard HMM, and finally a real-world, speaker independent continuous speech recognition task is performed.

5.2 Derivation of a Gradient-based Algorithm to train the ANN/HMM Hybrid

Let us consider the global criterion function C to be extremized by the model during training, namely the *likelihood* L of the acoustic observations given the model:

$$C = L \tag{5.1}$$

where we can write

$$L = \sum_{i \in \mathcal{F}} \alpha_{i,T} \tag{5.2}$$

where the sum is extended to the set $\mathcal{F}$ of all possible *final* states within the HMM (see Chapter 4) corresponding to the current phonetic transcription, which is supposed to have Q states, and T is the length of the current observation sequence $Y = \mathbf{y}_1, \ldots, \mathbf{y}_T$. *Forward* and *backward* terms for state i at time t are defined as:

$$\alpha_{i,t} = b_{i,t}(\mathbf{y}_t) \sum_j a_{ji} \alpha_{j,t-1} \tag{5.3}$$

and

$$\beta_{i,t} = \sum_j b_{j,t+1}(\mathbf{y}_{t+1}) a_{ij} \beta_{j,t+1} \tag{5.4}$$

where a_{ij} denotes the transition probability between i-th and j-th states, $b_{i,t}(\mathbf{y}_t)$ denotes emission probability associated with i-th state over t-th acoustic observation $\mathbf{y}_t$, and the sums are extended to all possible states within the HMM. From now on, for simplicity, we will write $b_{i,t}$ instead of $b_{i,t}(\mathbf{y}_t)$. The initialization of the forward probabilities is accomplished as usual in standard HMMs, while the backward terms at time T are initialized as follows:

$$\beta_{i,T} = \begin{cases} 1 & \text{if } i \in \mathcal{F} \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

For a generic weight w of the ANN, hill climbing gradient-ascent prescribes a learning rule of the kind:

$$\Delta w = \eta \frac{\partial C}{\partial w} \tag{5.6}$$

that is to say, given Equation (5.1):

$$\Delta w = \eta \frac{\partial L}{\partial w}. \tag{5.7}$$

where η is the *learning rate*. *Gradient ascent* is required instead of the usual gradient-descent approach, since the aim herein is to *maximize* the likelihood, not to *minimize* any kind of error measure.

In the following we carry out analytical developments for $\frac{\partial L}{\partial w}$, in order to obtain a learning rule for the ANN aimed at maximizing the likelihood of the acoustic observations, which is the usual criterion adopted to train standard HMMs. Eventually, the learning rule can be applied in parallel with the Baum-Welch algorithm, limiting the latter to the ML estimation of those quantities (namely the initial and transition probabilities in the underlying HMM) that do not depend on the ANN weights.

Before starting the calculations, let us observe that the following simple property can be easily shown to hold true by straightforwardly taking the partial derivatives of the left- and right-hand sides of Equation (5.3) with respect to $b_{i,t}$:

$$\frac{\partial \alpha_{i,t}}{\partial b_{i,t}} = \frac{\alpha_{i,t}}{b_{i,t}}. \tag{5.8}$$

In addition, borrowing the scheme proposed by Bridle (Bridle, 1990) and Bengio, the following theorem can be proved, by induction, to hold true:
Theorem 5.1: $\frac{\partial L}{\partial \alpha_{i,t}} = \beta_{i,t}$, for each $i = 1, \ldots, Q$ and for each $t = 1, \ldots, T$.
Proof See Appendix A.2. $\mathcal{Q.E.D}$

Given Equation (5.8) and Theorem 5.1, repeatedly applying the chain rule we can expand $\frac{\partial L}{\partial w}$ by writing:

$$\begin{aligned}\frac{\partial L}{\partial w} &= \sum_i \sum_t \frac{\partial L}{\partial b_{i,t}} \frac{\partial b_{i,t}}{\partial w} \\ &= \sum_i \sum_t \frac{\partial L}{\partial \alpha_{i,t}} \frac{\partial \alpha_{i,t}}{\partial b_{i,t}} \frac{\partial b_{i,t}}{\partial w} \\ &= \sum_i \sum_t \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \frac{\partial b_{i,t}}{\partial w}.\end{aligned} \tag{5.9}$$

where the sums are extended over all states $i = 1, \ldots, Q$ of the HMM corresponding to the correct transcription of the training utterance under consideration, and to all $t = 1, \ldots, T$, respectively. From now on, attention is focused on the calculation of $\frac{\partial b_{i,t}}{\partial w}$, where $b_{i,t}$ is the output from i-th unit of the ANN at time t. Bengio adopted a scheme where the emission probabilities were modeled by continuous pdfs (e.g Gaussian or mixtures of Gaussians) and carried out calculations within that parametric assumption, where the output of the ANN yielded a novel feature space where such pdfs were defined. In the present case, the ANN acts as a non-parametric estimator of the emission probabilities, and the calculation of partial derivatives is intended as a learning scheme capable to yield a suitable non-parametric estimation model, instead of feasible features for fixed-form parametric estimates.

The calculations are definitely similar to those required in the well-known *Backpropagation* (BP) algorithm, but they differ slightly. Backpropagation of *deltas* from the output to the first hidden layer is not extended over all output units (thus requiring to distinguish only between weights in the topmost layer and weights in the hidden layers); in this case, the quantity $b_{i,t}$ is affected only by a subset of the weights in the topmost layer, thus requiring different calculations for backpropagation of deltas in the first hidden layer (which differ from standard BP) and in the (possible) following hidden layers (which are closer to standard BP).

Let us consider a multilayer feed-forward network (e.g. a MLP), the j-th output of which, computed over t-th input observation $\mathbf{y}_t$, is expected to be a non-parametric estimate of the emission probability $b_{j,t}$ associated with j-th state of the HMM at time t. An activation function $f_j(x_j(t))$, either linear or non-linear, is attached to each unit j of the ANN, where $x_j(t)$ denotes input to the unit itself at time t. The corresponding output $o_j(t)$ is given by $o_j(t) = f_j(x_j(t))$. The net is assumed to have n layers $\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_n$, where $\mathcal{L}_0$ is the input layer and is not counted, and $\mathcal{L}_n$ is the output layer. For notational convenience we write $i \in \mathcal{L}_k$ to denote the index of i-th unit in layer $\mathcal{L}_k$.

Consider first the case of a generic weight w_{jk} between k-th unit in layer $\mathcal{L}_{n-1}$ and j-th unit in the output layer. It is possible to write:

$$\frac{\partial b_{j,t}}{\partial w_{jk}} = \frac{\partial f_j(x_j(t))}{\partial w_{jk}} \tag{5.10}$$

$$
\begin{aligned}
&= f_j'(x_j(t)) \frac{\partial \sum_{i \in \mathcal{L}_{n-1}} w_{ji} o_i(t)}{\partial w_{jk}} \\
&= f_j'(x_j(t)) o_k(t).
\end{aligned}
$$

By defining the quantity

$$
\delta_i(j,t) = \begin{cases} f_j'(x_j(t)) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{5.11}
$$

for each $i \in \mathcal{L}_n$ (where dependency on the specific HMM state j under consideration and on current time t has been explicitly written, since this will turn out to be useful in the final formulation of the learning rule), we can rewrite Equation (5.10) in the compact form:

$$
\frac{\partial b_{j,t}}{\partial w_{jk}} = \delta_j(j,t) o_k(t). \tag{5.12}
$$

Consider now the case of connection weights in the first hidden layer. Let w_{kl} be a generic weight between l-th unit in layer $\mathcal{L}_{n-2}$ and k-th unit in layer $\mathcal{L}_{n-1}$.

$$
\begin{aligned}
\frac{\partial b_{j,t}}{\partial w_{kl}} &= \frac{\partial f_j(x_j(t))}{\partial w_{kl}} \qquad (5.13) \\
&= f_j'(x_j(t)) \sum_{i \in \mathcal{L}_{n-1}} \frac{\partial w_{ji} o_i(t)}{\partial w_{kl}} \\
&= f_j'(x_j(t)) \frac{\partial w_{jk} o_k(t)}{\partial o_k(t)} \frac{\partial o_k(t)}{\partial w_{kl}} \\
&= f_j'(x_j(t)) w_{jk} \frac{\partial f_k(x_k(t))}{\partial w_{kl}} \\
&= f_j'(x_j(t)) w_{jk} f_k'(x_k(t)) \frac{\partial \sum_{i \in \mathcal{L}_{n-2}} w_{ki} o_i(t)}{\partial w_{kl}} \\
&= f_j'(x_j(t)) w_{jk} f_k'(x_k(t)) o_l(t).
\end{aligned}
$$

Similarly to definition (5.11), we introduce the quantity:

$$
\delta_k(j,t) = f_k'(x_k(t)) \sum_{i \in \mathcal{L}_n} w_{ik} \delta_i(j,t) \tag{5.14}
$$

for each $k \in \mathcal{L}_{n-1}$, which allows us to rewrite Equation (5.13) in the form:

$$
\frac{\partial b_{j,t}}{\partial w_{kl}} = \delta_k(j,t) o_l(t) \tag{5.15}
$$

which is formally identical to Equation (5.12).

We finally need analytical developments for the partial derivatives of the ANN output with respect to a generic connection weight in the following hidden layers.

We carry out the calculations for a generic, hypothetic weight w_{lm} holding between m-th unit in layer $\mathcal{L}_{n-3}$ and l-th unit in layer $\mathcal{L}_{n-2}$, but the same results will apply also to subsequent layers $(\mathcal{L}_{n-4}, \mathcal{L}_{n-5}, \ldots, \mathcal{L}_0)$.

$$\begin{aligned}
\frac{\partial b_{j,t}}{\partial w_{lm}} &= \frac{\partial f_j(x_j(t))}{\partial w_{lm}} \\
&= f_j'(x_j(t)) \sum_{i \in \mathcal{L}_{n-1}} \frac{\partial w_{ji} o_i(t)}{\partial w_{lm}} \\
&= f_j'(x_j(t)) \sum_{i \in \mathcal{L}_{n-1}} \frac{\partial w_{ji} o_i(t)}{\partial o_i(t)} \frac{\partial o_i(t)}{\partial w_{lm}} \\
&= f_j'(x_j(t)) \sum_{i \in \mathcal{L}_{n-1}} w_{ji} \frac{\partial o_i(t)}{\partial w_{lm}}.
\end{aligned} \tag{5.16}$$

Efforts are now concentrated on the development of the term $\frac{\partial o_i(t)}{\partial w_{lm}}$. This is accomplished by writing:

$$\begin{aligned}
\frac{\partial o_i(t)}{\partial w_{lm}} &= \frac{\partial f_i(x_i(t))}{\partial w_{lm}} \\
&= f_i'(x_i(t)) \sum_{k \in \mathcal{L}_{n-2}} \frac{\partial w_{ik} o_k(t)}{\partial w_{lm}} \\
&= f_i'(x_i(t)) \frac{\partial w_{il} o_l(t)}{\partial o_l(t)} \frac{\partial o_l(t)}{\partial w_{lm}} \\
&= f_i'(x_i(t)) w_{il} \frac{\partial f_l(x_l(t))}{\partial w_{lm}} \\
&= f_i'(x_i(t)) w_{il} f_l'(x_l(t)) \frac{\partial x_l(t)}{\partial w_{lm}} \\
&= f_i'(x_i(t)) w_{il} f_l'(x_l(t)) \sum_{k \in \mathcal{L}_{n-3}} \frac{\partial w_{lk} o_k(t)}{\partial w_{lm}} \\
&= f_i'(x_i(t)) w_{il} f_l'(x_l(t)) o_m(t)
\end{aligned} \tag{5.17}$$

which can be substituted into Equation (5.16) obtaining:

$$\frac{\partial b_{j,t}}{\partial w_{lm}} = f_j'(x_j(t)) \sum_{i \in \mathcal{L}_{n-1}} w_{ji} f_i'(x_i(t)) w_{il} f_l'(x_l(t)) o_m(t). \tag{5.18}$$

Again, as in (5.11) and (5.14), for each $l \in \mathcal{L}_{n-2}$ we define the quantity[1]:

$$\delta_l(j,t) = f_l'(x_l(t)) \sum_{i \in \mathcal{L}_{n-1}} w_{il} \delta_i(j,t) \tag{5.19}$$

[1] For lower layers the sum in Equation (5.19) must be accomplished over the units belonging to the immediately upper layer.

and rewrite Equation (5.18) in the familiar, compact form:

$$\frac{\partial b_{j,t}}{\partial w_{lm}} = \delta_l(j,t) o_m(t) \tag{5.20}$$

that is formally identical to Equations (5.12) and (5.15).

Using the above developments to expand Equation (5.9) and substituting it into Equation (5.7), the latter can now be restated in the form of a general *learning rule* for a generic weight w_{jk} of the network, by writing:

$$\Delta w_{jk} = \eta \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \delta_j(i,t) o_k(t) \tag{5.21}$$

where the term $\delta_j(i,t)$ is computed via Equation (5.11), Equation (5.14) or Equation (5.19), according to the fact that the layer under consideration is the output layer, the first hidden layer, or one of the other hidden layers, respectively. In the following, we will refer to learning rule (5.21) as the *Bare Maximum Likelihood* (BML) training algorithm for the ANN/HMM hybrid.

Note that the backpropagation scheme for *deltas* in the hidden layers (from $\mathcal{L}_{n-2}$ down to $\mathcal{L}_0$) is identical to that of standard BP, but since the deltas at the output and first hidden layer differ from those of standard BP, the same quantities yielded by BP itself cannot be straightforwardly used.

Once trained, the ANN can be used along with the Viterbi algorithm to perform decoding of novel sequences of acoustic observations.

5.3 Implementation Issues

Direct implementation of the learning rule (5.21) may turn out to be difficult or even unfeasible due to computational complexity problems. In addition, the algorithm developed so far deals with a single training sequence Y, while actual training sets are composed of hundreds or even thousands of training utterances collected among a number of speakers. Finally, the analytical developments of Section 5.2 are concentrated on the likelihood L and rely on quantities which are defined in the linear domain, while numerical stability problems in HMMs suggest to perform calculations in the logarithmic domain, aiming at the maximization of the *log-likelihood.* The present Section discusses in detail each one of these topics, providing comments, discussion and further analytic developments needed to overcome the implementation problems.

5.3.1 Rewriting the basic equation

It is convenient to rewrite Equation (5.21) in the following way:

$$\Delta w_{jk} = \eta \sum_{t=1}^{T} o_k(t) \sum_{i=1}^{Q} \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \delta_j(i,t). \tag{5.22}$$

This reduces the number of calculations by a factor Q, since multiplying by $o_k(t)$ for Q times within the argument of the second sum is no longer needed.

5.3.2 On-line versus batch training

Section 5.2 introduced an algorithm to train an ANN to learn a maximum-likelihood non-parametric estimation of an individual acoustic observation sequence Y. More generally, we are interested in training the ASR system over a training set $\mathcal{Y}$ containing a number of training sequences, i.e. $\mathcal{Y} = \{Y_1, Y_2, \dots, Y_n\}$. The same requirement holds in standard HMMs, where slight modifications of the training algorithms are needed (Rabiner, 1989). As a matter of fact, a new training criterion $\mathcal{C}$ has to be considered, namely the overall likelihood $\mathcal{L}$ of the training sequences $\mathcal{Y}$ given the model $\mathcal{M}$:

$$\mathcal{C} = P(\mathcal{Y} \mid \mathcal{M}) = \prod_{i=1}^{n} P(Y_i \mid \mathcal{M}). \tag{5.23}$$

Being the learning rule (5.22) aimed at the maximization of the criterion C, defined by Equation (5.1), its sequential application to $Y_1, Y_2, \dots, Y_n$ may possibly fail in the maximization of $\mathcal{C}$. The problem can be qualitatively stated by saying that the "local" maximization of the likelihood of a given acoustic sequence Y_i does not imply maximization of the global likelihood to any extent.

Before developing a mathematically correct learning rule focused on $\mathcal{C}$, let us consider the fact that in standard ANNs two instances of the training algorithm are possibly applied: the *on-line* (or *stochastic*) version (Bengio, 1996), and the so-called *batch* version. The former prescribes application of the gradient-descent method for weight update for each individual pattern of the training set. The latter, analytically correct, requires calculation and summation of the gradients over all the patterns in the training set, before updating the weights. In practice, the on-line version is known to perform better than the batch in a variety of situations. The BP algorithm is usually applied in its stochastic version. The rationale behind this behavior is the benefit introduced by local "fluctuations" in the weight space, induced by the on-line version: fluctuations allow for escaping local minima of the criterion function.

Similar arguments may be expected to hold in the present setup. In the following, we will refer to Equation (5.22) as the *on-line* learning rule for the ANN/HMM hybrid in the linear domain[2]. Although its application after presentation of each training sequence $Y_1, Y_2, \dots, Y_n$ does not imply maximization of $\mathcal{C}$, its behavior will

[2]Equations in the logarithmic domain are introduced in the next Sections.

be evaluated throughout the experiments. As a matter of fact, the following paragraphs highlight that the complexity in space required by rule (5.22) is much lighter than in the batch case.

Let us now concentrate on the mathematical developments required to obtain a learning rule aimed at the maximization of $\mathcal{C}$. The amount of change prescribed by gradient ascent for a given weight w in the ANN is:

$$\Delta w = \eta \frac{\partial \mathcal{C}}{\partial w} = \eta \frac{\partial}{\partial w} \prod_{i=1}^{n} P(Y_i \mid \mathcal{M}) \tag{5.24}$$

where Equation (5.23) has been taken into account. Taking the partial derivatives of the product with respect to w we can rewrite Equation (5.24) in the following form:

$$\Delta w = \eta \sum_{i=1}^{n} \left\{ \frac{\partial P(Y_i \mid \mathcal{M})}{\partial w} \prod_{j \neq i} P(Y_j \mid \mathcal{M}) \right\} \tag{5.25}$$

that implies the calculation of $\frac{\partial P(Y_i \mid \mathcal{M})}{\partial w}$ as in the on-line case, and prescribes to update the weights of the ANN according to a linear combination of individual contributes yielded by learning rule (5.22) computed over each sequence of the training set. In the following, we will refer to Equation (5.25) as the *batch* learning rule in the linear domain.

This scheme requires to store all the values of Δw for each weight in the ANN and for each training sequence Y_i, in addition to the likelihoods $L_i = P(Y_i \mid \mathcal{M})$ for $i = 1, \ldots, n$. For certain cases, e.g. whenever n is hundreds or thousands and the ANN has hundreds of neurons, the number of real values in double precision to be stored becomes excessive.

If $P(Y_i \mid \mathcal{M}) > 0$ for each observation sequence Y_i, e.g. if the emission pdfs were Gaussian or mixtures of Gaussian components, one way to overcome the problem is to rewrite Equation (5.25) as:

$$\begin{aligned}
\Delta w &= \eta \sum_{i=1}^{n} \left\{ \frac{\partial P(Y_i \mid \mathcal{M})}{\partial w} \frac{\prod_{j=1}^{n} P(Y_j \mid \mathcal{M})}{P(Y_i \mid \mathcal{M})} \right\} \\
&= \eta \left\{ \prod_{j=1}^{n} P(Y_j \mid \mathcal{M}) \right\} \sum_{i=1}^{n} \left\{ \frac{\partial P(Y_i \mid \mathcal{M})}{\partial w} \frac{1}{P(Y_i \mid \mathcal{M})} \right\}
\end{aligned} \tag{5.26}$$

where the condition $P(Y_i \mid \mathcal{M}) > 0$ may be imposed by constraining the outputs of the ANN to be greater than zero. From an implementation viewpoint, applying Equation (5.26) as a batch learning rule in the linear domain requires to store only: (i) the value of $\prod_{j=1}^{n} P(Y_j \mid \mathcal{M})$, updated multiplying by $P(Y_i \mid \mathcal{M})$ at the end of the forward pass over each training sequence $Y_1, \ldots, Y_n$; and (ii) an *accumulator* for each weight of the ANN. Accumulators are set to zero before starting the training, and are updated after presentation of each new training sequence Y_i, by summing the quantity $\frac{\partial P(Y_i|\mathcal{M})}{\partial w}\frac{1}{P(Y_i|\mathcal{M})}$, computed *via* the on-line learning rule. As a matter of fact, given the arbitrariness in the choice of the learning rate η, the multiplicative factor $\prod_{j=1}^{n} P(Y_j \mid \mathcal{M})$ in Equation (5.26) could even be absorbed by η, defining a new learning rate $\eta' = \eta \prod_{j=1}^{n} P(Y_j \mid \mathcal{M})$. This simplification further reduces the storage requirements, since keeping track of the product of the $P(Y_j \mid \mathcal{M})$ is no longer needed.

In practice, in order to avoid forcing the ANN outputs to be non-null, and to tackle numerical stability problems that may arise due to the division by $P(Y_i \mid \mathcal{M})$ in the right-end side of Equation (5.26), a heuristic may be introduced and the following algorithm may be applied:

1. *Initialization:* set accumulators to zero.

2. *Loop over all training sequences:* for $i = 1, \ldots, n$

   (a) *Forward pass:* compute $P(Y_i \mid \mathcal{M})$

   (b) *Backward pass:* compute $\frac{\partial P(Y_i|\mathcal{M})}{\partial w}$ for each weight w, applying the on-line learning rule.

   (c) *Checkpoint:* if $P(Y_i \mid \mathcal{M}) < \epsilon$ then update the ANN weights according to the quantities computed in the *Backward pass*, i.e., apply the *on-line* learning rule; exit loop and jump to *Initialization*.

   (d) *Accumulators updating:* update the accumulators according to $\frac{\partial P(Y_i|\mathcal{M})}{\partial w}$ and $P(Y_i \mid \mathcal{M})$ (computed as in the on-line case).

3. *Learning:* update all the connection weights w in the ANN, according to the values of the accumulators, applying the learning rule (5.26).

4. *Iteration:* if other training epochs are required, jump to *Initialization*.

The heuristic is represented by the introduction of the *Checkpoint*. The latter compares the value of $P(Y_i \mid \mathcal{M})$ with that of a given (small) positive scalar ϵ. If the likelihood of Y_i turns out to be too low, numerical problems would occur in applying the rule (5.26): in that case, the heuristic prescribes to apply the on-line learning rule over sequence Y_i, in order to increase its likelihood by properly changing the ANN weights, and to restart the training from the first observation sequence using the new weights. From a qualitative point of view, this is equivalent to say that if the likelihood L_i of a sequence is below a given threshold, it calls for an immediate weight update aimed at increasing L_i, before continuing the global training. It is worth emphasizing that L_i is seldom expected to be that small, and in general the above algorithm is just an explicit formulation of the training scheme resulting from the adoption of the batch learning rule in the linear domain.

5.3.3 On-line learning in the logarithmic domain

Numerical stability considerations in standard HMMs suggest to carry out calculations in the logarithmic domain, since in so doing the computation of forward ($\alpha_{i,t}$) and backward ($\beta_{i,t}$) probabilities over long paths in the trellis is reduced to sums of logarithms, instead of products of infinitesimal quantities. This Section develops an on-line learning rule in the logarithmic domain, assuming that log-α's and log-β's are used along the trellis, and the ANN outputs are interpreted as estimates of the logarithm of the emission probabilities associated with states of the HMM.

Let us consider again the criterion (5.1), i.e. $C = L$ for a given observation sequence Y. Using the same notation introduced in Section 5.2 we can rewrite Equation (5.9) as:

$$\begin{aligned}\frac{\partial L}{\partial w} &= \sum_i \sum_t \frac{\partial L}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial w} \qquad (5.27)\\ &= \sum_i \sum_t \frac{\partial L}{\partial \alpha_{i,t}} \frac{\partial \alpha_{i,t}}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial w}\\ &= \sum_i \sum_t \beta_{i,t} \frac{\partial \alpha_{i,t}}{\partial \log \alpha_{i,t}} \frac{\partial \log \alpha_{i,t}}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial w}\\ &= \sum_i \sum_t \beta_{i,t} \frac{\partial e^{\log \alpha_{i,t}}}{\partial \log \alpha_{i,t}} \frac{\partial \log \alpha_{i,t}}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial w}\\ &= \sum_i \sum_t \beta_{i,t} \alpha_{i,t} \frac{\partial \log \alpha_{i,t}}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial w}.\end{aligned}$$

Given definition (5.3), the logarithm of $\alpha_{i,t}$ can be immediately written as:

$$\log \alpha_{i,t} = \log b_{i,t}(\mathbf{y}_t) + \log(\sum_j a_{ji} \alpha_{j,t-1}) \qquad (5.28)$$

and its partial derivative with respect to $\log b_{i,t}$ is:

$$\frac{\partial \log \alpha_{i,t}}{\partial \log b_{i,t}} = 1 \qquad (5.29)$$

that, along with Equation (5.27), allows us to write:

$$\frac{\partial L}{\partial w} = \sum_i \sum_t \beta_{i,t} \alpha_{i,t} \frac{\partial \log b_{i,t}}{\partial w}. \qquad (5.30)$$

Since the ANN outputs $f_i(x_i(t))$ are expected to yield the logarithms of the emission probabilities, the quantity $\frac{\partial \log b_{i,t}}{\partial w}$ in Equation (5.30) can be replaced by $\frac{\partial f_i(x_i(t))}{\partial w}$. The same calculations accomplished in Section 5.2, in the linear case, to compute the latter quantity can still be applied (Equations 5.10 and following), without any changes. The same definitions of $\delta_j(i,t)$ given therein can be used. From these considerations and from Equation (5.30) the *on-line learning rule in the logarithmic domain* turns out to be the following:

$$\Delta w_{jk} = \eta \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \alpha_{i,t} \delta_j(i,t) o_k(t) \tag{5.31}$$

where w_{jk} is a generic connection weight between k-th and j-th units in adjacent layers of the ANN, the outputs of the latter are assumed to be logarithms of probabilities, the α's and β's are in the linear domain (explicit conversion *via* the exponential function is thus required), and the other quantities are defined in the same manner as in Section 5.2.

Note that the rule (5.31) is similar to the learning rule (5.21) for the linear case, but it is more compact and offers more numerical stability, since no division by the value of the emission probabilities is required.

5.3.4 Batch learning in the logarithmic domain

Extending the learning rule in the logarithmic domain to the batch case, i.e. whenever n training observation sequences $Y_1, \ldots, Y_n$ are considered, follows the steps outlined in Section 5.3.2. In particular, the criterion (5.23) is used, and Equation (5.26) still holds. The difference is that in the present case the on-line learning rule for the logarithmic domain, namely Equation (5.30), is used to compute $\frac{\partial L_i}{\partial w}$, where L_i is the likelihood of i-th observation sequence Y_i, as usual. The same considerations concerning implementation and numerical stability topics given in Section 5.3.2 apply also to this case.

5.3.5 Nature of ANN outputs

Although sigmoidal activation functions are commonly used in ANNs neurons, the output range of sigmoids may not be suitable for the present estimation task, since values of emission likelihoods range in the $[0, +\infty)$ interval (linear case), or in the $(-\infty, +\infty)$ interval (logarithmic domain). Using linear activations in the output layer may be feasible. A promising alternative in the logarithmic case is to use logarithms as output activation functions in the ANN. The mathematics developed so far is independent from the analytic nature of the activation functions. In practice, especially if linear activations are used (which are intrinsically unbound), the considerations provided in the previous sections suggest to force the outputs within meaningful ranges.

5.4 The "divergence problem"

In Section 5.2 we developed a gradient-ascent learning rule suitable for ANN training with the goal of maximizing the likelihood of the acoustic observations given the model. The learning rule developed therein is expected to work given the fact that the ANN can be actually treated as a non-parametric estimate of the emission probabilities of the underlying HMM. Unfortunately, in order to satisfy the latter condition, the ANN itself should be a (general and non-parametric) model of a pdf, i.e. the integral of the function computed by the ANN over its whole domain (the acoustic space) should equal 1. While ANNs, being "universal approximators", *may* reasonably approximate any pdf as close as desired, they completely lack of any constraints ensuring that, at any given time during training, they actually *are* pdfs. In practice, a direct application of the above training scheme may not lead to a correct solution, since the training criterion (ML) simply encourages the ANN to yield high output values, i.e. high emission probabilities, over all input patterns. In so doing, the overall likelihood of the training input sequences diverges toward infinity, since the ANNs do not estimate pdfs, but they simply develop larger values for their connection weights. The resulting model turns out to be meaningless (and useless). We refer to such a phenomenon as the "divergence problem". A strictly related scenario was pointed out by Bengio (Bengio, 1996), where the ANNs for feature extraction might simply learn to concentrate all the inputs closely around the mean vectors of the Gaussian emission probabilities, the latter quickly becoming Dirac's deltas. While using Gaussian mixtures provided Bengio with the satisfaction of the "integral-equal-to-one" condition, in the present setup the problem is even harder.

Bengio states that, in practice, the problem is not crucial, since the ANN usually reaches "more interesting" (Bengio, 1996) extremes of the criterion function. Similar reasoning could lead us to the expectation that a direct application of the algorithm described in Section 5.2 could provide us with interesting solutions, maybe at the expense of a few training trials with different connectionist architectures and different starting points in the weight space. Although some experiments reported in Section 5.5 do confirm this expectation to some extent, the problem is definitely relevant, and further theoretical developments are sought in order to obtain a more reliable and mathematically motivated training scheme.

The present Section is devoted to the development of alternative solutions to the problem, *via* modification (or completion) of the proposed algorithm. Since explicit introduction of the probabilistic constraint, i.e. "integral equals 1", in the training criterion does not lead to any feasible analytical formulation, the following alternatives are considered herein:

1. introduction of architectural constraints on the ANN topology in order to ensure a suitable probability density estimation through a direct application of the above learning rules;

2. application of a probabilistic constraint on the network weights. The criterion is no longer ML, but a joint probability that combines the likelihood of the observations *and* the probability of the ANN weights given the model. This is similar to some well-known *regularization* techniques (Bishop, 1995), where

"simpler" solutions, i.e. with small weight values, are privileged over complex solutions, e.g. with huge weights;

3. factorization of the emission probability using Bayes' theorem. Some terms of the factorization can be estimated separately from the training set, before ANN training. The ANN is then used to estimate the posterior probability of a HMM state given the acoustic input. This means that the emission probability is no longer expected to be the output of the ANN, but the latter is combined with other terms to obtain the emission probability itself. Being the ANN output interpreted as a probability, the constraints to be satisfied are: (i) that each output unit is forced in the range $[0, 1]$, and (ii) that the sum over all output units be exactly 1. Such constraints can be explicitly introduced in the training scheme;

4. adoption of a "discriminant" training criterion, namely *Maximum A Posteriori*, instead of the bare Maximum Likelihood, in order to force probabilities along "incorrect" paths of the HMM to be discouraged.

The different alternatives are presented in Sections 5.4.1 through 5.4.4 below. We stress the fact that, in practice, the ANN is not necessarily required to realize a pdf, but *divergence* of its output values should rather be avoided. Some of the solutions that are outlined in the following are thus aimed at the definition of architectures/training algorithms that prevent the ANN to develop unbounded outputs while increasing the likelihood of the acoustic observations.

5.4.1 Imposing architectural constraints on the ANN topology

In Section 2.2 we introduced a feed-forward connectionist model called RBF, and we pointed out that, under certain circumstances, it can be interpreted as a mixture of Gaussian components. In particular, if the values of the connection weights that link the radial basis activation functions of the hidden layer with the linear output units sum to one, the model actually satisfies the requirements of a pdf. For this reason, RBFs may look promising in the present framework. In fact, there are two major drawbacks that severely limit the applicability of RBFs herein: (i) since they realize bare mixture of Gaussians, no significant gain in modeling capabilities may be expected over the standard CDHMM; (ii) whenever trained via the gradient method, hidden-to-output weights may diverge in order to increase the overall likelihood (as discussed above) since no explicit constraint ensures that, for each output unit, they sum to one. At least, a normalization to one should be forced after each training iteration. It is evident that such a heuristic, while preserving the pdf requirements, would have an impact on the training scheme. In spite of these problems, the RBF remains interesting - especially in a validation perspective - since it allows for an immediate numerical comparison with the standard HMM. For instance, the RBF may be initialized with the same parameters of a pre-trained HMM, and further trained using gradient-ascent. Section 5.5.1 ("Round 1") will take advantage of a similar initialization procedure over a slightly modified ANN architecture, namely a kind of log-RBF where the logarithms of Gaussian activation functions are used in the hidden layer.

An alternative is represented by a MLP having Gaussian output units. Presence of the latter ones could ensure the satisfaction of the pdf constraints, tackling the "divergence problem". Although such an architecture may appear unpromising, due to the fact that it basically replaces mixtures of Gaussians (as in standard CDHMMs) with individual Gaussians (one for each output unit, that is one for each state of the underlying HMM), it should be noticed that the MLP performs a non-linear transformation of the input features that is aimed at maximizing the likelihood. This means that the output Gaussians are defined over a new, transformed space which is expected to better suit the ML framework for the recognition problem at hand. This architecture, along with the BML algorithm, provides us with a scheme that strictly resembles Bengio's paradigm, i.e. the ANN extracts "optimal" features to be modeled with a Gaussian-based HMM. The "divergence problem" may thus rise again in the same form as in Bengio's case (as outlined in the previous Section): indeed, one possible way for the ANN to maximize the likelihood is simply to project all the input patterns onto a common internal representation - developed by the hidden layers - and then to map such a representation to a common centroid, shared among all output units, after driving all of the Gaussians toward the same mean vector.

5.4.2 Imposing a probabilistic constraint on the ANN weights

The present approach is inspired from several *regularization* techniques (Bishop, 1995), in particular *weight decay* (Plaut et al., 1986), *soft weight sharing* (Nowlan and Hinton, 1992), and the so-called *Bayesian framework* by MacKay (MacKay, 1992a; MacKay, 1992b). The basic idea is to avoid divergence of weight values by encouraging simpler solutions, i.e. with small weights, with respect to complex solutions, i.e. with large weights. The ANN training scheme must be modified in order to extremize the criterion while keeping the weights "as small as possible".

In the present setup, the idea can be formalized imposing a probabilistic constraint on the ANN weights, by replacing the criterion function (5.1) (the bare likelihood of the acoustic observations given the model) with a new criterion C', namely a *joint* probability having the form

$$C' = P(Y \mid \mathcal{M})P(W \mid \mathcal{M}) \tag{5.32}$$

where $L = P(Y \mid \mathcal{M})$ is the usual likelihood of the observation sequence Y given the model $\mathcal{M}$ under consideration, and $W = (w_1, \ldots, w_s)$ is the random vector of all connection weights in the ANN. Weights are assumed to be random variables distributed according to certain pdfs. The goal is now the maximization of criterion C'.

Exactly like in Equation (5.6), we can use gradient ascent to maximize C' by taking:

$$\Delta w = \eta \frac{\partial C'}{\partial w} \tag{5.33}$$

for a generic weight w. From Equation (5.32) it can be obtained

$$\begin{aligned} \frac{\partial C'}{\partial w} &= \frac{\partial P(Y \mid \mathcal{M}) P(W \mid \mathcal{M})}{\partial w} \\ &= P(W \mid \mathcal{M}) \frac{\partial P(Y \mid \mathcal{M})}{\partial w} + P(Y \mid \mathcal{M}) \frac{\partial P(W \mid \mathcal{M})}{\partial w}. \end{aligned} \tag{5.34}$$

The choice of the form for $P(W \mid \mathcal{M})$ is crucial. For instance, if a uniform distribution is chosen, then no constraint on the weights are imposed, Equation (5.34) reduces to Equation (5.9), and the criteria C and C' coincide[3]. In order to encourage small weights, a distribution with a peak around zero, and penalizing large arguments, is sought. A Normal distribution centered in zero appears promising in this respect: the zero-mean Gaussian is the pdf underlying the classical regularization techniques for ANNs training. Under this hypothesis, assuming that the weights are independent random variables, we can write:

$$P(W \mid \mathcal{M}) = \prod_{j=1}^{s} P(w_j \mid \mathcal{M}) = \prod_{j=1}^{s} N(w_j; 0, \sigma_j^2) \tag{5.35}$$

where $N(w_j; 0, \sigma_j^2)$ denotes the j-th (univariate) Gaussian, having zero mean and variance σ_j^2, evaluated over current value of weight w_j. Partial derivatives with respect to a generic weight w can be explicitly calculated:

$$\frac{\partial P(W \mid \mathcal{M})}{\partial w} = \frac{\prod_{j=1}^{s} P(w_j \mid \mathcal{M})}{P(w \mid \mathcal{M})} \frac{\partial P(w \mid \mathcal{M})}{\partial w}. \tag{5.36}$$

and substituted into Equation (5.34) to obtain a proper learning rule. It is possible to introduce the quantity[4] $\lambda = \frac{1}{\sigma^2}$, such that $P(w \mid \mathcal{M}) = \sqrt{\frac{\lambda}{2\pi}} \exp(-\frac{\lambda}{2} w^2)$, and to write:

$$\frac{\partial P(w \mid \mathcal{M})}{\partial w} = -\sqrt{\frac{\lambda}{2\pi}} \lambda w \exp(-\frac{\lambda}{2} w^2) = -\lambda w P(w \mid \mathcal{M}) \tag{5.37}$$

[3] Except for the multiplicative constant $P(W \mid \mathcal{M})$, which is irrelevant in the criterion extremization perspective.

[4] This notation is introduced to emphasize the relationship with regularization techniques, in particular the trainable amplitude algorithm described in Chapter 3, as discussed in the following.

that can be substituted into Equation (5.36) and the latter, in turn, into Equations (5.34) and (5.33) to obtain the following *on-line learning rule*:

$$\Delta w_{jk} = \eta P(W \mid \mathcal{M}) \left\{ \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \delta_j(i,t) o_k(t) - P(Y \mid \mathcal{M}) \lambda_{jk} w_{jk} \right\} \tag{5.38}$$

once the term $\frac{\partial P(Y|\mathcal{M})}{\partial w}$ has been calculated using Equations (5.9) through (5.20) of Section 5.2, which are not affected by the present setup. The quantities $P(W \mid \mathcal{M})$ and $P(Y \mid \mathcal{M})$ (the likelihood L) can be computed, at each iteration of the algorithm, in a straightforward manner from Equations (5.35) and (5.2), respectively. The term λ_{jk} is the inverse of the variance of the Gaussian pdf associated with weight w_{jk}, and the other symbols have the same meaning as in Equation (5.21). We will refer to learning rule (5.38) as the *Soft Weight Sharing ML* (SWS-ML) training algorithm for the ANN/HMM hybrid.

In passing, note that the multiplicative terms η and $P(W \mid \mathcal{M})$ may be conveniently absorbed within a unique *variable learning rate* η', defined as $\eta' = \eta P(W \mid \mathcal{M})$, that is implicitly *adapted* at each iteration of the learning process.

The above calculations can be simplified by means of a result presented in (Nowlan and Hinton, 1992). The latter states that assuming zero-mean Gaussian distributions having variance $1/\lambda_j$ for weight w_j, $j = 1, \ldots, s$, is equivalent to introducing in the criterion function an *additive* term in the form $\frac{1}{2}\sum_j \lambda_j w_j^2$. Extremizing the latter can be described as maximizing the *log* probability density of the weights[5]. In (Nowlan and Hinton, 1992) the criterion is the minimization of the empirical error: this motivates the summation operation. On the contrary, a *subtraction* is required herein, since the present criterion (the likelihood) has to be maximized. The new criterion is then:

$$C'' = L - \frac{1}{2} \sum_{j=1}^{s} \lambda_j w_j^2 \tag{5.39}$$

which emphasizes the relationship with the regularization technique relying on trainable amplitudes λ_j described in Chapter 3. As in Equations (5.6) and (5.33), gradient ascent prescribes to apply change $\Delta w = \eta \frac{\partial C''}{\partial w}$ to weight w and, in turn, given Equation (5.39):

$$\Delta w = \eta \left(\frac{\partial L}{\partial w} - \lambda w \right). \tag{5.40}$$

The term $\frac{\partial L}{\partial w}$ can be calculated as above. Exactly like in Equation (5.21), Equation (5.40) can now be rewritten in the form of the following learning rule:

[5] The present approach thus implies moving from the probability to the log-probability domain, i.e. simplification in calculations implicitly moves away from the theoretically motivated criterion (5.32), unless the log-likelihood of the acoustic observation sequences is considered as an overall criterion instead of the bare likelihood.

$$\Delta w_{jk} = \eta \left\{ \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \delta_j(i,t) o_k(t) - \lambda_{jk} w_{jk} \right\} \tag{5.41}$$

where the symbols have the usual meaning.

Concerning now the *implementation issues* (Section 5.3), the arguments used in Section 5.3.1 can still be applied, and the on-line learning rule (5.38) becomes:

$$\Delta w_{jk} = \eta P(W \mid \mathcal{M}) \left\{ \sum_{t=1}^{T} o_k(t) \sum_{i=1}^{Q} \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \delta_j(i,t) - P(Y \mid \mathcal{M}) \lambda_{jk} w_{jk} \right\}. \tag{5.42}$$

In the *batch* case (Section 5.3.2) the probabilistic criterion becomes:

$$\tilde{C}' = P(\mathcal{Y} \mid \mathcal{M}) P(W \mid \mathcal{M}) = \left\{ \prod_{i=1}^{n} P(Y_i \mid \mathcal{M}) \right\} P(W \mid \mathcal{M}) \tag{5.43}$$

and its partial derivative with respect to a generic weight w becomes:

$$\frac{\partial \tilde{C}'}{\partial w} = P(W \mid \mathcal{M}) \frac{\partial \prod_{i=1}^{n} P(Y_i \mid \mathcal{M})}{\partial w} + \prod_{i=1}^{n} P(Y_i \mid \mathcal{M}) \frac{\partial P(W \mid \mathcal{M})}{\partial w} \tag{5.44}$$

where $\frac{\partial \prod_{i=1}^{n} P(Y_i|\mathcal{M})}{\partial w}$ can be computed as in Equations (5.24)-(5.26), and the calculations outlined above to compute $\frac{\partial P(W|\mathcal{M})}{\partial w}$ and to obtain Equation (5.38) can still be applied in order to yield the following *batch learning rule*:

$$\Delta w_{jk} = \eta P(W \mid \mathcal{M}) \prod_{j=1}^{n} P(Y_j \mid \mathcal{M}) \left\{ \sum_{i=1}^{n} \frac{\partial P(Y_i \mid \mathcal{M})}{\partial w} \frac{1}{P(Y_i \mid \mathcal{M})} - \lambda_{jk} w_{jk} \right\}. \tag{5.45}$$

Let us now turn attention to the logarithmic case. Consider again criterion (5.32) and compute Δw according to Equations (5.33) and (5.34). As in Section 5.3.3, we can use Equations (5.27) through (5.30) to develop $\frac{\partial P(Y|\mathcal{M})}{\partial w}$. As in Equation (5.31) we obtain the following *on-line learning rule in the logarithmic domain*:

$$\Delta w_{jk} = \eta P(W \mid \mathcal{M}) \left\{ \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \alpha_{i,t} \delta_j(i,t) o_k(t) - P(Y \mid \mathcal{M}) \lambda_{jk} w_{jk} \right\} \tag{5.46}$$

In the *batch* case, the development of the logarithmic learning rule starts by considering the criterion (5.43), and Equation (5.45) still holds. The difference is that, in the present case as well as in Section 5.3.4, Equation (5.30) is used to compute the quantity $\frac{\partial P(Y_i|\mathcal{M})}{\partial w}$, that is the likelihood L_i of i-th observation sequence Y_i.

Note that, in the definition of criterion C'', different choices for the values $\lambda_1, \ldots, \lambda_s$ are possible:

1. A single value λ, shared among all the weights of the ANN, is used. This is similar to "Case 1" of the algorithm to train amplitudes (see Chapter 3), and is close to the regularization technique of weight decay.

2. As in the previous case, a single λ is used, but it is applied to the weights of the output layer only, i.e. a null value of λ_j is adopted for all lower layers (equivalent to a uniform distribution for weights in non-output layers). This emphasizes that the divergence problem mostly concerns the topmost layer, and the training scheme for lower layers can be simplified consequently.

3. Different values λ_l can be defined on a layer-by-layer basis, i.e. each layer is characterized by its own regularizing constant or, in probabilistic terms, the distribution of weight values is layer-dependent. This is similar to "Case 2" of the algorithm to train amplitudes, and coincides with the conclusions drawn by David MacKay within his Bayesian framework (namely, that each layer of the ANN should be given an *ad hoc* regularizing constant in order to increase Bayesian *evidence* of the model). Note that the previous case can be interpreted as a particular instance of the present case.

4. Different λ_j's, for $j = 1, \ldots, s$ can be used on an individual weight basis, i.e. each weight is drawn from a Gaussian pdf having its own variance. This reminds us of "Case 3" of the trainable amplitudes algorithm. A kind of *weight grouping* (Bishop, 1995) can be introduced by imposing that certain subsets of the weights, independently from the layer they belong to, do share the same λ value. In spite of its apparent flexibility, this case has several drawbacks, e.g. difficulty to choose actual λ_j values for individual weights, increased complexity, etc.

Due to the difficulty of finding an adequate parametric representation for the pdf that characterizes the distribution of connection weight, i.e. for the value λ, the present approach may turn out to be more effective when used to extremize the Maximum A Posteriori criterion (Section 5.4.4), instead of the bare Maximum Likelihood. In that context, the SWS-ML is no longer intended as a way to escape the divergence problem, but rather as a regularization technique, capable to improve generalization properties of the ANN/HMM hybrid.

5.4.3 Factorizing using Bayes' theorem

Using Bayes' theorem (Duda and Hart, 1973) we can rewrite the emission probability for state q_j at time t over t-th acoustic frame $\mathbf{y}_t$ as:

$$b_{j,t}(\mathbf{y}_t) = Pr(\mathbf{y}_t \mid q_j) = \frac{Pr(q_j \mid \mathbf{y}_t)p(\mathbf{y}_t)}{\Pi_j} \tag{5.47}$$

where $p(\mathbf{y}_t)$ is the value of the pdf of the acoustic observations in the feature space, evaluated over input $\mathbf{y}_t$, and $\Pi_j = Pr(q_j)$ is the prior probability of state q_j. For notational convenience, we write $Pr(.)$ to denote a probability, and $p(.)$ to denote a pdf. The distribution $p(.)$ and the quantities Π_j for $j = 1, \ldots, Q$ are independent of time, i.e. they refer to the whole universes of possible acoustic vectors and HMM states, regardless of the time they occur. Throughout this Section the meaning of Q is slightly different, since we need to consider *all* the states in the HMM, not only those actually implied in the current phonetic transcription. Assuming that, according to their definitions, the forward and backward terms are null for non-implied states, then all of the formulas developed in the previous Sections still hold true. Note that Equation (5.47) holds (in its first part, i.e. $b_{j,t}(\mathbf{y}_t) = Pr(\mathbf{y}_t \mid q_j)$) only under the assumption, implicit in the definition of HMM, that the emission probability associated with a specific state is independent of time. The second part of the equation, namely $Pr(\mathbf{y}_t \mid q_j) = \frac{Pr(q_j|\mathbf{y}_t)p(\mathbf{y}_t)}{\Pi_j}$, is always valid; but, removing that assumption, it would be useless in the present setup.

In order to use Equation (5.47) in the training algorithm, feasible ways to estimate the quantities involved in the factorization are sought. The estimation is accomplished as follows:

- $p(\mathbf{y})$ models the distribution of all acoustic vectors within the acoustic feature space. Given the training data, and pooling all the acoustic frames $\mathbf{y}$ regardless of the observation sequence they belong to and the time they occur, estimating $p(\mathbf{y})$ is a typical *density estimation* problem. The latter can be approached relying on statistical parametric (e.g. maximum likelihood, minimax or Bayesian learning) or non-parametric (e.g. *k-means* clustering) techniques (Duda and Hart, 1973), or using connectionist models (unsupervised ANNs, such as *competitive neural networks* or *Learning Vector Quantization*) (Hertz et al., 1991). Maximum-likelihood estimation of the parameters of a mixture of Gaussian components will be considered as the reference approach in the present Thesis. Estimation is carried out on the whole training set, before actually starting the training of the HMM/ANN hybrid.

- The state-priors Π_j for $j = 1, \ldots, Q$ can be estimated before starting the training, as well, over the whole training set and on the basis of high-level considerations concerning the task domain, the linguistic constraints (e.g. vocabulary and grammar) and the topology of the HMM (including the initial and, possibly, the transition probabilities of the latter whenever fixed or known in advance). For instance, if 1-state models are used to represent the digits $0, \ldots, 9$ in a digit recognition task, and if digits are expected to share equal probability in the application domain and to have the same average length in time, then $\Pi_j = 0.1$ for $j = 1, \ldots, 10$. If no high-level constraints or statistics on the linguistic domain are known, the state-priors can be estimated from the relative frequencies of individual states in the correct concatenations of HMM associated with the phonetic transcriptions of the training utterances (the same concatenations used for standard Baum-Welch training). For instance, if a 1-state model is used to represent the "background noise" or the "silence" between adjacent words in a continuous speech task, its (unique) state is expected to have a high prior probability (definitely higher than the average), since it is present at the beginning and at the end of nearly every utterance, as well as between several pairs of adjacent words. More details on a technique to estimate the state-priors is presented in Section 5.5.

- The estimation of the quantity $Pr(q_j \mid \mathbf{y}_t)$ is critical and much more difficult than the previous. We will use an ANN to carry out the estimation. More precisely, a "normalized" version of the ANN outputs is considered, by taking:

$$Pr(q_j \mid \mathbf{y}_t) = \frac{f_j(x_j(t))}{\sum_{i=1}^{Q} f_i(x_i(t))} \tag{5.48}$$

  for $j = 1, \ldots, Q$ (an ANN output for each HMM state is used), where $f_i(x_i(t))$ denotes the i-th output from the ANN when fed with input $\mathbf{y}_t$, as usual. The normalization of the ANN outputs is a suitable way to ensure that the estimates of state-posterior probabilities sum to one ($\sum_{i=1}^{Q} Pr(q_j \mid \mathbf{y}_t) = 1$). The only other constraint that must be satisfied in order to ensure a correct probabilistic interpretation of ANN outputs is that each output ranges in the $[0, 1]$ interval. Activation functions of output neurons must be chosen accordingly. The sigmoid is perfectly appropriate. Note how the Bayes' factorization has moved the focus from the unfeasible problem of constraining an ANN to estimate a pdf, to the viable requirement of ensuring proper connectionist posterior probabilities estimation. It should be noticed that, although the ANN is now used to estimate state-posterior probabilities, the overall approach is still far from Bourlard and Morgan's paradigm. Indeed, the probabilistic quantities estimated by the ANN are different, and different is the role they play in the ANN/HMM calculations. In addition, we keep on moving within a "joint" ANN/HMM training framework, aiming at the optimization of a global criterion, and no heuristic scheme is used. Finally, the present "normalization" of network outputs is mathematically motivated (and does not require any step back from the original theoretical framework).

From now on, efforts are concentrated on modifying the calculations developed in Section 5.2 according to the new role of the ANN. Equations (5.1) through (5.9) are not affected by the present discussion, and they still hold. Starting from Equation (5.10) the calculation is accomplished by taking the factorization (5.47) into account. The partial derivative of the emission probability $b_{j,t}$ with respect to a *generic* weight w is now given by:

$$\frac{\partial b_{j,t}}{\partial w} = \frac{\partial}{\partial w}\left\{\frac{Pr(q_j \mid \mathbf{y}_t)p(\mathbf{y}_t)}{\Pi_j}\right\} \tag{5.49}$$

$$
\begin{aligned}
&= \frac{p(\mathbf{y}_t)}{\Pi_j}\frac{\partial Pr(q_j \mid \mathbf{y}_t)}{\partial w} \\
&= \frac{p(\mathbf{y}_t)}{\Pi_j}\frac{\partial}{\partial w}\left\{\frac{f_j(x_j(t))}{\sum_{i=1}^{Q} f_i(x_i(t))}\right\} \\
&= \frac{p(\mathbf{y}_t)}{\Pi_j}\left[\sum_{i=1}^{Q} f_i(x_i(t))\right]^{-2}\left\{\sum_{i=1}^{Q} f_i(x_i(t))\frac{\partial f_j(x_j(t))}{\partial w} - f_j(x_j(t))\sum_{i=1}^{Q}\frac{\partial f_i(x_i(t))}{\partial w}\right\}.
\end{aligned}
$$

For a weight $w = w_{\iota k}$ in the topmost (output) layer, the same calculation carried out in Equation (5.10) allows us to write:

$$\frac{\partial f_\iota(x_\iota(t))}{\partial w_{\iota k}} = f'_\iota(x_\iota(t))o_k(t) \tag{5.50}$$

while, for $i = 1, \dots, Q$, $i \neq \iota$, it is immediately seen that

$$\frac{\partial f_i(x_i(t))}{\partial w_{\iota k}} = 0. \tag{5.51}$$

Applying definition (5.11), Equation (5.49) takes now the following form:

$$\frac{\partial b_{j,t}}{\partial w_{\iota k}} = \frac{p(\mathbf{y}_t)}{\Pi_j}\left[\sum_{i=1}^{Q} f_i(x_i(t))\right]^{-2}\left\{\delta_\iota(j,t)o_k(t)\sum_{i=1}^{Q} f_i(x_i(t)) - f_j(x_j(t))\sum_{i=1}^{Q}\delta_\iota(i,t)o_k(t)\right\}. \tag{5.52}$$

Let us now turn attention to a generic weight $w = w_{kl}$ in the first hidden layer. Calculations in Equation (5.13) can be followed, yielding:

$$\frac{\partial f_j(x_j(t))}{\partial w_{kl}} = f'_j(x_j(t))w_{jk}f'_k(x_k(t))o_l(t) \tag{5.53}$$

for $j = 1, \dots, Q$. Definition (5.14) and Equation (5.53) enable us to rewrite Equation (5.49) as:

$$\frac{\partial b_{j,t}}{\partial w_{kl}} = \frac{p(\mathbf{y}_t)}{\Pi_j}\left[\sum_{i=1}^{Q} f_i(x_i(t))\right]^{-2}\left\{\delta_k(j,t)o_l(t)\sum_{i=1}^{Q} f_i(x_i(t)) - f_j(x_j(t))\sum_{i=1}^{Q}\delta_k(i,t)o_l(t)\right\}. \tag{5.54}$$

Finally, let us consider a connection weight in the following hidden layers. As in Section 5.2, the calculations are accomplished for a generic weight $w = w_{lm}$ holding between m-th unit in layer $\mathcal{L}_{n-3}$ and l-th unit in layer $\mathcal{L}_{n-2}$, and the same results apply also to subsequent layers ($\mathcal{L}_{n-4}, \mathcal{L}_{n-5}, \dots, \mathcal{L}_0$).

The calculations in Equations (5.16) and (5.17) can still be used to develop the quantity $\frac{\partial f_j(x_j(t))}{\partial w_{lm}}$:

$$\frac{\partial f_j(x_j(t))}{\partial w_{lm}} = f'_j(x_j(t)) \sum_{i \in \mathcal{L}_{n-1}} w_{ji} f'_i(x_i(t)) w_{il} f'_l(x_l(t)) o_m(t) \tag{5.55}$$

for $j = 1, \ldots, Q$. Note that for lower layers the summation must be accomplished over the units belonging to the immediately upper layer. Again, definition (5.19) can be applied to Equation (5.55), allowing us to rewrite Equation (5.49) in the form:

$$\frac{\partial b_{j,t}}{\partial w_{lm}} = \frac{p(\mathbf{y}_t)}{\Pi_j} \left[\sum_{i=1}^{Q} f_i(x_i(t)) \right]^{-2} \left\{ \delta_l(j,t) o_m(t) \sum_{i=1}^{Q} f_i(x_i(t)) - f_j(x_j(t)) \sum_{i=1}^{Q} \delta_l(i,t) o_m(t) \right\}. \tag{5.56}$$

Expanding Equation (5.9) according to Equations (5.52), (5.54) or (5.56), and substituting into Equation (5.7) yields the new *learning rule* for a generic weight w_{jk} in the ANN:

$$\begin{aligned} \Delta w_{jk} &= \eta \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \frac{p(\mathbf{y}_t)}{\Pi_i} \left[\sum_{m=1}^{Q} f_m(x_m(t)) \right]^{-2} \times \\ & \times \left\{ \delta_j(i,t) o_k(t) \sum_{m=1}^{Q} f_m(x_m(t)) - f_i(x_i(t)) \sum_{m=1}^{Q} \delta_j(m,t) o_k(t) \right\} \end{aligned} \tag{5.57}$$

where the notation for the indexes has been properly modified to avoid ambiguities. In the following, we will refer to the present training scheme as the *Bayes* training algorithm for the ANN/HMM hybrid.

This approach is intrinsically discriminative, since weights for states not implied in the current transcription are modified to increase the likelihood of implied states.

In the batch case, criterion (5.23) is adopted and Equation (5.26) still holds, but the mathematics developed in the present Section must be used to compute the term $\frac{\partial P(Y_i|\mathcal{M})}{\partial w}$ to be substituted into (5.26), i.e.

$$\begin{aligned} \frac{\partial P(Y_l \mid \mathcal{M})}{\partial w_{jk}} &= \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \frac{p(\mathbf{y}_{lt})}{\Pi_i} \left[\sum_{m=1}^{Q} f_m(x_m(t)) \right]^{-2} \times \\ & \times \left\{ \delta_j(i,t) o_k(t) \sum_{m=1}^{Q} f_m(x_m(t)) - f_i(x_i(t)) \sum_{m=1}^{Q} \delta_j(m,t) o_k(t) \right\} \end{aligned} \tag{5.58}$$

where $\mathbf{y}_{lt}$ denotes t-th acoustic vector of l-th observation sequence Y_l. The other terms of Equation (5.58) (e.g. $\alpha_{i,t}$, $\beta_{i,t}$, $x_m(t)$, etc.) are those obtained while processing Y_l on the trellis in the usual manner.

Concerning the algorithm in the logarithmic domain, consider again criterion (5.1). Equations (5.27) through (5.30) still hold. Let us now concentrate efforts on the term $\frac{\partial \log b_{jt}}{\partial w}$ for j-th emission probability at time t, and for a generic ANN weight w. From Equation (5.47) we can write:

$$\frac{\partial \log b_{jt}}{\partial w} = \frac{\partial}{\partial w} \left\{ \log Pr(q_j \mid \mathbf{y}_t) + \log p(\mathbf{y}_t) - \log \Pi_j \right\}. \tag{5.59}$$

Since $p(\mathbf{y}_t)$ and Π_j do not depend on w, Equation (5.59) reduces to

$$\frac{\partial \log b_{jt}}{\partial w} = \frac{\partial}{\partial w} \log Pr(q_j \mid \mathbf{y}_t) \tag{5.60}$$

that, according to Equation (5.48), can be rewritten in the following form:

$$\begin{aligned} \frac{\partial \log b_{jt}}{\partial w} &= \frac{\partial}{\partial w} \left\{ \log f_j(x_j(t)) - \log \sum_{i=1}^{Q} f_i(x_i(t)) \right\} \\ &= \frac{1}{f_j(x_j(t))} \frac{\partial f_j(x_j(t))}{\partial w} - \frac{1}{\sum_{i=1}^{Q} f_i(x_i(t))} \sum_{i=1}^{Q} \frac{\partial f_i(x_i(t))}{\partial w}. \end{aligned} \tag{5.61}$$

If $w = w_{\iota k}$, being k a generic unit in the first hidden layer and $\iota \in \{1, \ldots, Q\}$, Equations (5.50) and (5.51) hold and Equation (5.61) becomes

$$\frac{\partial \log b_{jt}}{\partial w_{jk}} = \frac{1}{f_j(x_j(t))} \delta_\iota(j,t) o_k(t) - \frac{1}{\sum_{i=1}^{Q} f_i(x_i(t))} \delta_\iota(i,t) o_k(t) \tag{5.62}$$

where definition (5.11) has been applied.

For a generic weight $w = w_{kl}$ in the first hidden layer, Equation (5.53) holds, and Equation (5.61) can be rewritten in the form:

$$\frac{\partial \log b_{jt}}{\partial w_{kl}} = \frac{1}{f_j(x_j(t))} f_j'(x_j(t)) w_{jk} f_k'(x_k(t)) o_l(t) - \frac{1}{\sum_{i=1}^{Q} f_i(x_i(t))} \sum_{i=1}^{Q} f_i'(x_i(t)) w_{ik} f_k'(x_k(t)) o_l(t) \tag{5.63}$$

that, applying definition (5.14), reduces to:

$$\frac{\partial \log b_{jt}}{\partial w_{kl}} = \frac{1}{f_j(x_j(t))} \delta_k(j,t) o_l(t) - \frac{1}{\sum_{i=1}^{Q} f_i(x_i(t))} \sum_{i=1}^{Q} \delta_k(i,t) o_l(t) \tag{5.64}$$

which is in the same form of Equation (5.62).

Finally, if a generic weight $w = w_{lm}$ in any of the following hidden layers is considered, then Equation (5.55) holds, and from Equation (5.61) we can write:

$$\begin{aligned} \frac{\partial \log b_{jt}}{\partial w_{lm}} &= \frac{1}{f_j(x_j(t))} f_j'(x_j(t)) \sum_{i \in \mathcal{L}_{n-1}} w_{ji} f_i'(x_i(t)) w_{il} f_l'(x_l(t)) o_m(t) - \frac{1}{\sum_{\iota=1}^{Q} f_\iota(x_\iota(t))} \times \\ &\quad \times \sum_{\iota=1}^{Q} f_\iota'(x_\iota(t)) \sum_{i \in \mathcal{L}_{n-1}} w_{\iota i} f_i'(x_i(t)) w_{il} f_l'(x_l(t)) o_m(t) \end{aligned} \tag{5.65}$$

that, after application of definition (5.19), can be rewritten as:

$$\frac{\partial \log b_{jt}}{\partial w_{lm}} = \frac{1}{f_j(x_j(t))}\delta_l(j,t)o_m(t) - \frac{1}{\sum_{i=1}^{Q} f_i(x_i(t))}\sum_{i=1}^{Q}\delta_l(i,t)o_m(t) \tag{5.66}$$

which, once again, shares the same form of Equations (5.62) and (5.64). As usual, an on-line learning rule in the logarithmic domain for a generic weight $w_{\iota k}$ can be obtained from Equation (5.30), using Equation (5.62), (5.64), or (5.66) to compute $\frac{\partial \log b_{it}}{\partial w_{\iota k}}$ according to the fact that $w_{\iota k}$ is in the output layer, first hidden layer, or lower hidden layers, respectively, as follows:

$$\Delta w_{\iota k} = \eta \sum_{i=1}^{Q}\sum_{t=1}^{T}\beta_{i,t}\alpha_{i,t}\left\{\frac{1}{f_i(x_i(t))}\delta_\iota(i,t)o_k(t) - \frac{1}{\sum_{j=1}^{Q} f_j(x_j(t))}\sum_{j=1}^{Q}\delta_\iota(j,t)o_k(t)\right\} \tag{5.67}$$

where the indexes have been properly adjusted. Note that learning rule (5.67) is simpler than rule (5.57), i.e. computation in the logarithmic domain may appear again more promising than the corresponding computation in the linear domain.

As far as concerns the batch training in the logarithmic domain, criterion (5.23) is considered. Equation (5.26) can thus be used, relying on Equation (5.30) to compute $\frac{\partial P(Y_i|\mathcal{M})}{\partial w}$, and applying Equations(5.62), (5.64), or (5.66) to compute $\frac{\partial \log b_{i,t}}{\partial w}$ accordingly.

5.4.4 Using the Maximum A Posteriori discriminative training criterion

The adoption of a "discriminant" training criterion, namely the *Maximum A Posteriori* (MAP) criterion (Bengio, 1996), instead of the bare Maximum Likelihood (ML), appears promising from two distinct viewpoints: (i) divergence of ANN outputs is avoided, by forcing probabilities along "incorrect" paths of the HMM to be small in order to increase the posterior probability of the correct model given the acoustic observations; (ii) better overall performance is expected, due to the discriminative training procedure (Bourlard and Morgan, 1994a). The MAP criterion was successfully used in a similar manner by Bengio to tackle the problem, stated at the beginning of the present Section, of Gaussian pdfs that reduce to Dirac's Deltas in his model. Once the MAP criterion has been fixed, a new learning rule is obtained that inherits the same calculations and properties of the training schemes developed in the ML setup, where the ANN may be used to estimate the emission probabilities (with or without probabilistic constraints on its weights) or the state-posterior probabilities (as in the previous Section), i.e. the MAP criterion can be combined with any techniques among those described so far.

We formally define the MAP criterion $\mathcal{C}_{MAP}$ as follows:

$$\mathcal{C}_{MAP} = P(\mathcal{M} \mid Y) = \frac{P(Y \mid \mathcal{M})P(\mathcal{M})}{P(Y)} \tag{5.68}$$

where Y is the observation sequence, as usual, and the meaning of $\mathcal{M}$ (the model) is the same as in Section 5.3.2. Bayes' theorem was applied in order to obtain the right-end side of equation (5.68): $P(\mathcal{M})$ is the *prior* probability of the model, and $P(Y)$ is the overall likelihood of the acoustic observation sequence Y, that does not depend on the specific choice for the model $\mathcal{M}$.

$P(\mathcal{M})$ is independent from the ANN outputs, but it depends only on the language model. It can thus be computed separately (if individual word models having equal prior probabilities are considered, it may be even dropped from Equation (5.68)). In particular, in the on-line learning case a suitable training algorithm is required to maximize the following, equivalent criterion:

$$\mathcal{C}'_{MAP} = \frac{P(Y \mid \mathcal{M})}{P(Y)}, \tag{5.69}$$

a formulation that emphasizes the strict relationship with the *Maximum Mutual Information* (MMI) criterion (Bengio, 1996) (extremization is herein accomplished in the linear domain, instead of logarithmic). A gradient-based learning rule for a generic connection weight w can be obtained by taking the partial derivative of $\mathcal{C}'_{MAP}$ with respect to w as follows:

$$\begin{aligned}
\Delta w &= \eta \frac{\partial \mathcal{C}'_{MAP}}{\partial w} \\
&= \eta \frac{\partial}{\partial w}\left\{\frac{P(Y \mid \mathcal{M})}{P(Y)}\right\} \\
&= \frac{\eta}{[P(Y)]^2}\left\{P(Y)\frac{\partial P(Y \mid \mathcal{M})}{\partial w} - P(Y \mid \mathcal{M})\frac{\partial P(Y)}{\partial w}\right\} \\
&= \frac{\eta}{P(Y)}\left\{\frac{\partial P(Y \mid \mathcal{M})}{\partial w} - \frac{P(Y \mid \mathcal{M})}{P(Y)}\frac{\partial P(Y)}{\partial w}\right\}.
\end{aligned} \tag{5.70}$$

In practice, the quantity $P(Y \mid \mathcal{M})$ and its derivative with respect to w are computed according to the calculations introduced in the previous Sections, since it is the usual likelihood of the observations given the model $\mathcal{M}$, the latter being the "correct" model, i.e. the model *constrained* by the (known) transcription of acoustic observation sequence Y. Accordingly, the ANN can be used to estimate the emission probabilities or the state-posteriors, both in linear or logarithmic domains. The quantity $P(Y)$ and its derivative with respect to w are computed by applying the same calculations to an *unconstrained*, or *recognition* model, as explained in (Bengio, 1993; Trentin et al., 1998). The latter is the same model used for *recognition* with the Viterbi algorithm, where no prior knowledge on the exact transcription of the observations under consideration is given. It is representative of all allowable paths through the HMM states. For instance, if the recognition task consists of c isolated words modeled via c distinct left-to-right HMMs, the unconstrained model can be obtained by stacking into a single trellis the c disjoint models (with null transition probabilities between pairs of them). We will refer to the present training scheme as the *MAP* learning algorithm for the ANN/HMM hybrid.

From a qualitative point of view, the maximization of the MAP criterion can be interpreted as the formalization of the familiar idea of modifying the connection weights in order to increase the output over "positive" samples, while lowering the output over "negative" ones. This idea is commonly applied to ANNs trained to perform pattern recognition tasks. In the present setup, an acoustic observation sequence Y represents a positive sample for the corresponding constrained (correct) model $\mathcal{M}$, while it represents a negative sample for the unconstrained model (i.e., for "all the other models"). In this perspective, it is worth observing that, given equation (5.70), whenever the likelihood of Y given the correct model is low, the weight w is modified in order to extremize the MAP criterion by increasing $P(Y \mid \mathcal{M})$, nearly regardless of the behavior of the ANN over the unconstrained model. When, on the contrary, $P(Y \mid \mathcal{M})$ is already high, the MAP criterion is mostly increased by lowering the likelihood of Y over the unconstrained model. In particular, divergence of ANN outputs is avoided. In all the other mid-range cases, the learning rule for w imposes modifications that are a trade-off between maximization of $P(Y \mid \mathcal{M})$ and minimization of the likelihood of Y over the unconstrained model, realizing the discriminative training that was sought. In passing, note also that, for implementation purposes, it may be convenient to define an alternative learning rate $\eta' = \eta / P(Y)$, i.e. the likelihood $P(Y)$ can be dropped and considered "absorbed" within the learning rate itself, given the arbitrariness of the latter.

Equation (5.70) defines an *on-line* learning rule based on the MAP criterion. In the *batch* case, considering again the setup introduced in Section 5.3.2, with n observation sequences $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_n\}$, and maximizing the C_{MAP} criterion over $\mathcal{Y}$, a learning rule similar to equation (5.70) can be obtained, namely

$$\Delta w = \frac{\eta}{[P(\mathcal{Y})]^2} \left\{ P(\mathcal{Y}) \frac{\partial P(\mathcal{Y} \mid \mathcal{M})}{\partial w} - P(\mathcal{Y} \mid \mathcal{M}) \frac{\partial P(\mathcal{Y})}{\partial w} \right\} \tag{5.71}$$

where: the likelihood $P(\mathcal{Y})$ is calculated as the product (joint probability) of the likelihoods $P(Y_i)$, $i = 1, \ldots, n$ of the individual observation sequences; $P(\mathcal{Y} \mid \mathcal{M})$, in a similar manner, is calculated as the product (joint probability) of the quantities $P(Y_i \mid \mathcal{M})$, $i = 1, \ldots, n$; the derivative $\frac{\partial P(\mathcal{Y} \mid \mathcal{M})}{\partial w}$ is computed as outlined in the previous Sections for the batch case, with the caution of using the correct (constrained) model $\mathcal{M}_i$ for each $Y_1, \ldots, Y_n$ (the symbol $\mathcal{M}$ is intended here as the *set* of correct models for the different observation sequences); finally, $\frac{\partial P(\mathcal{Y})}{\partial w}$ is computed applying the above formulas for the batch case to the unconstrained model.

Applying Equations (5.24-5.26), to the constrained model $P(\mathcal{Y} \mid \mathcal{M})$ and to the unconstrained model $P(\mathcal{Y})$, respectively, we can write:

$$\frac{\partial P(\mathcal{Y} \mid \mathcal{M})}{\partial w} = P(\mathcal{Y} \mid \mathcal{M}) \sum_{i=1}^{n} \{< Const >_i\} \tag{5.72}$$

and

$$\frac{\partial P(\mathcal{Y})}{\partial w} = P(\mathcal{Y}) \sum_{i=1}^{n} \{< Unconst >_i\} \tag{5.73}$$

where the $< Unconst >_i$ and $< Const >_i$ terms are properly inherited from Equation (5.26). Equation (5.71) can be rewritten as:

$$\begin{aligned} \Delta w &= \frac{\eta}{[P(\mathcal{Y})]^2} \left\{ P(\mathcal{Y}) P(\mathcal{Y} \mid \mathcal{M}) \sum_{i=1}^{n} \{< Const >_i\} - P(\mathcal{Y} \mid \mathcal{M}) P(\mathcal{Y}) \sum_{i=1}^{n} \{< Unconst >_i\} \right\} \\ &= \eta \frac{P(\mathcal{Y} \mid \mathcal{M})}{P(\mathcal{Y})} \left\{ \sum_{i=1}^{n} \{< Const >_i - < Unconst >_i\} \right\} \end{aligned} \tag{5.74}$$

and a learning rate $\eta' = \eta \frac{P(\mathcal{Y}|\mathcal{M})}{P(\mathcal{Y})}$ can be defined (i.e., as in the on-line case, likelihoods are absorbed within the learning rate itself) considerably simplifying the implementation and providing for increased numerical stability.

5.5 Experiments

Let us introduce the experimental setup investigated throughout the research. *Clean* speech signals from the *SPK* database[6], collected in laboratory conditions at ITC-irst, was considered. A close-talk microphone was used for the recordings, under quiet laboratory conditions. The corpus is divided into two parts: the first of them, called `idigits`, consists of 100 speakers (male and female), each repeating the *isolated* Italian digits (from 0 to 9) during 5 different sessions, for a total of 5000 signals. The second part (`cdigits`) includes continuous speech, namely 1000 utterances of connected Italian digit strings having length 8 (for a total of 8000 words), acquired over 40 different speakers (21 male and 19 female). Spectral analysis of the speech signals (acquired at a sampling rate of 16kHz) was accomplished over 20ms Hamming windows having an overlap of 10ms, in order to extract 8 *Mel Frequency Scaled Cepstral Coefficients* (MFSCCs) and the signal log-energy as acoustic features. The latter were normalized in order to have input values distributed in a uniform manner over the $[0, 1]$ interval, before feeding them into the connectionist models.

First of all, Section 5.5.1 focuses on a reduced-scale problem, obtained from the experimental setup introduced above, in order to analyze in detail the behavior of the different training techniques described so far whenever applied to different connectionist architectures. Comparisons among them and with a standard HMM are carried out, searching for the most promising to be applied over full-range ASR problems. An instance of the latter is then investigated in Section 5.5.3.

[6]SPK is available from the European Language Resources Association (ELRA).

5.5.1 Demonstration: reduced-scale problem

Since processing the whole SPK database for speaker independent, continuous speech recognition is computationally expensive, and evaluating the behavior of the proposed training schemes over different architectures is difficult when large amounts of data are used, we first introduce a simple case study drawn from SPK. Results obtained herein highlight the properties of the algorithms and provide us with significant cues on the most promising training schemes, parameters and architectures to be applied to the more complex ASR tasks.

The reduced-scale setup was built by choosing one individual speaker at random from SPK (speaker code: *gila0*), i.e. a *speaker dependent* recognizer is developed. A subset of the original feature space was considered, precisely a 3-dimensional space composed of the first two MFSCCs and the corresponding log-energy. The ASR task was the recognition of *isolated words*: 120 acoustic signals, corresponding to 12 different utterances of each Italian digit were used for training, while 80 isolated signals (8 instances for each digit) constituted the test set.

Three different approaches to the problem are presented in the following Sections, each corresponding to different initialization schemes for particular ANN architectures. First ("Round 1"), we investigate a feed-forward model with log-Gaussian activations in the hidden layer and linear outputs, to be initialized with the parameters of a standard HMM having a single Gaussian emission pdf associated with its states. Then ("Round 2"), a MLP with linear outputs is used, along with a particular initialization technique that relies on the log-emissions of the standard HMM. Finally ("Round 3"), the most general case is covered, i.e. a generic MLP with sigmoidal hidden and output units, initialized via a *Bourlard-like* procedure.

Round 1: ANN initialized with log-Gaussian kernels

A standard, simple HMM was first trained and evaluated, providing a comparison baseline for the following algorithms, as well as a set of parameters suitable for the *initialization* of certain connectionist models, as explained below. For initialization purposes, the HMM was kept as simple as possible (in the present scenario, attention is paid to the behavior and relative performance of different models, not to the absolute suitability of the recognition rate to real-world applications). The topology was a set of 10 word (digit) models, each based on a 3-state left-to-right HMM, plus a 1-state model for the background noise (silence model), for a total of 31 states. Emission probabilities were modeled via individual Gaussian pdfs $G_i(\mathbf{x})$ associated with each state i of the HMM, where $G_i(\mathbf{x}) = \prod_{j=1}^{3} g_{ij}(x_j)$ and $g_{ij}(x_j)$ is a univariate normal density, defined over j-th dimension of the feature space[7]. All such pdfs shared a common value θ for their variances θ_{ij}. The *segmental k-means* algorithm (Rabiner, 1989) was applied in order to initialize the HMM. Training

[7] In fact, standard HMMs assume independence among the components of the feature vectors, and multivariate Gaussians with diagonal covariances are used. This was pointed out in Section 1.2 as a potential drawback of standard HMMs with respect to ANNs.

was accomplished via the Baum-Welch algorithm, and the Viterbi decoding technique was applied over the test set. A 70.00% *Word Recognition Rate* (WRR) was obtained.

Architecture/algorithm	*WRR (%)*	*Final log-likelihood*
HMM	70.00	-7.5054e+03
BML	68.75	-7.0964e+03
SWS-ML	71.25	-7.4166e+03
BML with Early Stopping	73.75	-7.4248e+03
MAP	72.50	-7.7269e+03

Table 5.1: *Round 1, ANN with log-Gaussian kernels: Word recognition rate (WRR) on test set and final log-likelihood of the training acoustic observations given the model, yielded by the proposed algorithms applied to the SPK isolated digits, speaker-dependent (*gila0*) problem. 3-dim acoustic space: 2 MFSCCs and signal log-energy.*

The connectionist model was then defined as a two-layer feed-forward net, with 93 units in the hidden layer (one for each univariate Gaussian $g_{ij}(.)$ in the HMM, $i = 1, \ldots, 31$, $j = 1, \ldots, 3$), having the following form for the corresponding activation functions $f_{ij}(x)$ over a generic input x:

$$f_{ij}(x) = -\frac{1}{2}\left(\frac{x - \mu_{ij}}{\theta}\right)^2 = log(g_{ij}(x)) + c \tag{5.75}$$

where c is a constant (namely the logarithm of the usual normalization constant for Normal densities having standard deviation θ). Input-to-hidden weights $w_{ij,k}$, connecting k-th input unit with ij-th hidden unit, were defined as

$$w_{ij,k} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases} \tag{5.76}$$

for each $k = 1, \ldots, 3$. Hidden-to-output weights $w_{l,ij}$, connecting ij-th hidden unit with l-th output unit ($l = 1, \ldots, 31$), were defined as

$$w_{l,ij} = \begin{cases} 1 & \text{if } i = l \\ 0 & \text{otherwise} \end{cases} \tag{5.77}$$

along with linear output units, one for each state l of the HMM, thus realizing the function

$$o_l = \sum_{j=1}^{3} log(g_{lj}(x_j)) = log(G_l(\mathbf{x})). \tag{5.78}$$

The resulting architecture is a kind of *log-RBF* network, that yields exactly the same emission probabilities as the standard HMM involved in its construction process. It is suitable for those algorithms of Sections 5.2-5.4 which are defined in the logarithmic domain, i.e. where the ANN is expected to provide the logarithm of the emissions over the input vectors: this means that it cannot be used for evaluating the *Bayes* learning rule described in Section 5.4.3, since state-posterior probability estimates from the ANN outputs are required therein[8].

This initialization technique has the advantage of being quite simple and straightforward once a HMM has been trained, although the resulting architecture may not be expected to yield significant improvement over the bare HMM. Better results are obtained with the topologies introduced in the next Sections. Nonetheless, it can be easily applied in conjunction with the regularization algorithms described in Chapter 3, as well as with the inversion technique developed in Chapter 6, to provide improved generalization performance over the HMM, i.e. robustness toward noise.

It is worth noticing that, although in principle this model actually is a HMM with Gaussian emissions, during training the values of the weights (and biases) are modified. This means that: (i) the log-Gaussian kernels of the hidden layer are progressively fed with linear combinations of the input features (instead of being defined over separate, individual features), learning to exploit correlations between the acoustic features to some extent; (ii) the output units learn to yield values that are no longer based on the evaluation of a single normal pdf associated with the corresponding state, but also on the values of other Gaussians associated with other states (i.e., some Gaussian pdfs are shared among distinct states to better fit the corresponding emission distributions). These aspects are the rationale behind the improvement obtained in the experiments. Results yielded by the different training schemes are summarized in Table 5.1.

The BML learning rule (Equation 5.31) was first applied. Cross-validation techniques were used to fix the learning rate, i.e. $\eta = 9.0e - 06$, and to compare the on-line and the batch training cases. In particular, in this setup the batch learning rule turned out to be slightly advantageous, since in the on-line case all shorter training sequences (corresponding to utterances of shorter words) yielded higher likelihood values (being the latter the product of probabilities along shorter paths in the trellis), thus inducing heavier changes in ANN weights that biased the training process toward those words. A kind of "normalization" with respect to the length of each training sequence tackles the problem: such a normalization is indeed implicit in the batch learning rule (refer to Equation 5.26), where the contribution from each sequence is divided by the corresponding likelihood. Figure 5.1 shows the evolution of the overall log-likelihood of the training set given the model for the BML (upper curve) as a function of the number of training epochs. As pointed out in Section 5.4, the BML may continuously increase the likelihood, but at the expense of a degenerated model with unbound values for the connection weights in the hidden-to-output

[8] Attemptes were made to convert outputs from logarithmic to the linear domain via application of exponential activation functions in the output layer, but they practically failed due to numerical problems in the gradient-ascent (implied by the nature of this particular function).

layer: in fact, the word recognition rate is decreased with respect to the HMM at the end of training. Attempts to apply more than 20 training epochs resulted in the "explosion" of such pseudo-likelihood values (no longer manageable in the double-precision representation of the Unix workstation used for the simulations): from now on, we will restrict our attention to the first 20 epochs.

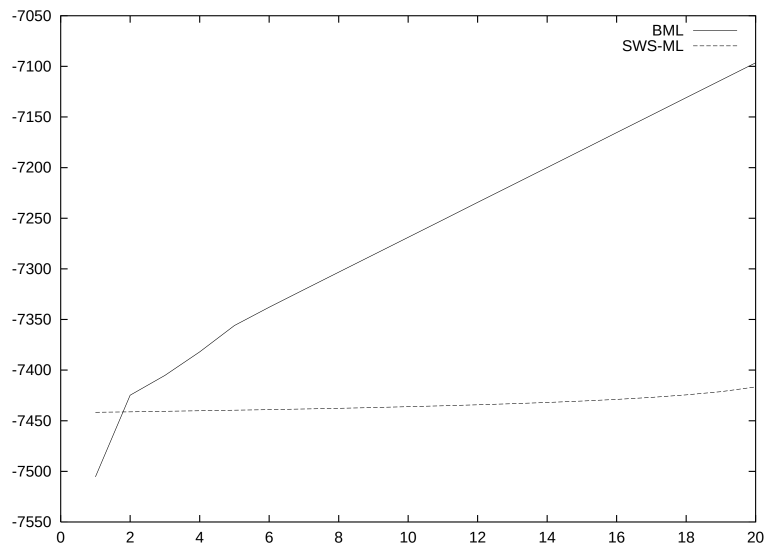

Figure 5.1: *Learning curves (log-likelihood of the acoustic observation sequences in the training set) as a function of the number of training epochs, for the BML and SWS-ML algorithms.*

The SWS-ML scheme (Equation 5.46) allowed to tackle the divergence problem, as expected. Values of $\lambda \approx 0.0005$ emerged to be effective from a preliminary cross-validation step. The evolution of the corresponding log-likelihood during the training process is represented by the lower curve in Figure 5.1. The final likelihood is lower than in the BML case, although increased with respect to the HMM. In particular, the likelihood increases very slowly (it may be even reduced at certain iterations): this is due to the fact that the SWS-ML algorithm aims at maximizing the joint probability of the acoustic observations *and* of the connection weights, i.e. when the latter tend to become too big, the learning rule forces a reduction in their size, even at the expense of the likelihood of the observations, avoiding divergence. Indeed, the WRR turns out to be improved.

It is interesting to note that, with the present simple architecture, better recognition results can be obtained with the BML scheme along with *early stopping* (Bishop, 1995). In particular, experiments carried out applying crossvalidation techniques showed that a single training epoch allows for improved performance due to the tendency of BML to increase the likelihood immediately, while the stopping mechanism prevents the model from divergence. Unfortunately, early stopping turns out to be effective only in the present scenario, since the particular initialization via HMM provides a significant starting point for the training procedure, but it is no longer suitable to generic ANN topologies, as pointed out in the following Sections.

Finally, the MAP variant of the algorithm improved recognition performance over the HMM, as well as over the SWS-ML scheme, by increasing quantity (5.69), i.e. the likelihood-ratio, while preventing the bare likelihood of the acoustic observations from divergence: the likelihood (of the constrained model) may even decrease at certain training epochs, whenever a broader reduction of the likelihood of the unconstrained model ensures an increment in the likelihood ratio. Figure 5.2 shows the log-likelihood of the training sequences given the constrained model (lower curve), and given the unconstrained model (upper curve), as functions of the number of training epochs. Evolution of the monotonically-increasing likelihood ratios, computed at each epoch over the training set, is plotted in Figure 5.3.

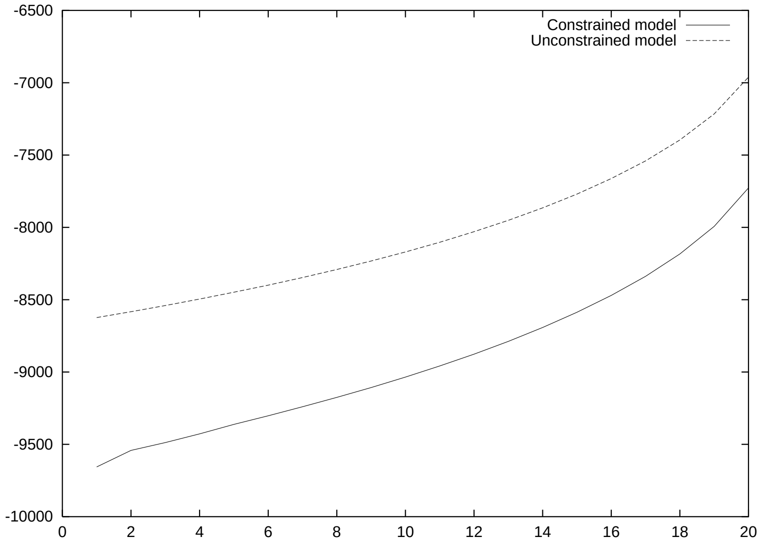

Figure 5.2: *Log-likelihood of the acoustic observation sequences in the training set, as a function of the number of training epochs, for the MAP algorithm. Upper curve: unconstrained model; lower curve: constrained model.*

Round 2: generic MLP, logarithmic domain

In this Section we apply the algorithms to the generic 2-layer MLP architecture with hidden sigmoidal activation functions and linear output units. The topology is inherited from the previous experiments, i.e. 93 hidden units along with 31 outputs are used. Results are reported in Table 5.2. The first row of the table reports the baseline performance obtained with a standard HMM with Gaussian emission probabilities (a single Gaussian for each state) having variable means and covariances. A second HMM with mixtures of 8 Gaussian components was then trained and evaluated, for initialization and comparison purposes (second row of the Table). Initialization of the MLP was accomplished according to the following algorithm:

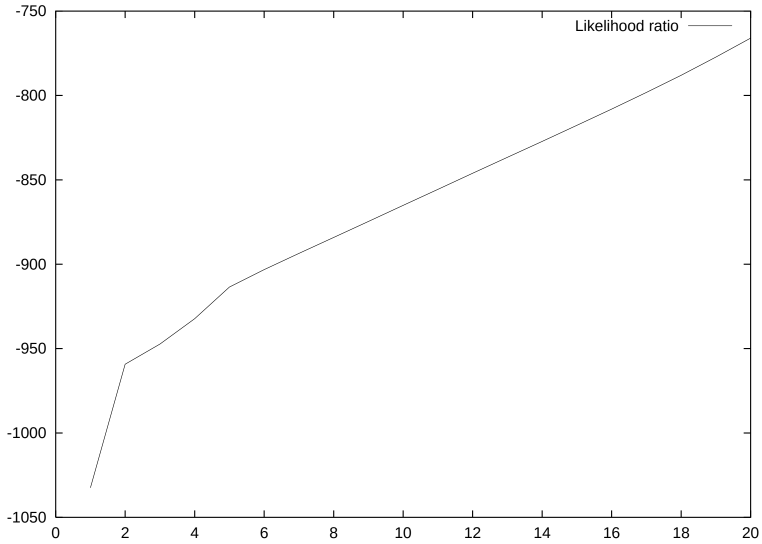

Figure 5.3: *Likelihood ratios between the constrained and the unconstrained models, evaluated over the acoustic observation sequences in the training set as a function of the number of training epochs (MAP algorithm).*

1. For each acoustic vector $\mathbf{x}$ in the training set, compute the value of the log-emission probabilities $y_i = log[b_i(\mathbf{x})]$, for $i = 1, \ldots, 31$, yielded by the HMM. Let $\mathbf{y} = (y_1, \ldots, y_{31})$.

2. Define a labeled training set $\mathcal{T} = \{(\mathbf{x}, \mathbf{y})\}$ including all such vector pairs.

3. Train the MLP with standard BP to minimize the *Mean Squared Error* over $\mathcal{T}$.

The MLP is basically bootstrapped in order to "mimic" the behavior of the HMM, providing for a reasonable approximation of log-emissions. This initialization scheme reminds us of Bourlard and Morgan's (Bourlard and Morgan, 1994a) training scheme. The results obtained applying Viterbi to the initialized net without any further hybrid-training procedure (third row of the Table) do validate the above initialization technique.

Given the nature of the output units and the fact that they are initialized to estimate logarithmic values, the present architecture (like that described in Round 1) is not suitable for application of the Bayes' learning rule described in Section 5.4.3. Instead, all the other training techniques (BML, SWS-ML and MAP in their logarithmic variants) can be used herein.

As expected from the lack of any architectural constraints in the present model, training the MLP with the BML algorithm without early stopping (not reported in the Table) indefinitely increased the likelihood, lowering the WRR in a systematic manner, even for choices of small learning rates. Application of early stopping (after

Architecture/algorithm	*WRR (%)*	*Final log-likelihood*
HMM with single Gaussians	77.50	1.1474e+04
HMM with 8-Gaussian mixtures	81.25	2.0520e+04
MLP initialized via BP	80.00	-9.1109e+04
SWS-ML	80.00	-9.1073e+04
BML with Early Stopping	81.25	-9.1052e+04
MAP	85.00	-9.3754e+04

Table 5.2: *Round 2, generic MLP with sigmoidal hidden units and linear outputs: WRR on test set and final log-likelihood of the training acoustic observations given the model, yielded by the proposed algorithms applied to the SPK isolated digits, speaker-dependent problem. 3-dim acoustic space: 2 MFSCCs and signal log-energy.*

only 1 epoch), as in the previous Section, allowed for a gain with respect to the bare initialized MLP, obtaining the same WRR as the HMM with Gaussian mixtures. In the SWS-ML case, divergence in likelihood values was avoided only by choosing very small values for λ, that did not allow for any tangible improvement over the initialized net.

Training in the logarithmic domain with the MAP criterion for 10 epochs (as suggested by preliminary cross-validation) in the on-line mode allowed for a significant WRR improvement over the standard HMM and the other hybrid-training techniques, yielding a 85.00% WRR. Evolution of the log-likelihood ratio $\log P(\mathcal{Y} \mid \mathcal{M}) - \log \mathcal{P}(\mathcal{Y})$ between the constrained and the unconstrained models is shown in Figure 5.4. Application of the *batch* MAP learning rule was less effective in the present setup, yielding a 82.50% WRR.

Round 3: generic MLP, linear domain

Finally, we apply a generic MLP to estimate probabilistic quantities in the linear domain, in order to evaluate the linear version of the training algorithms, as well as to allow application of the *Bayes* training scheme, which requires ANN outputs to be probabilities, i.e. in the range $[0, 1]$. To ensure probabilistic values in output, sigmoids were introduced in the output layer. The initialization procedure is similar to that described above in the logarithmic domain, but it may be improved as follows:

1. an initial segmentation of the training set is obtained on the basis of prior knowledge on the correct HMM associated with each utterance and on the Viterbi hypothesis of segmentation provided by the pre-trained standard HMM;

2. target values are assigned according to this segmentation, namely a positive

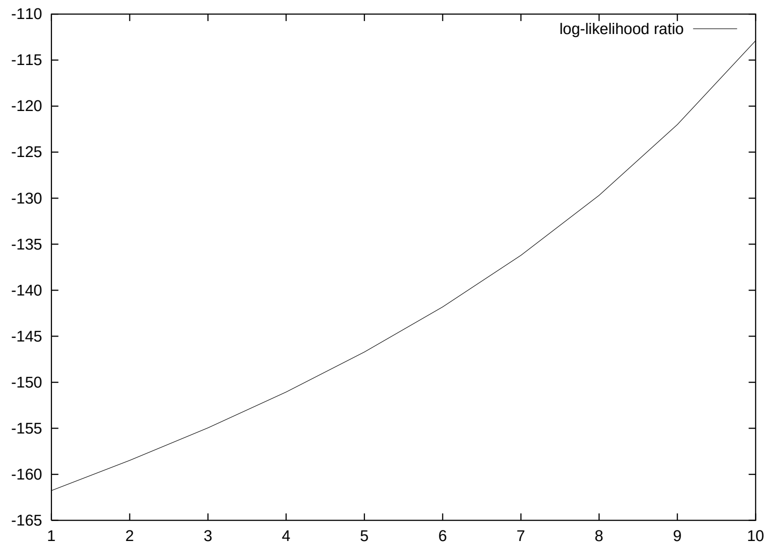

Figure 5.4: *Round 3: log-likelihood ratios between the constrained and the unconstrained models, evaluated over the training set as a function of the number of training epochs (MAP algorithm).*

target[9] for unit i is assigned over all patterns belonging to state i given the segmentation, while a null target is given to all the other units (discriminative approach);

3. BP training of the MLP is accomplished over the labeled training set;

4. the trained MLP is used within the hybrid architecture along with the Viterbi algorithm to yield a novel segmentation of the training sequences;

5. the last two steps of the algorithm may be repeated for a certain number of iterations.

Note that this initialization technique basically coincides with Morgan and Bourlard's hybrid training scheme (Bourlard and Morgan, 1994a), once a probabilistic interpretation of ANN outputs is given in terms of "state posterior probabilities", thus providing us with a comparison baseline for the new hybrid paradigm.

Table 5.3 shows the results obtained with the different techniques. The HMM with 8 Gaussians per state is the same used in round 2. The proposed initialization turns out to perform better than in the previous case, even improving over the standard HMM, i.e. Bourlard and Morgan's hybrid behaved better than the HMM, as expected. All the novel hybrid schemes improved over the HMM and over the initialized MLP, with the exception of the BML that increased the likelihood but

[9] A target value equal to "1" is the default, but larger values may help improving convergence of the algorithm in practice.

worsened the WRR (not reported in the table). In all cases, the on-line version of the algorithms yielded slightly better results than the batch case. A systematic improvement with respect to Round 2 is met. Again, early stopping (after 5 epochs) was effective in preventing the divergence problem.

The results do validate also the *Bayes* training scheme that could not be evaluated in the previous rounds. The state-priors Π_i to be used in the learning rule were estimated from the relative frequencies of the CDHMM states over the training set, during the initialization alignment procedure. No explicit estimation of the acoustic observations distribution $p(\mathbf{y})$ was needed to apply the *Bayes* algorithm herein: a uniform distribution of the features in the $[0, 1]$ interval was indeed guaranteed by the normalization of the input data. In Table 5.3, the term *MAP+Bayes* refers to the MAP scheme (constrained vs. unconstrained models, see Section 5.4.4) combined with the *Bayes* algorithm (Section 5.4.3), i.e. MLP outputs are assumed to be the state posterior probabilities given the input acoustic observation.

Architecture/algorithm	*WRR (%)*
HMM with 8-Gaussian mixtures	81.25
MLP initialized via *Bourlard-like algorithm*	83.75
BML with Early Stopping	85.00
SWS-ML	90.00
Bayes	87.50
MAP	91.25
MAP+Bayes	87.50

Table 5.3: *Round 3, generic MLP with sigmoidal hidden and output units: Word recognition rate (WRR) on test set yielded by the proposed algorithms applied to the SPK isolated digits, speaker-dependent problem. 3-dim acoustic space: 2 MFSCCs and signal log-energy.*

Figures 5.5 and 5.6 show the log-likelihood of the training set given the model and the WRR, respectively, as functions of the parameter λ of Equation (5.38) in the SWS-ML case. It is seen that a kind of "optimal" WRR is reached in correspondence with a value of λ that represents a tradeoff between higher likelihood values (i.e. the model tends to degenerate when the likelihood diverges), corresponding to wide Gaussian pdfs for the ANN weights, and lower likelihood values due to too narrow Gaussians, i.e. oversimplified models.

Finally, the WRR yielded by the MAP algorithm, in particular, turns out to be definitely sound, and the present architecture appears to be promising in a real-world application perspective.

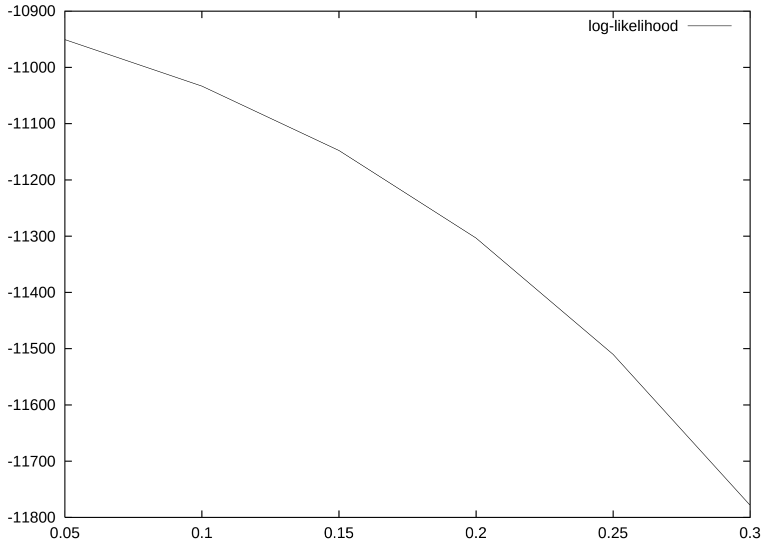

Figure 5.5: *Round 3, SWS-ML algorithm: log-likelihood evaluated over the training set as a function of the parameter* λ *that controls the probability distribution of network weights.*

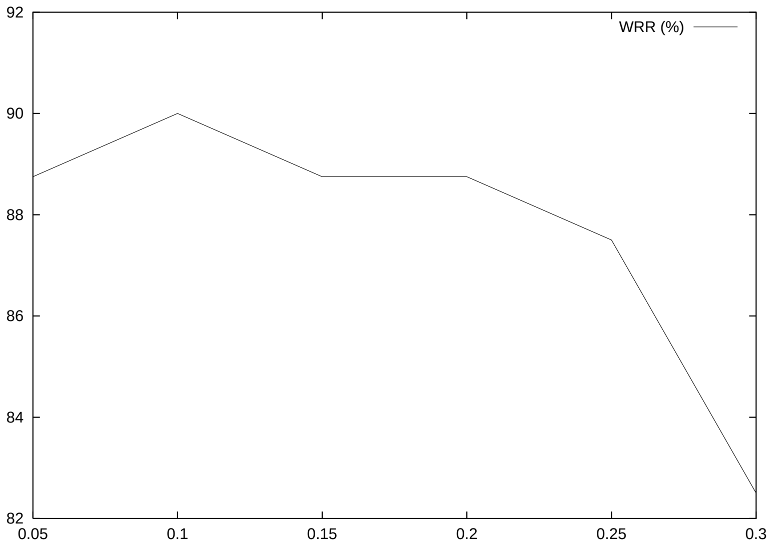

Figure 5.6: *Round 3, SWS-ML algorithm: WRR evaluated over the test set as a function of the parameter* λ.

5.5.2 Preliminary conclusions

From the above experimental investigation it turns out that:

1. The BML algorithm always increases the overall likelihood of the acoustic observations given the model, but an instance of the "divergence problem" systematically arises after a few training epochs, worsening the recognition performance (unless early stopping is applied).

2. All the proposed approaches to the divergence problem (namely SWS-ML, Bayes and MAP) allow for viable solutions, resulting in increased likelihood and improved recognition performance.

3. All of them actually improve over the standard CDHMMs. In the present setup, improvement is sound.

4. Although both on-line and batch learning rules do converge to similar solutions, the on-line version of the algorithms usually allows for better results than in the batch case. This fact is perfectly aligned with the analogous phenomenon that occurs in standard BP for static ANNs.

5. The MLP architectures (Rounds 2 and 3) outperform the log-RBF initialized via CDHMM parameters.

6. The MLP with output sigmoids jointly with the learning rules defined in the linear domain behaves better than the corresponding MLP with linear outputs and the logarithmic version of the algorithms.

7. Finally, the MAP version of the algorithms results particularly effective.

The information contained in the above list is exploited in the choice of architectures and algorithms to be applied in the speaker independent, continuous speech task described in the next Section.

5.5.3 Real-world ASR problem

On the basis of the preliminary conclusions drawn in Section 5.5.2, let us turn our attention to a real-world ASR problem, namely SI, continuous speech recognition. The task is the recognition of connected Italian digits strings from the `cdigits` part of the SPK database, introduced at the beginning of Section 5.5. The whole dataset was divided into two equally-sized subsets, to be used for training and test, respectively. Each dataset was thus composed of 500 utterances (each utterance consisting in 8 digits, for a total of 4000 words); 20 speakers (10 male and 10 female) were inserted in the training set, while the others (11 male and 9 female) formed the test set. The first 8 MFSCCs along with the signal log-energy were used as acoustic features (this is a usual choice in ASR systems), defining a 9-dimensional feature space.

Words in the dictionary were modeled using individual, left-to-right HMMs having 3 to 6 states, according to the phonetic transcription of each Italian digit, plus a 1-state model for the "silence" (or "background noise", *@bg*), as reported in Table 5.4, for a total of 40 states. The phonetic transcription was carried out using the SAMPA (Speech Assessment Methods Phonetic Alphabet) units, that is a machine-readable phonetic alphabet originally developed under the ESPRIT project *SAM* (Speech Assessment Methods).

Word	*Phonetic transcription*	*Number of HMM states*
@bg (silence)		1
uno	u n o	3
due	d u e	3
tre	t r e	3
quattro	k w a tt r o	6
cinque	tS i n k w e	6
sei	s e i	3
sette	s e tt e	4
otto	o tt o	3
nove	n o v e	4
zero	ts e r o	4

Table 5.4: *Phonetic transcription of the dictionary (in terms of SAMPA units) and corresponding number of HMM states for the Italian connected digits task.*

Mixtures of 8 Gaussian components were used to model emission probabilities for each state of the standard CDHMM (overall number of free parameters: 6080), that was initialized via Segmental k-Means and trained by applying the Baum-Welch algorithm. The topology of the MLP was kept as close as possible to that successfully applied over the reduced-scale problem in the linear domain, with 93 sigmoids in the hidden layer, the differences being the size of the input layer (augmented from 3 to 9 input units), and 40 sigmoidal outputs, one for each state of the underlying HMM (4690 free parameters).

The ANN architecture and the version of the novel hybrid training algorithms were chosen according to the guidelines emerged from the experiments on the reduced-scale problem, as summarized in Section 5.5.2. In practice, the *Bourlard-like* initialization technique was applied (Section 5.5.1, "Round 3") and training was accomplished via on-line SWS-ML, Bayes and MAP in the linear domain.

Since we are dealing with word *sequences*, further evaluation criteria may be introduced in addition to the WRR (or *unit accuracy*), that was defined in Chapter 1, Equation (1.2). Assuming that N_{words} is the total number of words in the test set, and that Sub, Ins, Del are the total numbers of substitutions, insertions and deletions, respectively, the performance of the recognition system may also be expressed by the following alternative measure, known as *Percent Correct* (PC):

$$PC = 100\frac{N_{words} - (Sub + Del)}{N_{words}}\% \tag{5.79}$$

which does not take into consideration insertion errors; indeed, Ins may be considerably reduced by imposing a suitable language model. Furthermore, evaluating the error rate at the word-strings level (i.e., the fraction of test sequences that are recognized without any errors), the *String Recognition Rate* (SRR) is obtained. It is crucial to observe that in continuous speech recognition the language model, defining the probability of moving from one word of the dictionary to another, acts as a balance between deletions and insertions, possibly yielding significantly different values for WRR, PC and SRR. In the present setup, since an objective comparison between the *acoustic* models is sought, the language model is implicitly defined in order to have the same number of insertions and deletions for any given acoustic model.

Experimental results are reported in Table 5.5. A comparison with Bourlard and Morgan's hybrid architecture is provided: it was carried out over the same ANN topology, starting from the same initial segmentation of training data (as explained above in "Round 3") labeled with 0/1 target values. The variant "Case 4" of the algorithm, described in (Bourlard and Morgan, 1994a) on page 164, was applied. Normalization of ANN outputs, i.e. division by the corresponding state-priors (estimated from the training set) in order to reduce to "likelihoods", was accomplished at recognition time, as recommended in (Bourlard and Morgan, 1994a), Chapter 7.

As far as concerns the novel ANN/HMM hybrid, the BML algorithm (not shown in the Table) did not allow for any significant results, even applying early stopping after a single training epoch. This is due to the increased cardinality of the training set, i.e., contributions from a number of input patterns increase the likelihood considerably, but bringing about the onset of the divergence problem. All the other variants of the algorithm (SWS-ML, Bayes and MAP) allowed for an improvement over the standard HMM, as well as over Bourlard and Morgan's approach. In particular, it is seen that the ANN/HMM recognizer trained with the MAP version of the algorithm yields a 46% WER reduction over the standard HMM.

5.6 Chapter Remarks

This Chapter introduced a novel ANN/HMM hybrid system where a feed-forward ANN carries out estimation of emission probabilities associated with the states of an underlying HMM topology. Initial and transition probabilities are estimated via Baum-Welch, while gradient-ascent techniques aimed at the maximization of the ML or MAP criteria were developed. The usual Viterbi decoding strategy is then applied at recognition time.

Architecture/algorithm	*SRR (%)*	*PC (%)*	*WRR (%)*
HMM with 8-Gaussian mixtures	46.60	93.10	**90.03**
Bourlard and Morgan's hybrid	47.40	93.20	**90.20**
ANN/HMM hybrid trained via SWS-ML	64.80	95.67	**93.90**
ANN/HMM hybrid trained via Bayes	60.60	95.47	**93.25**
ANN/HMM hybrid trained via MAP	68.60	96.33	**94.65**

Table 5.5: *String recognition rate (SRR), percent correct (PC) and word recognition rate (WRR) on test set yielded by the proposed algorithms applied to the SPK connected digits, speaker-independent problem. 9-dim acoustic space: 8 MFSCCs and signal log-energy.*

The basic version of the algorithm, called BML, required extensions in order to tackle the "divergence problem". For this reason, the SWS-ML, Bayes and MAP versions of the algorithm were proposed, providing us with viable solutions to the problem.

Reduced-scale experiments on a SD, isolated digits recognition task allowed for the analysis of the behavior of the training schemes, as well as for the investigation of different connectionist architectures along with corresponding *ad hoc* initialization techniques. In particular, the MAP algorithm applied in the linear domain to a MLP with sigmoidal hidden and output units compared favorably with: (a) standard CDHMMs; (b) Bourlard and Morgan approach; and (c) with all the other versions of the novel hybrid. The same conclusion could be drawn after direct comparison in the SI, continuous speech task.

At this point, the problem of robustness to noise has to be faced. Experiments on noisy speech data, namely connected digit recognition with noise, are required to evaluate the generalization abilities (i.e., robustness) of the system on the field. In addition, generalization may be explicitly improved via a modified training scheme, as outlined in Chapter 3 for the case of feed-forward and recurrent networks. The introduction of *trainable amplitudes* within the hybrid training algorithms may be expected to yield a similar benefit as that obtained in stand-alone ANNs. These topics are covered in the next Chapter.

Chapter 6

The Missing Ring: introducing Adaptive Amplitudes within the Hybrid Model

The scope of the present chapter is twofold: first of all, *robustness*, i.e. generalization capability of the novel ANN/HMM hybrid has to be evaluated (and compared with standard acoustic models) in recognition tasks over noisy data. Then, an explicit technique to improve learning and generalization of the hybrid is developed and, in turn, evaluated. In Chapter 3 a gradient-based training scheme was introduced to improve learning and generalization capabilities in feed-forward and recurrent networks, relying on the idea of adaptive amplitudes for non-linear activation functions. Three instances of the algorithm were developed, according to the fact that a unique amplitude over the whole network, or layer-by-layer amplitudes, or unit-by-unit amplitudes are considered. The training techniques, to be applied in parallel with standard BP for connection weights training, provided for significant experimental results, and a theoretical analysis was carried out to investigate the rationale behind such an improvement. In the present chapter, adaptive amplitudes are allowed within the novel ANN/HMM hybrid paradigms described and evaluated in Chapter 5, Since the motivations outlined in the case of conventional ANNs may be extended – in s straightforward manner – to the hybrid case, an improved training procedure is expected: capable to provide models that better suit the data at hand; and, above all, responsible for an increased generalization performance when tested on new data. The latter aspect is crucial: indeed, Chapter 4 pointed out that one of the potential advantages of the application of ANNs to ASR may be expected as the consequence of overcoming the lack of generalization capabilities in standard HMMs, the latter being bare memory-based learning machines. In the next sections we deal with a specific instance of the generalization problem that takes particular relevance in ASR, i.e. recognition of speech signal acquired in noisy conditions. The HMM and the ANN/HMM hybrid with/without adaptive amplitudes of activation functions are evaluated over different noisy test sets. Section 6.1 starts our discussion by developing the three cases of the algorithms to learn amplitudes in the ANN/HMM hybrid.

6.1 Derivation of a Gradient-based Algorithm to learn the Amplitudes of the ANN/HMM hybrid

Let us consider a feed-forward network with L layers, namely $\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_L$, where $\mathcal{L}_0$ is the input layer (conventionally it is not counted) and $\mathcal{L}_L$ is the output layer. In the present Chapter we write $w_{i,j,\ell}$ to denote the connection weight that holds between the j-th unit in layer $\mathcal{L}_{\ell-1}$ and the i-th unit in layer $\mathcal{L}_\ell$. Input to the the latter unit at time t is $a_{i,\ell}(t)$, and the corresponding output is $o_{i,\ell}(t)$, such that:

$$a_{i,\ell}(t) = \sum_{j \in \mathcal{L}_{\ell-1}} w_{i,j,\ell} o_{j,\ell-1}(t). \tag{6.1}$$

Sums over $k \in \mathcal{L}_\ell$ are assumed to be extended to all the *indexes* of units of layer $\mathcal{L}_\ell$. An activation function $f_{i,\ell}(.)$ is associated with each unit of the network, defining the corresponding input/output relationship:

$$o_{i,\ell}(t) = f_{i,\ell}(a_{i,\ell}(t)). \tag{6.2}$$

The nature (analytical form) of each activation function may be unit-specific. We assume that $f_{i,\ell}(.)$ can be either in the form

$$f_{i,\ell}(a_{i,\ell}(t)) = \lambda_{i,\ell} \tilde{f}_{i,\ell}(a_{i,\ell}(t)) + \sigma_{i,\ell} \tag{6.3}$$

which is the case of those functions, e.g. sigmoids, that actually have a learnable amplitude $\lambda_{i,\ell}$ and a possible offset (shift) $\sigma_{i,\ell}$ along the ordinate axis ($\sigma_{i,\ell}$ in turn can be a constant - e.g., null - or, more generally, a function of $\lambda_{i,\ell}$); or in the form

$$f_{i,\ell}(a_{i,\ell}(t)) = \tilde{f}_{i,\ell}(a_{i,\ell}(t)) \tag{6.4}$$

that is the case of functions not depending upon any adaptive amplitudes (linear activation, for instance). For notational convenience, in the following the symbol $\tilde{f}_{i,\ell}(a_{i,\ell}(t))$ will refer to the analytical expression of a function of $a_{i,\ell}(t)$ which does not explicitly depend on $\lambda_{i,\ell}$.

In the present scenario, the global criterion function C to be maximized during training is the *likelihood* of the acoustic observations given the model, exactly like in Section 5.2. Formally:

$$C = \sum_{i \in \mathcal{F}} \alpha_{i,T} \tag{6.5}$$

where the meaning of the symbols is inherited from the notation adopted in Chapter 5. Along the directions outlined in Chapter 3 for feedforward and recurrent ANNs, three major Cases may be taken into consideration at this point: (*i*) a unique, adaptive λ is shared among all the units of the ANN for which a trainable amplitude is required; (*ii*) a different value λ_ℓ of the amplitude is defined for each layer $\mathcal{L}_\ell$, and possibly shared among the units belonging to the same layer; (*iii*) different $\lambda_{i,\ell}$ are possibly defined for each unit i in each layer $\mathcal{L}_\ell$ of the network. In all three cases, in the following we develop gradient-ascent training algorithms for the amplitudes of the ANN within the ANN/HMM hybrid framework. These algorithms are then applied in parallel with the hybrid learning rules calculated in the previous Chapter for the connection weights, i.e. at each training epoch both weights and amplitudes update occurs simultaneously. The same learning rate *cannot*, in general, be sued for both weights and amplitudes update.

Case 1, unique amplitude λ

Let us first consider the case of a single adaptive amplitude λ defined over the whole ANN, i.e. $\lambda_{i,\ell} = \lambda$ and $\sigma_{i,\ell} = \sigma$ for all the units of the net for which equation (6.3) holds. Hill climbing gradient-ascent prescribes a learning rule of the kind:

$$\Delta\lambda = \eta \frac{\partial C}{\partial \lambda}. \tag{6.6}$$

where η is the *learning rate*. In the following we develop calculations for $\frac{\partial C}{\partial \lambda}$, such as to obtain a learning rule for the ANN amplitude aimed at maximizing the likelihood of the acoustic observations. According to Chapter 5, namely Equation (5.8) and Theorem 5.1, we can write:

$$\begin{aligned}
\frac{\partial C}{\partial \lambda} &= \sum_i \sum_t \frac{\partial C}{\partial b_{i,t}} \frac{\partial b_{i,t}}{\partial \lambda} \qquad (6.7)\\
&= \sum_i \sum_t \frac{\partial C}{\partial \alpha_{i,t}} \frac{\partial \alpha_{i,t}}{\partial b_{i,t}} \frac{\partial b_{i,t}}{\partial \lambda} \\
&= \sum_i \sum_t \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \frac{\partial b_{i,t}}{\partial \lambda}.
\end{aligned}$$

where the sums are extended over all states $i = 1, \dots, Q$ of the HMM corresponding to the correct transcription of the training utterance under consideration, and to all $t = 1, \dots, T$, respectively.

An explicit formulation for $\frac{\partial b_{i,t}}{\partial \lambda}$ is now required; it has to be compact, computationally feasible and, especially, independent from the specific ANN topology (e.g., from the number of hidden layers). Since in the present ANN/HMM hybrid paradigm we assume that $b_{i,t}$ is the output from i-th unit in layer $\mathcal{L}_L$ of the ANN at time t, we are allowed to write:

$$\frac{\partial b_{i,t}}{\partial \lambda} = \frac{\partial o_{i,L}(t)}{\partial \lambda}. \tag{6.8}$$

Note that two distinct sub-cases may occur at this point:

Case 1.1: $f_{i,L}(a_{i,L}(t)) = \lambda \tilde{f}_{i,L}(a_{i,L}(t)) + \sigma$. Calculating the derivative of $o_{i,L}(t)$ with respect to λ yields:

$$\frac{\partial o_{i,L}(t)}{\partial \lambda} = \tilde{f}_{i,L}(a_{i,L}(t)) + \lambda \frac{\partial \tilde{f}_{i,L}(a_{i,L}(t))}{\partial \lambda} + \frac{\partial \sigma}{\partial \lambda} \tag{6.9}$$

where

$$\begin{aligned} \frac{\partial \tilde{f}_{i,L}(a_{i,L}(t))}{\partial \lambda} &= \frac{\partial \tilde{f}_{i,L}(a_{i,L}(t))}{\partial a_{i,L}(t)} \frac{\partial a_{i,L}(t)}{\partial \lambda} \\ &= \tilde{f}'_{i,L}(a_{i,L}(t)) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}(t)}{\partial \lambda} \end{aligned} \tag{6.10}$$

that can be substituted into equation (6.9), obtaining:

$$\frac{\partial o_{i,L}(t)}{\partial \lambda} = \tilde{f}_{i,L}(a_{i,L}(t)) + \lambda \tilde{f}'_{i,L}(a_{i,L}(t)) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}(t)}{\partial \lambda} + \frac{\partial \sigma}{\partial \lambda}. \tag{6.11}$$

Case 1.2: $f_{i,L}(a_{i,L}(t)) = \tilde{f}_{i,L}(a_{i,L}(t))$ (activation function does not depend on λ). We can write:

$$\begin{aligned} \frac{\partial o_{i,L}(t)}{\partial \lambda} &= \frac{\partial \tilde{f}_{i,L}(a_{i,L}(t))}{\partial a_{i,L}(t)} \frac{\partial a_{i,L}(t)}{\partial \lambda} \\ &= \tilde{f}'_{i,L}(a_{i,L}(t)) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}(t)}{\partial \lambda}. \end{aligned} \tag{6.12}$$

The formulations for $\frac{\partial o_{i,L}(t)}{\partial \lambda}$ in Cases 1.1 and 1.2 are different, but a common expression may be obtained by introducing the following *auxiliary functions*:

$$g_{k,\ell}(a_{k,\ell}(t)) = \begin{cases} \tilde{f}_{k,\ell}(a_{k,\ell}(t)) + \frac{\partial \sigma}{\partial \lambda} & \text{if } f_{k,\ell}(a_{k,\ell}(t)) = \lambda \tilde{f}_{k,\ell}(a_{k,\ell}(t)) + \sigma \\ 0 & \text{if } f_{k,\ell}(a_{k,\ell}(t)) = \tilde{f}_{k,\ell}(a_{k,\ell}(t)) \end{cases} \tag{6.13}$$

and

$$h_{k,\ell}(a_{k,\ell}(t)) = \begin{cases} \lambda \tilde{f}'_{k,\ell}(a_{k,\ell}(t)) & \text{if } f_{k,\ell}(a_{k,\ell}(t)) = \lambda \tilde{f}_{k,\ell}(a_{k,\ell}(t)) + \sigma \\ \tilde{f}'_{k,\ell}(a_{k,\ell}(t)) & \text{if } f_{k,\ell}(a_{k,\ell}(t)) = \tilde{f}_{k,\ell}(a_{k,\ell}(t)). \end{cases} \tag{6.14}$$

It is now possible to rewrite equations (6.11) and (6.12) in a common form:

$$\frac{\partial o_{i,L}(t)}{\partial \lambda} = g_{i,L}(a_{i,L}(t)) + h_{i,L}(a_{i,L}(t)) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}(t)}{\partial \lambda}. \tag{6.15}$$

Finally, let us introduce the definition of *expansion* function $x_{k,\ell}(.)$ of unit k of layer $\mathcal{L}_\ell$, for $\ell = 1, \ldots, L$ as follows:

$$x_{k,\ell}(a_{k,\ell}(t)) = \begin{cases} g_{k,\ell}(a_{k,\ell}(t)) & \text{if } \ell = 1 \\ g_{k,\ell}(a_{k,\ell}(t)) + h_{k,\ell}(a_{k,\ell}(t)) \sum_{n \in \mathcal{L}_{\ell-1}} w_{k,n,\ell} x_{n,\ell-1}(a_{n,\ell-1}(t)) & \text{otherwise.} \end{cases} \tag{6.16}$$

The following theorem can now be easily proved to hold true.

Theorem 6.1: $\frac{\partial o_{k,\ell}(t)}{\partial \lambda} = x_{k,\ell}(a_{k,\ell}(t))$ for each $\ell = 1, \ldots, L$ and for each $k \in \mathcal{L}_\ell$.

Proof: The proof is straightforward (by induction on the total number of layers L) once the equivalent steps in the proof of Theorem 3.1 (Chapter 3) have been followed. $\mathcal{Q.E.D}$.

As a corollary of Theorem 6.1, and given Equation (6.8), equation (6.7) can be restated in the form of an *on-line* learning rule for the amplitude λ:

$$\Delta\lambda = \eta \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} x_{i,L}(a_{i,L}(t)). \tag{6.17}$$

Implementation of rule (6.17) – in terms of the recursive form for the expansion and auxiliary functions given above – is straightforward. Equation (6.17) is general and does not explicitly depend on the peculiar architecture of the ANN, or on the specific analytical form of the activation functions.

The algorithm developed so far deals with a single training sequence Y. In the *batch* case, a training set $\mathcal{Y}$ containing a number of training sequences, i.e. $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_n\}$ is considered. The training criterion becomes $\mathcal{C} = P(\mathcal{Y} \mid \mathcal{M}) = \prod_{i=1}^{n} P(Y_i \mid \mathcal{M})$, namely the overall likelihood of *all* the training sequences $\mathcal{Y}$ given the model $\mathcal{M}$. The amount of change prescribed by gradient ascent for the amplitude λ in the ANN can then be computed, as in Section 5.3.2, as follows:

$$\begin{aligned} \Delta\lambda &= \eta \frac{\partial \mathcal{C}}{\partial \lambda} \qquad (6.18) \\ &= \eta \frac{\partial}{\partial \lambda} \prod_{i=1}^{n} P(Y_i \mid \mathcal{M}) \\ &= \eta \sum_{i=1}^{n} \left\{ \frac{\partial P(Y_i \mid \mathcal{M})}{\partial \lambda} \prod_{j \neq i} P(Y_j \mid \mathcal{M}) \right\} \end{aligned}$$

that implies the calculation of $\frac{\partial P(Y_i \mid \mathcal{M})}{\partial \lambda}$ as in the on-line case, and prescribes to update the amplitude of the ANN according to a linear combination of individual contributes yielded by learning rule (6.17) computed over each sequence of the training set.

Whenever the computation has to be carried out in the *logarithmic domain*, e.g. to tackle the usual numerical stability problems in HMMs, the ANN outputs are interpreted as estimates of the logarithm of the emission probabilities associated with states of the underlying HMM, and we can rewrite Equation (6.7) as:

$$\begin{aligned}
\frac{\partial C}{\partial \lambda} &= \sum_i \sum_t \frac{\partial C}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial \lambda} \qquad (6.19)\\
&= \sum_i \sum_t \frac{\partial C}{\partial \alpha_{i,t}} \frac{\partial \alpha_{i,t}}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial \lambda} \\
&= \sum_i \sum_t \beta_{i,t} \frac{\partial \alpha_{i,t}}{\partial \log \alpha_{i,t}} \frac{\partial \log \alpha_{i,t}}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial \lambda} \\
&= \sum_i \sum_t \beta_{i,t} \frac{\partial e^{\log \alpha_{i,t}}}{\partial \log \alpha_{i,t}} \frac{\partial \log \alpha_{i,t}}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial \lambda} \\
&= \sum_i \sum_t \beta_{i,t} \alpha_{i,t} \frac{\partial \log \alpha_{i,t}}{\partial \log b_{i,t}} \frac{\partial \log b_{i,t}}{\partial \lambda} \\
&= \sum_i \sum_t \beta_{i,t} \alpha_{i,t} \frac{\partial \log b_{i,t}}{\partial \lambda}
\end{aligned}$$

where $\frac{\partial \log b_{i,t}}{\partial \lambda} = \frac{\partial o_{i,L}(t)}{\partial \lambda}$, that can be computed by applying Theorem 6.1, obtaining the following on-line learning rule in the logarithmic domain:

$$\Delta \lambda = \eta \sum_{i=1}^{Q} \sum_{t=1}^{T} \beta_{i,t} \alpha_{i,t} x_{i,L}(a_{i,L}(t)). \qquad (6.20)$$

The above calculations can be combined with Equation 6.18 in order to obtain a *batch* learning rule in the logarithmic domain, whenever required.

Case 2, layer-by-layer amplitudes

Let us now allow different values of the amplitude $\lambda_m, m = 1, \ldots, L$ (and of the corresponding σ_m) on a layer-by-layer basis, i.e. all the units of layer $\mathcal{L}_m$ for which equation (6.3) holds do share a common amplitude λ_m. Again, no other assumptions are made on the ANN topology, or on the analytical form of the (possibly unit-specific) activation functions within a given layer. As in Case 1, the goal is developing an on-line rule to learn λ_m in order to maximize the criterion function C. Using gradient ascent, and carrying out the same calculations as in Equation (6.7) we can write:

$$\frac{\partial C}{\partial \lambda_m} = \sum_i \sum_t \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \frac{\partial b_{i,t}}{\partial \lambda_m}. \qquad (6.21)$$

Again, being $\frac{\partial b_{i,t}}{\partial \lambda_m} = \frac{\partial o_{i,L}(t)}{\partial \lambda_m}$ and according to the calculations accomplished in Case 1 – as well as in Section 3.2.2 (Case 2 of adaptive amplitudes for feedforward ANNs) –

it is possible to define the layer-by-layer version of the *auxiliary functions for $\mathcal{L}_m$-th layer of the ANN* as:

$$g_{k,\ell,m}(a_{k,\ell}(t)) = \begin{cases} \tilde{f}_{k,\ell}(a_{k,\ell}(t)) + \frac{\partial\sigma_\ell}{\partial\lambda_m} & \text{if } f_{k,\ell}(a_{k,\ell}(t)) = \lambda_\ell\tilde{f}_{k,\ell}(a_{k,\ell}(t)) + \sigma_\ell \text{ and } m \geq l \\ 0 & \text{otherwise} \end{cases} \tag{6.22}$$

and

$$h_{k,\ell,m}(a_{k,\ell}(t)) = \begin{cases} 0 & \text{if } \ell = m \\ \lambda_\ell\tilde{f}'_{k,\ell}(a_{k,\ell}(t)) & \text{if } f_{k,\ell}(a_{k,\ell}(t)) = \lambda_\ell\tilde{f}_{k,\ell}(a_{k,\ell}(t)) + \sigma_\ell \text{ and } \ell \neq m \\ \tilde{f}'_{k,\ell}(a_{k,\ell}(t)) & \text{if } f_{k,\ell}(a_{k,\ell}(t)) = \tilde{f}_{k,\ell}(a_{k,\ell}(t)) \text{ and } \ell \neq m. \end{cases} \tag{6.23}$$

that are defined for each unit k of any layer $\mathcal{L}_\ell$, and for time $t = 1,\dots,T$. Such definitions do not depend on the architecture of the network, or on the kind of activation functions.

The m-th *expansion* of unit k of layer $\mathcal{L}_\ell$, for $\ell = 1,\dots,L$, and time $t = 1,\dots,T$ is finally defined in terms of the auxiliary functions as:

$$x_{k,\ell,m}(a_{k,\ell}(t)) = \begin{cases} g_{k,\ell,m}(a_{k,\ell}(t)) & \text{if } \ell = 1 \\ g_{k,\ell,m}(a_{k,\ell}(t)) + h_{k,\ell,m}(a_{k,\ell}(t))\sum_{n\in\mathcal{L}_{\ell-1}} w_{k,n,\ell}x_{n,\ell-1,m}(a_{n,\ell-1}(t)) & \text{otherwise.} \end{cases} \tag{6.24}$$

The following theorem can now be shown to hold true:.

Theorem 6.2: $\frac{\partial o_{k,\ell}(t)}{\partial\lambda_m} = x_{k,\ell,m}(a_{k,\ell}(t))$ for each $\ell = 1,\dots,L$, for each $k \in \mathcal{L}_\ell$, for each $m = 1,\dots,L$ and for each $t = 1,\dots,T$.

Proof: The proof is obtained by induction, in a straightforward manner, by following the scheme used in the proof of Theorems 3.2 (see the Appendix A.1) and 6.1. $\mathcal{Q.E.D.}$

As a corollary of Theorem 6.2, Equation (6.21) can be restated in the form of a layer-by-layer, *on-line* learning rule for the amplitude λ_m:

$$\Delta\lambda_m = \eta\sum_{i=1}^{Q}\sum_{t=1}^{T}\beta_{i,t}\frac{\alpha_{i,t}}{b_{i,t}}x_{i,L,m}(a_{i,L}(t)). \tag{6.25}$$

Taking advantage of the recursive form of the auxiliary and expansion functions, exactly as in rule (6.17), learning rule (6.25) turns out to be readily implementable. *Batch* and *logarithmic* versions of the present algorithm may be obtained as usual, following the scheme outlined at the end of Case 1.

Case 3, unit-by-unit amplitudes

Finally, let us consider the most general Case, where each activation function for a generic unit ι in any layer $\mathcal{L}_\ell$ may be either in form (6.3) (i.e. depending upon its own amplitude $\lambda_{\iota,\ell}$ and offset $\sigma_{\iota,\ell}$) or in the form (6.4). Given again the criterion C, an on-line rule to learn $\lambda_{\iota,\ell}$ that maximizes C is sought. Relying on gradient ascent, and repeating the calculations outlined in Equation (6.7) we have:

$$\frac{\partial C}{\partial \lambda_{\iota,\ell}} = \sum_i \sum_t \beta_{i,t} \frac{\alpha_{i,t}}{b_{i,t}} \frac{\partial b_{i,t}}{\partial \lambda_{\iota,\ell}}. \tag{6.26}$$

Again, being $\frac{\partial b_{i,t}}{\partial \lambda_{\iota,\ell}} = \frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,\ell}}$ and according to the calculations accomplished in Section 3.2.3 (Case 3 of adaptive amplitudes for feedforward ANNs) it is possible to develop the unit-by-unit version of the on-line learning rule, defined for each $\ell = 1, \ldots, L$ and for each $\iota \in \mathcal{L}_\ell$ for which equation (6.3) holds[1]. The calculations that follow do implicitly rely on the assumption that the activation function under consideration is in the form $f_{\iota,\ell}(a_{\iota,\ell}(t)) = \lambda_{\iota,\ell}\tilde{f}_{\iota,\ell}(a_{\iota,\ell}(t)) + \sigma_{\iota,\ell}$, since otherwise no adaptive amplitude would be required. The derivation of the present learning rule turns out to be much closer to the BML algorithm for the ANN/HMM hybrid (introduced in Chapter 5) than in the previous Cases 1 and 2, as it will be seen shortly. It actually implies the "backpropagation" of *deltas* from the ANN topmost layer down to lower layers.

First of all, let ι be an output unit, i.e., $\ell = L$. If $\iota \neq i$ then $\frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,L}} = 0$. Otherwise, we can write:

$$\begin{aligned} \frac{\partial o_{\iota,L}(t)}{\partial \lambda_{\iota,L}} &= \tilde{f}_{\iota,L}(a_{\iota,L}(t)) + \lambda_{\iota,L} \frac{\partial \tilde{f}'_{\iota,L}(a_{\iota,L}(t))}{\partial \lambda_{\iota,L}} + \frac{\partial \sigma_{\iota,L}}{\partial \lambda_{\iota,L}} \\ &= \tilde{f}_{\iota,L}(a_{\iota,L}(t)) + \frac{\partial \sigma_{\iota,L}}{\partial \lambda_{\iota,L}}. \end{aligned} \tag{6.27}$$

Defining the quantity:

$$\delta_{j,L}(i,t) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \tag{6.28}$$

we can summarize the above Equations in the form:

$$\frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,L}} = \delta_{\iota,L}(i,t) \left\{ \tilde{f}_{\iota,L}(a_{\iota,L}(t)) + \frac{\partial \sigma_{\iota,L}}{\partial \lambda_{\iota,L}} \right\}. \tag{6.29}$$

Let us now consider the case of units in the first hidden layer, that is to say $\ell = L-1$. Taking partial derivative of $o_{i,L}(t)$ with respect to $\lambda_{\iota,L-1}$ and repeatedly applying the *chain rule* yield:

$$\begin{aligned} \frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,L-1}} &= \frac{\partial o_{i,L}(t)}{\partial o_{\iota,L-1}(t)} \frac{\partial o_{\iota,L-1}(t)}{\partial \lambda_{\iota,L-1}} \\ &= \frac{\partial o_{i,L}(t)}{\partial o_{\iota,L-1}(t)} \left\{ \tilde{f}_{\iota,L-1}(a_{\iota,L-1}(t)) + \frac{\partial \sigma_{\iota,L-1}}{\partial \lambda_{\iota,L-1}} \right\} \end{aligned} \tag{6.30}$$

[1]A different instance of the rule, possibly with different learning rates $\eta_{\iota,\ell}$, is to be used for each unit.

where the same passages as those used in equation (6.27) have been applied to expand $\frac{\partial o_{\iota,L-1}(t)}{\partial \lambda_{\iota,L-1}}$. The calculation of $\frac{\partial o_{i,L}(t)}{\partial o_{\iota,L-1}(t)}$ can be accomplished by writing:

$$\begin{aligned} \frac{\partial o_{i,L}(t)}{\partial o_{\iota,L-1}(t)} &= \frac{\partial o_{i,L}(t)}{\partial a_{i,L}(t)} \frac{\partial a_{i,L}(t)}{\partial o_{\iota,L-1}(t)} \\ &= f'_{i,L}(a_{i,L}(t)) w_{i,\iota,L}. \end{aligned} \tag{6.31}$$

Substituting equation (6.31) into equation (6.30) we have:

$$\begin{aligned} \frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,L-1}} &= w_{i,\iota,L} f'_{i,L}(a_{i,L}(t)) \left\{ \tilde{f}_{\iota,L-1}(a_{\iota,L-1}(t)) + \frac{\partial \sigma_{\iota,L-1}}{\partial \lambda_{\iota,L-1}} \right\} \\ &= \left\{ \sum_{k \in \mathcal{L}_L} w_{k,\iota,L} \delta_{k,L}(i,t) f'_{k,L}(a_{k,L}(t)) \right\} \left\{ \tilde{f}_{\iota,L-1}(a_{\iota,L-1}(t)) + \frac{\partial \sigma_{\iota,L-1}}{\partial \lambda_{\iota,L-1}} \right\} \\ &= \delta_{\iota,L-1}(i,t) \left\{ \tilde{f}_{\iota,L-1}(a_{\iota,L-1}(t)) + \frac{\partial \sigma_{\iota,L-1}}{\partial \lambda_{\iota,L-1}} \right\} \end{aligned} \tag{6.32}$$

where we defined

$$\delta_{\iota,L-1}(i,t) = \sum_{k \in \mathcal{L}_L} w_{k,\iota,L} \delta_{k,L}(i,t) f'_{k,L}(a_{k,L}(t)). \tag{6.33}$$

Next, consider the case of units in the following hidden layer, i.e. $\ell = L - 2$. Taking partial derivative of $o_{i,L}(t)$ with respect to $\lambda_{\iota,L-2}$ and repeatedly using the *chain rule*, as in Equation (6.30), yield:

$$\frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,L-2}} = \frac{\partial o_{i,L}(t)}{\partial o_{\iota,L-2}(t)} \left\{ \tilde{f}_{\iota,L-2}(a_{\iota,L-2}(t)) + \frac{\partial \sigma_{\iota,L-2}}{\partial \lambda_{\iota,L-2}} \right\}. \tag{6.34}$$

The calculation of $\frac{\partial o_{i,L}(t)}{\partial o_{\iota,L-2}(t)}$ can be accomplished by writing:

$$\begin{aligned} \frac{\partial o_{i,L}(t)}{\partial o_{\iota,L-2}(t)} &= \frac{\partial o_{i,L}(t)}{\partial a_{i,L}(t)} \frac{\partial a_{i,L}(t)}{\partial o_{\iota,L-2}(t)} \\ &= f'_{i,L}(a_{i,L}(t)) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}(t)}{\partial o_{\iota,L-2}(t)} \\ &= f'_{i,L}(a_{i,L}(t)) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} \frac{\partial o_{j,L-1}(t)}{\partial a_{j,L-1}(t)} \frac{\partial a_{j,L-1}(t)}{\partial o_{\iota,L-2}(t)} \\ &= f'_{i,L}(a_{i,L}(t)) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} f'_{j,L-1}(a_{j,L-1}(t)) \frac{\partial a_{j,L-1}(t)}{\partial o_{\iota,L-2}(t)} \\ &= f'_{i,L}(a_{i,L}(t)) \sum_{j \in \mathcal{L}_{L-1}} w_{i,j,L} f'_{j,L-1}(a_{j,L-1}(t)) \sum_{k \in \mathcal{L}_{L-2}} w_{j,k,L-1} \frac{\partial o_{k,L-2}(t)}{\partial o_{\iota,L-2}(t)} \end{aligned} \tag{6.35}$$

$$
\begin{aligned}
&= f'_{i,L}(a_{i,L}(t)) \sum_{j\in\mathcal{L}_{L-1}} w_{i,j,L} f'_{j,L-1}(a_{j,L-1}(t)) w_{j,\iota,L-1} \\
&= \sum_{j\in\mathcal{L}_{L-1}} \{w_{i,j,L} f'_{i,L}(a_{i,L}(t))\} f'_{j,L-1}(a_{j,L-1}(t)) w_{j,\iota,L-1} \\
&= \sum_{j\in\mathcal{L}_{L-1}} \{\sum_{k\in\mathcal{L}_L} w_{k,j,L}\delta_{k,L}(i,t) f'_{k,L}(a_{k,L}(t))\} f'_{j,L-1}(a_{j,L-1}(t)) w_{j,\iota,L-1} \\
&= \sum_{j\in\mathcal{L}_{L-1}} w_{j,\iota,L-1}\delta_{j,L-1}(i,t) f'_{j,L-1}(a_{j,L-1}(t)).
\end{aligned}
$$

Substituting equation (6.35) into equation (6.34) we obtain:

$$
\begin{aligned}
\frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,L-2}} &= \left\{\sum_{j\in\mathcal{L}_{L-1}} w_{j,\iota,L-1}\delta_{j,L-1}(i,t) f'_{j,L-1}(a_{j,L-1}(t))\right\}\left\{\tilde{f}_{\iota,L-2}(a_{\iota,L-2}(t)) + \frac{\partial \sigma_{\iota,L-2}}{\partial \lambda_{\iota,L-2}}\right\} \\
&= \delta_{\iota,L-2}(i,t)\left\{\tilde{f}_{\iota,L-2}(a_{\iota,L-2}(t)) + \frac{\partial \sigma_{\iota,L-2}}{\partial \lambda_{\iota,L-2}}\right\} \qquad (6.36)
\end{aligned}
$$

where we defined:

$$\delta_{\iota,L-2}(i,t) = \sum_{k\in\mathcal{L}_{L-1}} w_{k,\iota,L-1}\delta_{k,L-1}(i,t) f'_{k,L-1}(a_{k,L-1}(t)) \qquad (6.37)$$

for each $\iota \in \mathcal{L}_{L-2}$. Finally, for subsequent layers, i.e. $\ell < L-2$ we can still write:

$$\frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,\ell}} = \frac{\partial o_{i,L}(t)}{\partial o_{\iota,\ell}(t)}\left\{\tilde{f}_{\iota,\ell}(a_{\iota,\ell}(t)) + \frac{\partial \sigma_{\iota,\ell}}{\partial \lambda_{\iota,\ell}}\right\} \qquad (6.38)$$

as in equation (6.30). Let us now assume that, for $m = \ell+1,\ldots,L-2$ and $h \in \mathcal{L}_m$, the following equation holds:

$$\frac{\partial o_{i,L}(t)}{\partial o_{h,m}(t)} = \delta_{h,m}(i,t) \qquad (6.39)$$

(the calculations accomplished so far already proved that this assumption is true for $m = L, L-1, L-2$, yielding the basis for the following induction), where $\delta_{k,\ell}(i,t)$ is recursively defined, generalizing definition (6.37), as:

$$\delta_{k,\ell}(i,t) = \sum_{j\in\mathcal{L}_{\ell+1}} w_{j,k,\ell+1}\delta_{j,\ell+1}(i,t) f'_{j,\ell+1}(a_{j,\ell+1}(t)). \qquad (6.40)$$

for each $\ell = 1,\ldots,L-1$ and for each $k \in \mathcal{L}_\ell$. With this hypothesis, it is possible to write:

$$\frac{\partial o_{i,L}(t)}{\partial o_{\iota,\ell}(t)} = \sum_{j\in\mathcal{L}_{\ell+1}} \frac{\partial o_{i,L}(t)}{\partial o_{j,\ell+1}(t)}\frac{\partial o_{j,\ell+1}(t)}{\partial o_{\iota,\ell}(t)} \qquad (6.41)$$

$$
\begin{aligned}
&= \sum_{j\in\mathcal{L}_{\ell+1}} \delta_{j,\ell+1}(i,t)\frac{\partial o_{j,\ell+1}(t)}{\partial o_{\iota,\ell}(t)} \\
&= \sum_{j\in\mathcal{L}_{\ell+1}} \delta_{j,\ell+1}(i,t)\frac{\partial o_{j,\ell+1}(t)}{\partial a_{j,\ell+1}(t)}\frac{\partial a_{j,\ell+1}(t)}{\partial o_{\iota,\ell}(t)} \\
&= \sum_{j\in\mathcal{L}_{\ell+1}} \delta_{j,\ell+1}(i,t) f'_{j,\ell+1}(a_{j,\ell+1}(t)) w_{j,\iota,\ell+1} \\
&= \delta_{\iota,\ell}(i,t)
\end{aligned}
$$

where we used the inductive hypothesis and definition (6.40) for $\delta_{\iota,\ell}(i,t)$. Substituting equation (6.41) into equation (6.38) we obtain:

$$
\frac{\partial o_{i,L}(t)}{\partial \lambda_{\iota,\ell}} = \delta_{\iota,\ell}(i,t)\left\{\tilde{f}_{\iota,\ell}(a_{\iota,\ell}(t)) + \frac{\partial \sigma_{\iota,\ell}}{\partial \lambda_{\iota,\ell}}\right\}. \tag{6.42}
$$

Considering equation (6.42), it immediately follows (by induction) that since assumption (6.39) holds for $m = L, L-1, L-2$, it also holds for all possible subsequent layers $L-3, \ldots, 1$.

In summary, Equations (6.27), (6.35) (6.36) and (6.42) can be substituted into Equation (6.26), obtaining an on-line learning rule in the common form:

$$
\Delta\lambda_{\iota,\ell} = \eta \sum_{i=1}^{Q}\sum_{t=1}^{T} \beta_{i,t}\frac{\alpha_{i,t}}{b_{i,t}}\delta_{\iota,\ell}(i,t)\left\{\tilde{f}_{\iota,\ell}(a_{\iota,\ell}(t)) + \frac{\partial \sigma_{\iota,\ell}}{\partial \lambda_{\iota,\ell}}\right\} \tag{6.43}
$$

for each layer $\ell = \mathcal{L}_1, \ldots, \mathcal{L}_L$ in the ANN, and for each unit ι in $\mathcal{L}_\ell$. According to Equations (6.28), (6.33), (6.37) and (6.40), the quantity $\delta_{\iota,\ell}(i,t)$ can be computed as:

$$
\delta_{\iota,\ell}(i,t) = \begin{cases} 1 & \text{if } \ell = L \text{ and } \iota = i \\ 0 & \text{if } \ell = L \text{ and } \iota \neq i \\ \sum_{j\in\mathcal{L}_{\ell+1}} w_{j,\iota,\ell+1}\delta_{j,\ell+1}(i,t) f'_{j,\ell+1}(a_{j,\ell+1}(t)) & \text{otherwise.} \end{cases} \tag{6.44}
$$

Batch as well as *logarithmic* versions of the present training scheme may be obtained, if desired, as in Cases 1 and 2, i.e. following the corresponding guidelines outlined in Chapter 5 for the BML training of the ANN/HMM hybrid.

6.2 From Maximum Likelihood to Maximum A Posteriori

The above algorithms (Cases 1, 2 and 3) are aimed at the maximization of the bare likelihood (BML) of acoustic observation sequences given the model. As pointed out in Chapter 5, the BML suffers from what we called the "divergence problem", i.e. the ANN may simply learn to yield higher and higher output values over all inputs, in order to increase the overall likelihood, whenever no explicit constraints are imposed on the ANN architecture or on the training algorithm. Several viable solutions were outlined, namely: (a) imposing probabilistic constraints on the free parameters (weights) of the model; (b) using Bayes' theorem to shift the nature of ANN outputs from (unconstrained) emission likelihoods to (constrained) state-posterior probabilities; and, above all, (c) using a discriminative training criterion, namely *Maximum A Posteriori* (MAP) (Bengio, 1996) instead of BML. The latter approach resulted particularly effective throughout the experiments, yielding the best performance. Other advantages of the MAP were discussed in Section 5.4.4, along the guidelines of (Bourlard and Morgan, 1994a). In the present scenario, the divergence problem may even turn out to be more severe, since the amplitudes of output units are encouraged by gradient-ascent to grow indefinitely, causing the model to degenerate.

For these reasons, a MAP version of the algorithms developed in Section 6.1 is described herein. Novel learning rules are obtained, relying on the calculations developed for BML. Applying Bayes' theorem, the MAP criterion $\mathcal{C}_{MAP}$ was formally defined as $\mathcal{C}_{MAP} = P(\mathcal{M} \mid Y) = \frac{P(Y|\mathcal{M})P(\mathcal{M})}{P(Y)}$ where Y is the observation sequence under consideration, with corresponding likelihood $P(Y)$, and $\mathcal{M}$ denotes the ANN/HMM model (as in Section 5.3.2), having *prior* probability $P(\mathcal{M})$. As observed in Section 5.4.4, $P(\mathcal{M})$ does not depend on the ANN, being given by the language model. As a consequence, it can be computed separately. As a matter of fact, a suitable *on-line* learning algorithm is obtained by maximizing the equivalent criterion:

$$\mathcal{C}'_{MAP} = \frac{P(Y \mid \mathcal{M})}{P(Y)}. \tag{6.45}$$

A gradient-ascent MAP learning rule for a generic trainable amplitude λ is obtained as follows:

$$\begin{aligned}
\Delta\lambda &= \eta \frac{\partial \mathcal{C}'_{MAP}}{\partial \lambda} \\
&= \eta \frac{\partial}{\partial \lambda} \left\{ \frac{P(Y \mid \mathcal{M})}{P(Y)} \right\} \\
&= \frac{\eta}{[P(Y)]^2} \left\{ P(Y) \frac{\partial P(Y \mid \mathcal{M})}{\partial \lambda} - P(Y \mid \mathcal{M}) \frac{\partial P(Y)}{\partial \lambda} \right\} \\
&= \frac{\eta}{P(Y)} \left\{ \frac{\partial P(Y \mid \mathcal{M})}{\partial \lambda} - \frac{P(Y \mid \mathcal{M})}{P(Y)} \frac{\partial P(Y)}{\partial \lambda} \right\}.
\end{aligned} \tag{6.46}$$

The quantities $P(Y \mid \mathcal{M})$ and $\frac{\partial P(Y|\mathcal{M})}{\partial \lambda}$ are computed according to the calculations introduced in Section 6.1, being $P(Y \mid \mathcal{M})$ the usual likelihood of the observations given the model $\mathcal{M}$. Here $\mathcal{M}$ is the "correct" model, i.e. the model *constrained* by the known transcription of acoustic observation sequence Y, as explained in

(Bengio, 1993; Trentin et al., 1998) and in Section 5.4.4. The quantities $P(Y)$ and $\frac{\partial P(Y)}{\partial \lambda}$ are computed in a similar manner, applying the same calculations to the *unconstrained* (or *recognition*) model (see again (Bengio, 1993; Trentin et al., 1998) and Section 5.4.4). The latter is the same model used for *recognition* with the Viterbi algorithm, where no prior knowledge on the exact transcription of the observations under consideration is given.

Three distinct instances of Equation (6.46) are obtained according to the fact that: (i) a single adaptive amplitude λ is considered (Case 1), i.e. learning rule (6.17) is applied within Equation (6.46); (ii) layer-by-layer amplitudes λ_ℓ are used (Case 2), relying on learning rule (6.25); or, (iii) unit-by-unit trainable amplitudes $\lambda_{m,\ell}$ are taken into consideration (Case 3), relying on learning rule (6.43).

In the *batch* case, n observation sequences $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_n\}$ are given, and the C_{MAP} criterion over $\mathcal{Y}$ has to be maximized. A learning rule similar to equation (6.46) is obtained:

$$\Delta\lambda = \frac{\eta}{[P(\mathcal{Y})]^2} \left\{ P(\mathcal{Y}) \frac{\partial P(\mathcal{Y} \mid \mathcal{M})}{\partial \lambda} - P(\mathcal{Y} \mid \mathcal{M}) \frac{\partial P(\mathcal{Y})}{\partial \lambda} \right\} \tag{6.47}$$

where: the overall likelihood $P(\mathcal{Y})$ is calculated as the product of the likelihoods $P(Y_i)$, $i = 1, \ldots, n$ of the individual observation sequences; similarly, $P(\mathcal{Y} \mid \mathcal{M})$ is calculated as the product of the quantities $P(Y_i \mid \mathcal{M})$, $i = 1, \ldots, n$; $\frac{\partial P(\mathcal{Y}|\mathcal{M})}{\partial \lambda}$ is computed as outlined in the previous Section for the batch case, taking care of using the correct (constrained) model $\mathcal{M}_i$ for each $Y_1, \ldots, Y_n$ (i.e., $\mathcal{M}$ denotes here the *set* of correct models for the different sequences); finally, $\frac{\partial P(\mathcal{Y})}{\partial \lambda}$ is computed applying the above formulas for the batch case to the unconstrained model.

6.3 Experimental Results

Two experimental setups are considered in the following. Both of them rely on a training step accomplished on the same dataset described and used in Chapter 5, namely the *cdigits* part of the *SPK* database, a corpus of clean speech signals collected under laboratory conditions (refer to Section 5.5 for details). This training set includes 500 utterances of Italian connected 8-digit strings (4000 words in total), collected among 20 speakers (10 male and 10 female) and sampled at a rate of 16kHz. Again, the feature space was defined by using 20ms Hamming windows, with an overlap of 10ms, and applying standard spectral analysis to obtain 8 *Mel Frequency Scaled Cepstral Coefficients* (MFSCCs) and the log-energy of the signal. Normalization of such feature space was accomplished to ensure ANNs input values uniformly distributed over the $[0, 1]$ interval.

The same topologies used in Chapter 5 are inherited, namely 11 left-to-right word models (one per digit, plus one for the background noise, or "silence" model), with a number of states equal to the length of the phonetic transcription of each Italian digit according to the SAMPA acoustic-phonetic units. The HMM contains

8 Gaussian pdfs per state, and the usual Segmental k-Means initialization, Baum-Welch training and Viterbi decoding algorithms are used. The ANN to be applied within the hybrid framework was chosen according to the best results obtained in the previous chapter: it is a 2-layer MLP with a 93-sigmoids hidden layer and a 40-sigmoids output layer, initialized according to the Bourlard and Morgan-like iterative BP/Viterbi scheme, and trained with the on-line version of the novel (MAP) hybrid algorithms in the linear domain.

During the initialization via BP, the algorithms presented in Chapter 3 are applied to bootstrap the amplitudes (possibly starting from their random initial values). The MAP procedure turned out to be necessary to tackle the "divergene problem" that arose while attempting to apply the BML algorithms during the experiments. Training of connection weights is accomplished in parallel with re-estimation of transition probabilities in the underlying HMM (via Baum-Welch), as well as with training of the amplitude(s) $\lambda_{\ell,i}$ - whenever required - relying on the algorithms developed in the previous section. In Case 1 the (unique) λ is initialized to 1.0, i.e. the usual standard sigmoid. In Cases 2 and 3, according to the conclusions drawn in Chapter 3, initialization of individual amplitudes is accomplished at random in a small uniform interval centered in 1.0: breaking initial ties, like in standard BP, actually enforces the convergence of the algorithms.

The distinction between the two setups considered herein concerns the different *test* sets. In the first instance (Section 6.3.1), noise (acquired in an office environment) is introduced in the *clean* test speech signals of the test part of *cdigits* assuming a simple additive model. This allows for an evaluation of the acoustic models at different Signal to Noise Ratios (SNRs), preserving the same conditions (e.g., the transducer) that underly the acquisition of the speech signals in the training set, and providing us with hints on the most promising training scheme to be selected for application in a real-world task.

In the second instance (Section 6.3.2), the models trained over the same clean training set are applied to the recognition task of connected digits strings acquired in a real car environment. The dataset used for these experiments, called *VODIS II/SpeechDatCar*, is described at the beginning of Section 6.3.2.

It is worth underlining that the *same* models (HMM and ANN/HMM hybrids) trained over the clean training set are applied throughout all the following recognition experiments. No further training, re-training or adaptation of model parameters is accomplished. Our goal is the evaluation and comparison of the robustness of the acoustic models themselves. Since the number of free parameters in the different models belongs to the same *magnitudo*, the comparison in terms of generalization (bias vs. variance) is fair, from a learning theory point of view. In order to develop effective, real-world ASR systems relying on such models, several techniques are then available to improve recognition in presence of noise; they may be used jointly with any given acoustic model. Examples are represented by noise reduction via spectral subtraction (Mokbel et al., 1998), blind source separation (Bellini and Rocca, 1988; Bell and Sejnowski, 1995), parameter adaptation via input feature *normalization* (relying on linear transformations (Feng, 1988) or connectionist acoustic front-ends (Trentin et al., 1996b; Furlanello et al., 1997; Trentin and Giuliani, 1997; Trentin and Giuliani, 2001)), etc.

6.3.1 Experiments with the additive-noise model

The portion pf the *SPK* corpus used herein is the test part of *cdigits*, the same used for testing purposes in Chapter 5. It consists of other 500 clean utterances (4000 words in total) from 20 speakers (11 male and 9 female) not involved in the training process. Real noise collected in an office environment, and mostly due to the presence of noisy workstations, was added to the clean speech signal, assuming an additive model for the overall signal $s(t)$ at time t in the form:

$$s(t) = x(t) + n(t) \tag{6.48}$$

where $x(t)$ is the clean signal and $n(t)$ is the additive noise. The amount of noise with respect to the clean signal was balanced according to a variety of choices for the SNR, throughout a number of experiments. The technique described in (Giuliani et al., 1999) was applied in order to compute such additive noise simulation.

Table 6.1 shows the results obtained with the standard HMM, Bourlard and Morgan's hybrid, the novel ANN/HMM hybrid (without adaptive amplitudes), and the three cases of the latter featuring adaptive amplitudes (unique, layer-by-layer and unit-by-unit, respectively) over the clean test set and over the noisy test set (at a reference SNR of 20dB). As in Chapter 5, results are expressed in terms of *String Recognition Rate* (SRR) and unit accuracy, i.e. *Word Recognition Rate* (WRR). As usual, language models were implicitly defined in order to equal the overall number of word insertions and deletions within text strings in the test set, prior to the application of such evaluation criteria.

Before drawing any conclusions from the present experiments, it is necessary to underline that the nature of the problem under consideration hardly offers space to substantial improvement over the *clean* signals. In fact, the bare ANN/HMM hybrid trained with MAP forms a strong baseline, as can be seen from a direct comparison with the standard HMM.

Results on the clean speech signals are the same reported in Chapter 5. The standard CDHMM drops its performance when fed with noisy signals, as expected from a memory-based model. Bourlard and Morgan's hybrid, that basically performs as the HMM over the clean data, turns out to be much more noise-tolerant, due to the generalization capabilities of the MLP. In the present experiments, Bourlard and Morgan's paradigm, uses the same ANN topology as that of the novel hybrid, starting from the same initial segmentation of training data (as explained in Chapter 5) labeled with 0/1 target values. The variant "Case 4" of the algorithm, described in (Bourlard and Morgan, 1994a) on page 164, was used. Normalization of net outputs via division by the corresponding state-priors was accomplished, as recommended in

	Clean Signals		*Noisy Signals*	
Acoustic Model	*SRR (%)*	*WRR (%)*	*SRR (%)*	*WRR (%)*
Standard HMM	46.60	90.03	10.20	**69.28**
Bourlard and Morgan's model	47.40	90.20	24.40	79.38
Novel ANN/HMM hybrid (fixed amplitudes)	68.60	94.65	41.40	**86.80**
Hybrid with single adaptive λ	68.40	94.67	42.00	87.33
Hybrid with layer-by-layer adaptive λ_ℓ's	68.20	94.62	44.00	88.33
Hybrid with neuron-by-neuron adaptive $\lambda_{\ell,i}$'s	68.40	94.83	47.00	**89.12**

Table 6.1: *String Recognition Rate (SRR) and Word Recognition Rate (WRR) yielded by the different acoustic models over the clean and noisy test signals of the* SPK *corpus. Noisy signals obtained via an additive noise model (noise collected in an office environment) at a SNR of 20dB.*

(Bourlard and Morgan, 1994a), Chapter 7, in order to reduce from state-posterior probabilities to "likelihoods" during the Viterbi step.

Not only the novel ANN/HMM hybrid (with fixed amplitudes) provides us with a significant gain in terms of WRR on the clean dataset, as discussed in Chapter 5, but it proves itself much more noise-tolerant than the standard models when applied over the noisy signals. This is a first, important hint in the evaluation of robustness of the novel paradigm.

At this point, let us analyze the results obtained from the introduction of trainable amplitudes within the hybrid. In the clean speech recognition task it is seen that Cases 1 through 3 allow for a progressively increasing, but limited, gain over the bare ANN/HMM. Given the nature of the problem under consideration and the performance of the HMM (that is, the state-of-the-art acoustic model in the speech community), this fact could suggest that the different variants of the hybrid are all very close to the global maximum of the criterion function for the present architecture (or, at least, that they converged to highly similar solutions nearby a common local maximum in the parameter space). Recognition performance on the noisy speech signals points out that this is not the case, at least as far as concerns Cases 2 and 3.

In Chapter 3 the application of Case 1 allowed for a significant improvement over conventional ANN training, sometimes even more evident than in Case 3. In the present setup, on the contrary, while Cases 2 and 3 actually improved over the bare MAP training scheme, the former Case yielded only a limited gain in recognition performance. We argue that such a situation has an explanation in terms of the particular architecture at hand, rather than in the training procedure itself. First of all, the presence of standard, normalized sigmoids in both hidden and output layers (with a probabilistic interpretation of the outputs, that are required to estimate state-specific likelihoods for the HMM) may be hardly expected to allow for the learning of a single "improved" amplitude, to be shared among *all* the units of the net. But, above all, a rationale behind the behavior of Case 1 may be searched for in the weight-grouping perspective. In Chapter 3 we observed that the particular groupings induced by the adaptive amplitudes were consistent with those groups that David MacKay (MacKay, 1992a; MacKay, 1992b; MacKay, 1994) discovered to have high *Bayesian evidence*. Such observation holds true for the specific architectures considered therein, but it does not apply to the present ANN topology. As a matter of fact, given a 2-layer MLP with sigmoidal hidden units and linear outputs, Case 1 implicitly defines two weight-groups (input-to-hidden and hidden-to-output weights) which are supported in terms of Bayesian evidence within MacKay's framework. On the contrary, in the present scenario (2-layer MLP with sigmoids in both hidden and output layers) a single group is induced, that substantially is "no group", i.e. a grouping that was shown to have low Bayesian evidence - exactly the same as in bare BP with a unique learning rate. The latter fact forms the basis for the understanding of why Case 1 converged to a solution which is similar to that obtained through the ANN/HMM hybrid with fixed amplitudes. This statemennt is further confirmed by the results presented in the next section.

Nonetheless, Case 1 may still be taken into consideration within the present setup in light of the following considerations: (a) it improves recognition performance in presence of additive noise, although to a limited amount only; (b) it is definitely easy to be implemented, once the computer code for the hybrid training procedure is available; (c) it is not computationally expensive; (d) according to the experimental investigations carried out in Chapter 3, it is likely to perform better whenever applied to different architectures, e.g. using linear output units. In the latter respect, let us point out that in the experiments accomplished in Chapter 5 a significant alternative was represented by a 2-layer net with hidden sigmoids and linear outputs, trained to estimate the *logarithms* of the emission probabilities, and initialized in a supervised manner relying on the parameters of the corresponding HMM to define the target outputs.

Case 2 and Case 3 still induce weight grouping with high Bayesian evidence (see Chapter 3), and they actually realize acoustic models that turn out to be robust to noise. It is interesting to note that, although they allow for a limited improvement on the clean data (seemingly as if they led to similar solutions, as in Case 1), they generalize much better on the noisy signals. This fact, along with the corresponding values of the log-likelihood ratios of the training acoustic observations given the models[2] (that is, the MAP training criterion itself) reported in Table 6.2, shows that *different* maxima of the criterion function are reached, indeed. Each such maximum represents a different solution in the weight space, that may lead to improved generalization capabilities. For instance, Case 3 converges to a solution that has a lower log-likelihood ratio than Case 2, but that generalizes better. At the same time, both solutions correspond to higher log-likelihood ratios than that reached by the hybrid with fixed amplitudes, but without overfitting the training

[2]The concept of likelihood ratio between the *constrained* and *unconstrained* models was introduced in Chapter 5

data (since both of them do generalize better on the test sets).

Hybrid training scheme	*Log-likelihood ratio*
Fixed amplitudes	-7.1541
Single adaptive λ (Case 1)	-7.4058
Layer-by-layer adaptive λ_ℓ's (Case 2)	-6.8395
Neuron-by-neuron adaptive $\lambda_{\ell,i}$'s (Case 3)	-6.9694

Table 6.2: *Log-likelihood ratios between the constrained and the unconstrained models, evaluated over the acoustic observations of the training set, at the end of training for each of the proposed algorithms.*

In particular, Case 3 yields a 64.58% relative WRR reduction with respect to the standard HMM, and a 17.58% relative reduction over the bare ANN/HMM hybrid without trainable amplitudes. Figure 6.1 plots the WRR curves for the HMM, the ANN/HMM hybrid and the hybrid with unit-by-unit adaptive amplitudes (the most significant case) as the amount of noise in the signal increases.

It is seen that: (i) the HMM rapidly worsens its performance in the presence of noise; at a SNR of 40dB (rather light noise) it scores a 87.17% WRR, i.e. about 3% absolute WRR loss with respect to its performance over the clean signal; (ii) the novel ANN/HMM hybrid shows a much more stable and noise-tolerant behavior: at a SNR of 30dB it is still well over the performance of the HMM evaluated on the clean signals, at 20dB the absolute gap in WRR between the two models is 17.32 points; (iii) the hybrid with adaptive amplitudes (Case 3) is constantly higher than the bare ANN/HMM, and the delta becomes significant when the noise level grows over a SNR of 30dB; at the latter SNR level its performance is comparable with that yielded by the bare hybrid evaluated on the clean data, while up to nearly 20dB it compares favorably with the performance of the standard HMM on the clean signals. Cases 1 and 2, not shown in the figure, lie in the middle between the bare hybrid and Case 3: Case 1 is very close to the ANN/HMM with fixed amplitudes. Instead, Case 2 turns out to be closer to Case 3 (although constantly worse), proving that the latter two Cases reached different maxima in the criterion function that allow for improved generalization capabilities of the resulting models.

6.3.2 ASR in a real car environment

Although in the mid Nineties much interest in the speech community concerned large vocabulary ASR under clean conditions, recently a significant deal of attention has been cathalyzed by small-vocabulary ASR tasks under severe noisy conditions. A reduced set of connected or isolated words (e.g., vocal commands for automatic devices, or digits to perform hands-free telephone calls within a car) is often required.

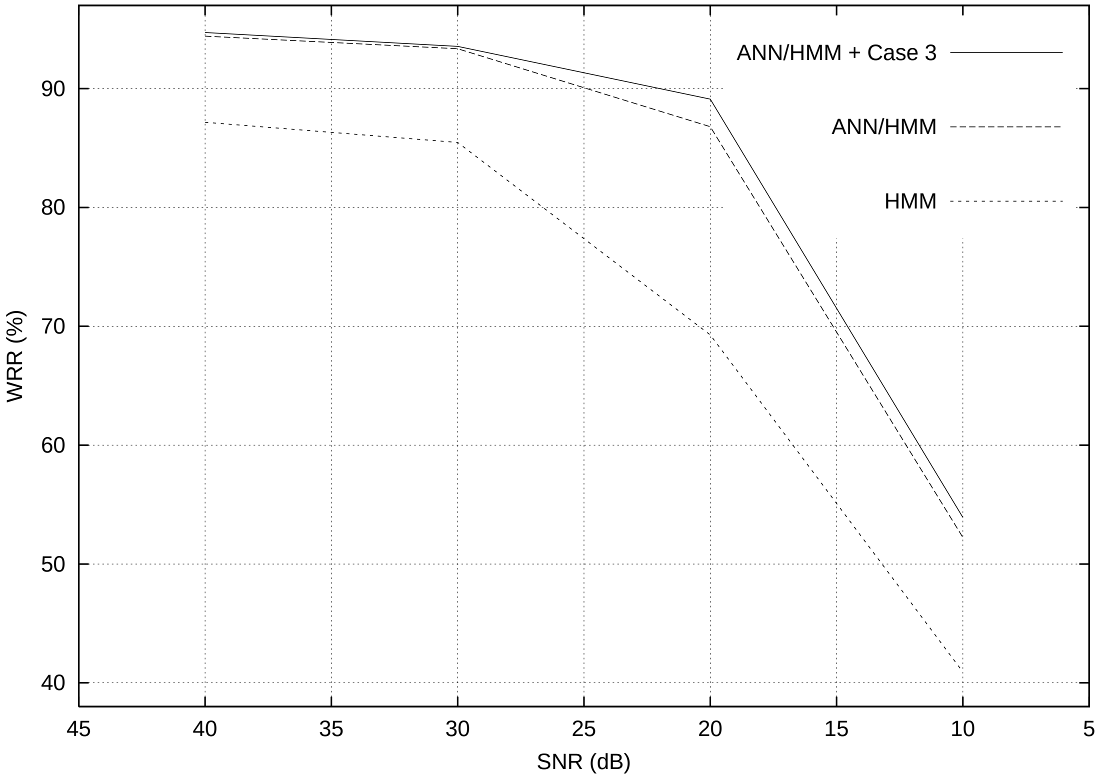

Figure 6.1: *WRR curves as functions of the SNR in the three major cases: standard HMM (lower curve), novel ANN/HMM hybrid (middle curve), and ANN/HMM hybrid with Case 3 of adaptive amplitudes (upper curve).*

The present noisy corpus, collected under the VODIS-II/SpeechDatCar projects[3] (Matassoni et al., 1999) in a real car environment, is available from ELRA. It consists of 200 speakers (100 male and 100 female) recorded under a variety of noisy car conditions (e.g. traffic, different car speed, various road/highway and weather conditions, etc.). Each speaker uttered variable-length, continuous Italian digit-strings, during two distinct sessions, for a total of 2700 utterances. Each string has variable length, ranging from a minimum of 1 to a maximum of 16 words, with an average length of about 6 words.

Signals were acquired using a close-mouth microphone placed in front of the driver. It is worth underlining that, while the 20 speakers from the SPK database used for training are mostly from the North-East of Italy, and are aged in the range 25-40, the VODIS-II/SpeechDatCar corpus was designed to cover the Country in a rather uniform manner (i.e., a variety of regional accents is represented), along with a much wider-range coverage of the age of speakers (from very young drivers to old man/women). These facts deepen the gap between the training and test conditions, that are no longer limited to the presence of noise as in the previous section. Further differences are brought in by the use of a different microphone, and by the spontaneous changes in vocal tracts induced in the speaker by the acoustic conditions of the environment, mainly due to the *Lombard* effect.

Of course, the same feature extraction process as in the previous experiments was applied: 8 MFSCCs and the signal log-energy were extracted as usual, in order to allow application of the same acoustic models trained above. Table 6.3 reports the results.

Acoustic Model	*SRR (%)*	*WRR (%)*
Standard HMM	37.58	**71.93**
Bourlard and Morgan's hybrid	41.80	73.96
ANN/HMM hybrid without adaptive amplitudes	53.68	**80.58**
ANN/HMM hybrid with single adaptive λ	53.68	80.56
ANN/HMM hybrid with layer-by-layer adaptive λ_ℓ's	56.50	82.74
ANN/HMM hybrid with neuron-by-neuron adaptive $\lambda_{\ell,i}$'s	57.02	**83.07**

Table 6.3: *String Recognition Rate (SRR) and Word Recognition Rate (WRR) yielded by the different acoustic models over the noisy test signals of the* VODIS II/SpeechDatCar *continuous-speech corpus. Signals collected in a real car environment, 200 test speakers.*

As a matter of fact, the same conclusions drawn in the previous section still hold, as far as concerns the comparison among the different models, their noise-tolerance, etc. In the present setup, a minor exception is represented by Case 1, that even

[3]Part of SpeechDatCar and Vodis-II projects is funded by the Commission of the EC, Telematics Applications Programme, Language Engineering, Contracts LE4-8334 and LE4-8336.

slightly worsens the performance of the bare ANN/HMM hybrid. The gain in terms of WRR provided by the connectionist models (including Bourlard and Morgan;s paradigm) over the HMM confirms the expectations of generalization properties in ANNs if compared with memory-based approaches (the HMM). Improvement in terms of WRR over the HMM and the bare ANN/HMM hybrid yielded by Cases 2 and 3 offers a full validation of these tecfhniques, and does not require any further comments.

6.4 Final Remarks

One of the most severe limitations in standard HMMs is their intrinsically limited generalization capabilities, since they are memory-based learning machines and they actually lack of *ad hoc* training schemes (e.g. regularization techniques) capable to explicitly tackle such problem. In fact, this problem has a strong negative influence on the test recognition performance under (noisy) acoustic conditions that differ from those that characrize the training set.

In this chapter we aimed at (a) evaluating robustness to noise of the novel ANN/HMM hybrid, and (b) developing a technique that explicitly improved learning and generalization properties of the model. The latter technique, relying on adaptive amplitudes of activation functions, was inherited from the algorithms proposed in Chapter 3 for feed-forward and recurrent ANNs. The algorithms were adapted to fit the present gradient ascent procedure for maximization of the likelihood of the training acoustic observations given the model.

An instance of the so-called “divergence problem” emerged as an immediate consequence, due to the expected tendency of output amplitudes to grow in an unbound manner while increasing the likelihood. The problem was tackled as in the bare ANN/HMM hybrid learning rules, applying the Maximum-A-Posteriori (*MAP*) training criterion. The latter attempts to maximize the likelihood given the constrained model, while decreasing the likelihood given the recognition model. The technique has been shown to avoid divergence of the model parameters.

The most performing hybrid model obtained in Chapter 5, namely a 2-layer MLP with hidden and output sigmoids, trained on-line in the linear domain, was tested and favorably compared with the standard HMM, and with Bourlard and Morgan’s paradigm, in two distinct noisy scenarios. In the former, noise collcetd in an office was introduced in the clean speech signals of the SPK database through a simple additive model, allowing for an analysis in function of the SNR. In the latter, signals from the VODIS II/SpeechDatCar database –collected among 200 speakers in a real car environment, under a number of acoustic conditions – were usd for the test. Results confirmed the robustness of the hybrid with respect to the HMM, due to the generalization capabilities of the connectionist model, as well as the effectiveness of the algorithms to train amplitudes. Latter ones, namely Case 2 (layer-by-layer version) and Case 3 (neuron-by-neuron version), allowed for convergence of the model toward different maxima in the parameter space, defining different search paths in the weight space that are a consequence of the modified training schemes, i.e., induced by the specific "soft" weight-grouping mechanism that they implicitly realize; groups that were proven to have high Bayesian evidence by David MacKay (MacKay, 1992b; MacKay, 1994) within his Bayesian framework for comparing different regularizers.

Chapter 7

Summary and Conclusions

Automatic Speech Recognition has been the most investigated topic in speech processing along the last few decades. In spite of the fact that ASR has resulted effective in several applicative scenarios, a number of problems arises when ASR technology is moved from laboratories to real-world conditions, where a dramatic degradation in performance is far from being an exception.

At the state of the art, hidden Markov models are the most popular approach to the problem of acoustic modeling in ASR. Although HMMs allow for good recognition performance under many circumstances, they suffer from major limitations. In particular:

- HMMs rely on strong assumptions on the statistical properties of the phenomena under consideration. The assumption of a given parametric form for the probability density functions that represent the emission probabilities associated with HMM states is arbitrary and constraining.
- In HMMs attention is focused on *learning* (i.e., "memorizing") the training set, and not on *generalization* capabilities; as a matter of fact, no regularization theory for HMMs has been developed so far.

The latter fact turns out to be critical in real-world, noisy applications where *robustness* of the ASR system can be expressed, in terms of machine learning theory, as the generalization capability of the recognizer itself. Given these limitations, the use of artificial neural networks appears definitely promising. Indeed, ANNs can be trained as non-parametric probabilistic models, and they show interesting generalization capabilities. The latter can be further improved relying on *ad hoc* regularization theories. In fact, in order to deal with noise in the speech signal, and with variations in acoustic conditions, *robust* connectionist models are sought. The first topic of the present Thesis was the development of a training technique capable to improve learning and generalization in feedforward and recurrent nets, and to evaluate its behavior over speech data. In this respect, we introduced novel algorithms to learn the amplitudes of non-linear activations in layered networks, without any assumption on their analytical form. Three instances of the algorithms were developed: (i) a common amplitude is shared among all nonlinear units; (ii) each layer has its own amplitude; (iii) neuron-specific amplitudes are allowed. It was shown that the algorithms can be seen as a particular double-step gradient-descent procedure, as gradient-driven adaptive learning rate schemes, or as weight-grouping techniques that are consistent with the Bayesian framework by David MacKay. The introduction of a pseudo-parameter λ (i.e., the amplitude) within the training scheme modifies the search space and, accordingly, the search paths for the minimum of the criterion function within an emerging hyperspace, resulting in increased learning and generalization properties and allowing for robustness to noise. Experimental results on function approximation, classification and regression tasks, with synthetic and real-world speech data, validated the approach to a large extent. An ASR system based on a static connectionist model, called Segmental Neural Network with Trainable Amplitudes, was proved to outperform a standard HMM in an isolated words, small vocabulary recognition task in noisy environments.

In spite of their ability to classify short-time acoustic-phonetic unis, such as individual phonemes or few isolated words, ANNs historically failed as a general framework for ASR. Failure emerges in dealing with long sequences of acoustic observations, like those required in order to represent words from a large dictionary or whole sentences (continuous speech recognition). This is mainly due to the lack of ability to model long-term dependencies in ANNs, even when recurrent architectures are considered. In the early Nineties this fact led to the idea of combining HMMs and ANNs within a unifying, novel model, broadly known as hybrid ANN/HMM. A number of significant, different hybrid paradigms for ASR were proposed in the literature. Early attempts basically aimed at emulating HMMs by ANNs. A few years later, Bourlard and Morgan proposed ANNs to estimate posterior probabilities of the states of a HMM, and Bengio (following Bridle) developed a "global" optimization scheme, where a single, overall training criterion is maximized jointly by the HMM and the ANNs. Connectionist Vector Quantization for discrete HMMs, and other more recent approaches were also considered.

Unfortunately, state-of-the-art ANN/HMM hybrid systems do not fully exploit the potential advantages that could be taken from the combination of the underlying paradigms, mainly due to the following reasons.

1. In most hybrid architectures ANNs play only a marginal role, while the kernel of the recognizer still relies on a standard HMM. This is the case of Bengio's approach. The latter basically brings into the resulting system most of the usual limitations of HMMs.

2. In the major cases of connectionist probability estimation for an underlying HMM, namely in Bourlard and Morgan's model and its many derivates, the lack of a globally-defined, mathematically motivated *training* scheme strongly weakens the theoretical framework, and substantially reduces the improvement in performance on the field.

3. No actual hybrid system stressed (enough) the generalization capabilities of ANNs in order to tackle the problem of ASR in noisy environments under different, adverse conditions.

The search for theoretically motivated as well as effective solutions to these problems formed the motivation for the development of a novel ANN/HMM hybrid paradigm.

We introduced novel algorithms based on a gradient-ascent technique, aimed at maximizing the likelihood of acoustic observations given the model, for global training of a hybrid ANN/HMM system. The ANN is used as a non-parametric estimator of emission probabilities associated with individual states of the HMM. The approach is clearly related to the hybrid systems proposed by Bourlard & Morgan and by Bengio, respectively, with the twofold goal of combining the benefits from both within a unifying framework, and of overcoming their limitations.

Once the generic (BML) training algorithm was given, a critical point had to be considered, referred to as the "divergence problem:: how can we ensure that the ANN outputs do actually estimate likelihoods? Major answers to the question were given by: (1) imposing a probabilistic constraint on ANN weights (SWS-ML algorithm); (2) factorizing the emission probabilities via Bayes' Theorem, i.e. performing connectionist *state-posterior* probability estimation (*Bayes* algorithm); (3) extremizing a *Maximum A Posteriori* training criterion, instead of the bare likelihood (MAP algorithm).

Experimental demonstrations were carried out, where a reduced-scale task (speaker dependent, isolated digits recognition) was used to analyze the behavior of the algorithms and to compare them with a standard HMM. In so doing, the MLP with sigmoidal hidden and output units, along with the on-line version of the hybrid training algorithm in the linear domain, emerged as the most promising. In addition, an effective initialization procedure strictly related with Bourlard and Morgan's training scheme was introduced and favorably evaluated. On these basis, a real-world, speaker independent continuous speech recognition task was approached. Results validated the novel hybrid framework, emphasizing its advantages over standard HMMs, as well as over Bourlard and Morgan's paradigm. All of the proposed solutions to the divergence problem (SWS-ML, Bayes and MAP) turned out to be viable on the field, allowing for improved recognition performance: in particular, the MAP algorithm yielded a 46.34% relative word error rate reduction with respect to the standard HMM.

At that point, the missing ring was represented by the combination between the ANN/HMM hybrid training scheme and the algorithms to train the amplitude of activation functions. Improved learning and generalization capabilities, i.e. robustness, of the overall recognition system, were expected. The strict relationship between noise-tolerance and generalization has to be stressed again, keeping in mind the above statements concerning the memory-based aspects of the learning process in HMMs. Exactly like in the case of static connectionist models, three instances of the algorithms were developed (unique λ, layer-by-layer, unit-by-unit). Experiments were accomplished on noisy ASR tasks, namely connected digits recognition, in two distinct setups: (i) office noise was added to clean signals from the SPK database at different SNRs; (ii) the VODIS II/SpeechDatCar database, collected in a real car environment, was used. The results confirmed a further improvement in terms of recognition performance, highlighting that the unit-by-unit case is best suited to the particular architecture/problem under consideration. In a broader perspective, once the whole ASR system relies on an ANN, techniques aimed at improving convergence and generalization properties of the latter (e.g., regularization theory) may be successfully applied to the ANN/HMM hybrid in a straightforward manner.

7.1 Where do we go from here?

The models developed throughout this Thesis turned out to be viable approaches to the problems stated at the very beginning, overcoming fundamental limitations of standard HMMs. They provide us with a significant improvement of performance in real-world ASR tasks, as well as with increased generalization capabilities, i.e. robustness to noise. Nonetheless, evolving strategies for the novel ANN/HMM hybrid are to be taken into full consideration at this point. Mainstreams for further developments may be pursued along two major directions:

1. application of the proposed models to harder ASR tasks. Latter ones include large dictionary speech recognition (e.g., more than 10000 words), modeling of more complex acoustic units (e.g. *biphones*) via context-dependent units, combination of different acoustic features, ASR over the telephone line, etc.

2. Development of globally-trained connectionist feature transformation modules, e.g. ANNs for speaker normalization or channel compensation. Such modules can be applied as acoustic front-ends to the ANN/HMM hybrid through a kind of *model inversion* (Moon and Hwang, 1997) strategy. ANNs that perform feature transformations, like those described in (Trentin and Giuliani, 1997; Trentin and Giuliani, 2001), are known to improve recognition performance in situations characterized by a significant mismatch between the acoustic conditions under which the training and the test sets are acquired. The experiments reported above, i.e. training over clean signals and recognition in presence of noise, are examples of such a scenario. At the state-of-the-art, standard BP training of the ANNs is usually accomplished over a certain *distance* (error) criterion, that is arbitrary and differs from the overall goal of the whole recognition system (the likelihood of the acoustic observations given the model). The inversion principle requires the optimization of the weights of the connectionist acoustic front-end upon backpropagation of the partial derivatives of the global criterion through the acoustic model (parameters of the latter are left unchanged).

In light of the results obtained so far it is our belief that, following such mainstreams and providing the hybrid acoustic model with a state-of-the-art language model, difficult ASR tasks can be performed, avoiding the inheritance of the well-known drawbacks of popular standard HMMs.

Appendix A

A.1 Proof of Theorem 3.2

The proof follows the inductive scheme on the total number of layers L used in the proof of Theorem 1. For $L = 1$ we need to prove that $\frac{\partial o_{k,1}}{\partial \lambda_m} = x_{k,1,m}(a_{k,1})$. Since $o_{j,0} = x_j, \forall j \in \mathcal{L}_0$, where x_j denotes the j-th component of the input vector $\mathbf{x}$, we have:

$$\frac{\partial o_{j,0}}{\partial \lambda_m} = 0 \tag{A.1}$$

and, recalling equation (3.32), we immediately obtain:

$$\begin{aligned} \frac{\partial o_{k,1}}{\partial \lambda_m} &= g_{k,1,m}(a_{k,1}) \\ &= x_{k,1,m}(a_{k,1}) \end{aligned} \tag{A.2}$$

where we used the definition (3.33) of Section 3.2.2, completing the inductive basis. Let us now assume that the theorem has been proven for up to $L - 1$ layers. We must show that, with this hypothesis, it holds also for networks with L layers. If $\frac{\partial o_{j,L-1}}{\partial \lambda_m} = x_{j,L-1,m}(a_{j,L-1})$ equation (3.32) (for unit k in layer $\mathcal{L}_L$) can be written as:

$$\begin{aligned} \frac{\partial o_{k,L}}{\partial \lambda_m} &= g_{k,L,m}(a_{k,L}) + h_{k,L,m}(a_{k,L}) \sum_{j \in \mathcal{L}_{L-1}} w_{k,j,L} x_{j,L-1,m}(a_{j,L-1}) \\ &= x_{k,L,m}(a_{k,L}) \end{aligned} \tag{A.3}$$

according to (3.33). This completes the proof. $\mathcal{Q}.\mathcal{E}.\mathcal{D}$.

A.2 Proof of Theorem 5.1

This proof was given by Bengio (Bengio et al., 1992b; Bengio, 1993; Bengio, 1996) in the case of *log-likelihoods*. Let us consider $t = T$. In this assumption we can write:

$$\begin{aligned} \frac{\partial L}{\partial \alpha_{i,T}} &= \frac{\partial \sum_{k \in \mathcal{F}} \alpha_{k,T}}{\partial \alpha_{i,T}} \\ &= \begin{cases} 1 & \text{if } i \in \mathcal{F} \\ 0 & \text{otherwise} \end{cases} \\ &= \beta_{i,T} \end{aligned} \tag{A.4}$$

which completes the inductive basis. Let us now assume that the theorem has been proved for $T, T-1, \ldots, t+1$ and let us show that it holds true for t.

Using the chain rule it is possible to write:

$$\frac{\partial L}{\partial \alpha_{i,t}} = \sum_k \frac{\partial L}{\partial \alpha_{k,t+1}} \frac{\partial \alpha_{k,t+1}}{\partial \alpha_{i,t}} \tag{A.5}$$

which, using the inductive hypothesis and the definition (5.3) of $\alpha_{i,t}$, can be rewritten as:

$$\begin{aligned} \frac{\partial L}{\partial \alpha_{i,t}} &= \sum_k \beta_{k,t+1} \frac{\partial b_{k,t+1} \sum_j a_{jk} \alpha_{j,t}}{\partial \alpha_{i,t}} \\ &= \sum_k \beta_{k,t+1} b_{k,t+1} a_{ik} \end{aligned} \tag{A.6}$$

that is, by definition (5.4) of $\beta_{i,t}$:

$$\frac{\partial L}{\partial \alpha_{i,t}} = \beta_{i,t}. \tag{A.7}$$

This completes the proof. $\mathcal{Q}.\mathcal{E}.\mathcal{D}$

Bibliography

Angelini, B., Brugnara, F., Falavigna, D., Giuliani, D., Gretter, R., and Omologo, M. (1994). Speaker independent continuous speech recognition using an acoustic-phonetic italian corpus. In *Proc. of ICSLP*, pages 1391–1394, Yokohama.

Austin, S., Zavaliagkos, G., Makhoul, J., and Schwartz, R. (1992). Speech recognition using segmental neural nets. In *International Conference on Acoustics, Speech and Signal Processing*, pages I–625–628, San Franscisco.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58.

Bell, A. J. and Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159.

Bellini, S. and Rocca, F. (1988). Near optimal blind deconvolution. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages vol. 4 pp2236–2239.

Bengio, S. and Bengio, Y. (1996). An EM algorithm for asynchronous input/output hidden Markov models. In Xu, L., editor, *International Conference On Neural Information Processing*, Hong-Kong.

Bengio, Y. (1990). Radial basis functions for speech recognition. In *Speech Recognition and Understanding: Recent Advances, Trends and Applications*, pages 293–298. NATO Advanced Study Institute Series F: Computer and Systems Sciences.

Bengio, Y. (1993). A connectionist approach to speech recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):647–667.

Bengio, Y. (1996). *Neural Networks for Speech and Sequence Recognition*. International Thomson Computer Press, London, UK.

Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. (1991). Phonetically motivated acoustic parameters for continuous speech recognition using artificial neural networks. In *Proceedings of EuroSpeech'91*.

Bengio, Y. and Frasconi, P. (1996). Input/Output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249.

Bengio, Y., Gori, M., and De Mori, R. (1992a). Learning the dynamic nature of speech with back-propagation for sequences. *Pattern Recognition Letters*, 13(5):375–386. (Special issue on Artificial Neural Networks).

Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. (1992b). Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transactions on Neural Networks*, 3(2):252–259.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166. Special Issue on Recurrent Neural Networks, March 94.

Bhat, N. V., Minderman, P. A., Jr, M. T., and Wang, N. S. (1990). Modeling chemical process systems via neural computation. *IEEE Control Syst. Mag.*, pages 24–30.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294.

Bourlard, H. and Morgan, N. (1993). Continuous speech recognition by connectionist statistical methods. *IEEE Trans. on Neural Networks*, 4(6):893–909.

Bourlard, H. and Morgan, N. (1994a). *Connectionist Speech Recognition. A Hybrid Approach*, volume 247 of *The Kluwer international series in engineering and computer science.* Kluwer Academic Publishers, Boston.

Bourlard, H. and Morgan, N. (1994b). *Connectionist Speech Recognition. A Hybrid Approach.* Kluwer Academic Publishers, Boston. page 117.

Bourlard, H. and Wellekens, C. (1990). Links between hidden Markov models and multilayer perceptrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:1167–1178.

Box, G. E. P., Hunter, W. G., and Hunter, J. S. (1978). *Statistics for Experimenters.* John Wiley and Sons, New York.

Bridle, J. (1990). Alphanets: a recurrent 'neural' network architecture with a hidden Markov model interpretation. *Speech Communication*, 9(1):83–92.

Carpenter, G. and Grossberg, S. (1987). ART2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930.

Carpenter, G. and Grossberg, S. (1988). The ART of adaptive pattern recognition by a self-organizing neural network. *Computer*, pages 77–88.

Cerf, P. L., Ma, W., and Compernolle, D. V. (1994). Multilayer perceptrons as labelers for hidden markov models. *IEEE Trans. on Speech and Audio Processing*, 2(1):185–193.

Chang, P. C. and Juang, B. H. (1993). Discriminative training of dynamic programming based speech recognizers. *IEEE Transactions on Speech and Audio Processing*, 1:135–143.

Che, C., Lin, Q., Pearson, J., Devries, B., and Flanagan, J. (1994). Microphone arrays and neural networks for robust speech recognition. In *Proceedings of ARPA HLT Workshop*, pages 342–347, New Jersey.

Chen, C. T. and Chang, W. D. (1996). A feedforward neural network with function shape autotuning. *Neural Networks*, 9(4):627–641.

Chen, W. Y., Chen, S. H., and Lin, C. J. (1996). A speech recognition method based on the sequential multi-layer perceptrons. *Neural Networks*, 9(4):655–669.

Chung, Y. J. and Un, C. K. (1996a). An MLP/HMM hybrid model using nonlinear predictors. *Speech Communication*, 19(4):307–316.

Chung, Y. J. and Un, C. K. (1996b). Multilayer perceptrons for state-dependent weighting of HMM likelihoods. *Speech Communication*, 18:79–89.

Cosi, P., Bengio, Y., and De Mori, R. (1990). Phonetically-based multi-layered networks for acoustic property extraction and automatic speech recognition. *Speech Communication*, 9(1):15–30.

Cosi, P., Frasconi, P., Gori, M., and Griggio, N. (1992). Phonetic recognition experiments with recurrent neural networks. In *Proc. of the International Conference on Spoken Language*, pages 1335–1338, Banff (Canada).

Cottrell, G., Munro, P., and Zipser, D. (1987). Learning internal representations from gray-scale images: An example of extensional programming. In *Ninth Annual Conference of the Cognitive Science Society*, pages 462–473, Seattle 1987. Lawrence Erlbaum, Hillsdale.

Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314.

Das, S., Nádas, A., Nahamoo, D., and Pichney, M. (1994). Adaptation techniques for ambience and microphone compensation in the IBM Tangora speech recognition system. In *International Conference on Acoustics, Speech and Signal Processing*, pages I–21–24, Adelaide.

Davis, S. B. and Mermelstein, P. (1980). Comparison of parametric representations of monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 28(4):357–366.

De Mori, R. (1998). *Spoken Dialogues with Computers*. Academic Press, London, UK.

Deller, J. R. J., Proakis, J. G., and Hansen, J. H. (1993). *Discrete Time Processing of Speech Signals*. Macmillan Publishing Company, New York, USA.

Deprettere, F. (1988). *SVD and Signal Processing Algorithms, Applications and Architectures*. North-Holland, Amsterdam.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley, New York.

Dugast, C., Devillers, L., and Aubert, X. (1994). Combining TDNN and HMM in a hybrid system for improved continuous-speech recognition. *IEEE Trans. on Speech and Audio Processing*, 2(1):217–223.

Dupont, S., Ris, C., Deroo, O., Fontaine, V., Boite, J. M., and Zanoni, L. (1997). Context-independent and context-dependent hybrid HMM/ANN systems for vocabulary independent tasks. In *Proc. of EUROSPEECH*, volume 4, pages 1947–1950, Rhodi.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Elman, J. and Zipser, D. (1988). Learning the hidden structure of speech. *Journal of the Acoustical Society of America*, 83:1615–1626.

Falavigna, D. (1995). Comparison of Different HMM Based Methods for Speaker Verification. In *Proceedings of EuroSpeech 95*, volume 1, pages 371–374, Madrid.

Farrel, K. R., Mammone, R. J., and Assaleh, K. T. (1994). Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions on Speech and Audio Processing*, 2(1):194–205.

Feng, M.-W. (1988). Improved speaker adaptation using text dependent spectral mappings. In *Proc. of ICASSP*, pages I–131–134.

Flammia, G. (1991). Speaker independent consonant recognition in continuous speech with distinctive phonetic features. Master's thesis, McGill University, School of Computer Science.

Flanagan, J. L., Berkley, D. A., Elko, G. W., West, J. E., and Sondhi, M. M. (1991). Autodirective microphone systems. *Acustica*, 75:58–71.

Franco, H., Cohen, M., Morgan, N., Rumelhart, D., and Abrash, V. (1994). Context-dependent connectionist probability estimation in a hybrid hidden Markov model-neural net speech recognition system. *Computer Speech and Language*, 8:211–222.

Franco, H. and Digalakis, V. (1995). Temporal correlation modeling in a hybrid neural network/hidden Markov model speech recognizer. In *Proc. of EUROSPEECH*, pages 1681–1684, Madrid.

Franzini, M., Lee, K., and Waibel, A. (1990). Connectionist Viterbi training: a new hybrid method for continuous speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages 425–428, Albuquerque, NM.

Frasconi, P., Gori, M., and Soda, G. (1990). Recurrent networks for continuous speech recognition. In *Computational Intelligence 90*, Milano (Italy). Elsevier.

Fukunaga, K. (1990). *Statistical Pattern Recognition.* Academic Press, San Diego, second edition.

Furlanello, C., Giuliani, D., Trentin, E., and Merler, S. (1997). Speaker normalization and model selection of combined neural nets. *Connection Science*, 9(1):31–50. Special Issue on Combining Neural Nets.

Furui, S. (1981). Cepstral analysis techniques for automatic speaker verification. *IEEE Transactions on ASSP*, 29(2):254.

Furui, S. and Sondhi, M. M., editors (1991). *Advances in Speech Signal Processing.* Marcel Dekker and Inc.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58.

Gemello, R., Albesano, D., and Mana, F. (1997). Continuous speech recognition with neural networks and stationary-transitional acoustic units. In *ICNN*, pages 2107–2111, Houston, Texas, USA.

Giraud, B. (1996). Flexibility, robustness and algebraic convergence of neural nets with neurons having a window-like response function. In *Workshop on Artificial Neural Networks (ANN)*, Dept. Informatique et Recherche Operationnelle, Universite de Montreal.

Girosi, F., Jones, M., and Poggio, T. (1995). Regularization theory and neural network architectures. *Neural Computation*, 7:1219–269.

Giuliani, D., Matassoni, M., Omologo, M., and Svaizer, P. (1999). Training of hmm with filtered speech material for hands-free recognit ion. In *International Conference on Acoustics, Speech and Signal Processing*, Phoenix.

Gori, M., Bengio, Y., and De Mori, R. (1989). BPS: A learning algorithm for capturing the dynamical nature of speech. In *Proceedings of the International Joint Conference on Neural Networks*, pages 643–644, Washington D.C. IEEE, New York.

Gori, M., Lastrucci, L., and Soda, G. (1996). Autoassociator-based models for speaker verification. *Pattern Recognition Letters*, 17:241–250.

Gori, M. and Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-14(1):76–86.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134.

Gurgen, F. S., Song, J. M., and King, R. W. (1994). A continuous HMM based preprocessor for modular speech recognition neural networks. In *Proc. of ICSLP*, pages 1507–1510, Yokohama.

Guyon, I. (1991). Applications of neural networks to character recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 5:353–382.

Haffner, P., Franzini, M., and Waibel, A. (1991). Integrating time alignment and neural networks for high performance continuous speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages 105–108, Toronto.

Haffner, P., Waibel, A., and Shikano, K. (1989). Fast back-propagation learning methods for large phonemic neural networks. In *Proceedings of Eurospeech'89*.

Hampshire, J. B. and Waibel, A. H. (1990). A novel objective function for improved phoneme recognition using time-delay neural networks. *IEEE Transactions on Neural Networks*, 1(2):216–228.

Haykin, S. (1994). *Neural Networks (A Comprehensive Foundation)*. Macmillan, New York.

Hennebert, J., Ris, C., Bourlard, H., Renals, S., and Morgan, N. (1997). Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems. In *Proc. of EUROSPEECH*, volume 4, pages 1951–1954, Rhodi.

Hermansky, H. (1990). Perceptual linear predictive (plp) analysis for speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752.

Hertz, J., Krogh, A., and Palmer, R. (1991). *Introduction to the Theory of Neural Computation*. Addison Wesley.

Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). Boltzmann machines: Constraint satisfaction networks that learn. Technical Report TR-CMU-CS-84-119, Carnegie-Mellon University, Dept. of Computer Science.

Hochberg, M. M., Renals, S. J., Robinson, A. J., and Cook, G. D. (1995). Recent improvements to the ABBOT large vocabulary csr system. In *International Conference on Acoustics, Speech and Signal Processing*, pages 69–72, Detroit.

Hochberg, M. M., Renals, S. J., Robinson, A. J., and Kershaw, D. J. (1994). Large vocabulary continuous speech recognition using a hybrid connectionist-HMM system. In *Proc. of ICSLP*, pages 1499–1502, Yokohama.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 79:2554–2558.

Hopfield, J. and Tank, D. (1986). Computing with neural circuits: A model. *Science*, 233:625–633.

Hu, Z. and Shao, H. (1992). The study of neural network adaptive control systems. *Control and Decision*, 7:361–366.

Huang, X. D. (1992). Speaker normalization for speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages I–465–468, San Franscisco.

Huang, X. D., Ariki, Y., and Jack, M. (1990). *Hidden Markov Models for Speech Recognition*. Edinburgh University Press, Edinburgh.

Hutter, H.-P. (1995). Comparison of a new hybrid connectionist-SCHMM approach with other hybrid approaches for speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3311–3314, Detroit.

Iso, K. and Watanabe, T. (1990). Speaker-independent word recognition using a neural prediction model. In *International Conference on Acoustics, Speech and Signal Processing*, pages 441–444, Albuquerque, NM.

Iwamida, H., Katagiri, S., and McDermott, E. (1991). Speaker-independent large vocabulary word recognition using an LVQ/HMM hybrid algorithm. In *International Conference on Acoustics, Speech and Signal Processing*, pages 553–556, Toronto.

Jacobs, R. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixture of local experts. *Neural Computation*, 3:79–87.

Jang, C. S. and Un, C. K. (1996). A new parameter smoothing method in the hybrid TDNN/HMM architecture for speech recognition. *Speech Communication*, 19(4):317–324.

Jelinek, F. (1998). *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MA.

Johansen, F. T. (1994). Global optimisation of HMM input transformations. In *Proc. of ICSLP*, volume 1, pages 239–242, Yokohama.

Johansen, F. T. (1996). A comparison of hybrid HMM-architectures using global discriminative training. In *Proc. of ICSLP*, pages 498–501, Philadelphia.

Johansen, F. T. and Johnsen, M. H. (1994). Non-linear input transformations for discriminative HMMs. In *International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 225–228, Adelaide.

Jordan, M. (1989). Serial order: A parallel, distributed processing approach. In Elman, J. and Rumelhart, D., editors, *Advances in Connectionist Theory: Speech.* Lawrence Erlbaum, Hillsdale.

Juang, B. H. and Katagiri, S. (1992). Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3054.

Junqua, J. C. and Haton, J. P. (1996). *Robustness in Automatic Speech Recognition: Fundamentals and Applications.* Kluwer Academic Publishers, Boston, USA.

Katagiri, S., Lee, C.-H., and Juang, B. H. (1991). New discriminative training algorithms based on the generalized probabilistic descent method. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 299–308.

Kimber, D., Bush, M. A., and Tajchman, G. N. (1990). Speaker-independent vowel classification using hidden Markov models and LVQ2. In *International Conference on Acoustics, Speech and Signal Processing*, pages 497–500, Albuquerque, NM.

Kirkpatrick, S., Jr., C. G., , and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220.

Kohonen, T. (1986). Learning vector quantization for pattern recognition. Report TKK-F-A601, Helsinki University of Technology, Espoo, Finland.

Kohonen, T. (1989). *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, 3 edition.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

Kohonen, T., Barna, G., and Chrisley, R. (1988). Statistical pattern recognition with neural networks: Benchmarking studies. In *IEEE International Conference on Neural Networks*, volume 1, pages 61–68, San Diego 1988. IEEE, New York.

Lang, K. J. and Hinton, G. E. (1988). The development of the time-delay neural network architecture for speech recognition. Technical Report CMU-CS-88-152, Carnegie-Mellon University.

Le Cun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 255–257. MIT Press.

LeCun, Y. (1986). Learning processes in an asymmetric threshold network. In Soulie, F. F., Bienenstock, E., and Weisbuch, G., editors, *Disordered Systems and Biological Organization*, pages 233–240. Springer-Verlag, Les Houches, France.

LeCun, Y. (1989). Generalization and network design strategies. Technical Report CRG-TR-89-4, Department of Computer Science, University of Toronto.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 396–404, Denver, CO. Morgan Kaufmann, San Mateo.

Lee, C.-H., Lin, C.-H., and Juang, B.-H. (1991). A study on speaker adaptation of the parameters of continuous density hidden Markov models. *IEEE Trans. on Signal Processing*, 39(4):806–814.

Lee, C. H., Soong, F. K., and Paliwal, K. K., editors (1996). *Automatic Speech and Speaker Recognition: Advanced Topics.* Kluwer Academic Publishers, Boston, USA.

Levin, E. (1990). Word recognition using hidden control neural architecture. In *International Conference on Acoustics, Speech and Signal Processing*, pages 433–436, Albuquerque, NM.

Levin, E., Pieraccini, R., and Bocchieri, E. (1992). Time-warping network: a hybrid framework for speech recognition. In Moody, J., Hanson, S., and Lippmann, R., editors, *Advances in Neural Information Processing Systems 4*, pages 151–158, Denver, CO.

Li, K. P. and Naylor, J. A. (1992). A whole word recurrent neural network for keyword spotting. In *International Conference on Acoustics, Speech and Signal Processing*, pages II–81–84, San Franscisco.

Li, K. P. and Porter, J. E. (1988). Normalizations and selection of speech segments for speaker recognition scoring. In *Proceedings of ICASSP 88*, page 595.

Linden, A. and Kindermann, J. (1989). Inversion of multilayer nets. In *International Joint Conference on Neural Networks*, pages 425–430, Washington, DC.

Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22.

Lippmann, R. (1989). Review of neural networks for speech recognition. *Neural Computation*, 1:1–38.

Lippmann, R. and Singer, E. (1993). Hybrid neural-network/HMM approaches to wordspotting. In *International Conference on Acoustics, Speech and Signal Processing*, pages I–565–568.

Lippmann, R. P. and Gold, B. (1987). Neural classifiers useful for speech recognition. In *IEEE Proc. First Intl. Conf. on Neural Networks*, volume IV, pages 417–422, San Diego, CA.

Ma, W. and Compernolle, D. V. (1990). TDNN labeling for a HMM recognizer. In *International Conference on Acoustics, Speech and Signal Processing.*

MacKay, D. J. C. (1992a). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

MacKay, D. J. C. (1992b). A practical bayesian framework for backpropagation networks. *Neural Computation*, 4(3):448–472.

MacKay, D. J. C. (1994). Bayesian non-linear modeling for the prediction competition. *ASHRAE Transactions*, 100:1–16.

Mäntysalo, J., Torkkola, K., and Kohonen, T. (1992). Experiments on the use of LVQ in phoneme-level segmentation of speech. In *Proc. of the Second Workshop on Neural Networks for Speech Processing*, pages 39–52, Firenze (Italy). LINT.

Mari, J.-F., Fohr, D., Anglade, Y., and Junqua, J.-C. (1994). Hidden Markov models and selectively trained neural networks for connected confusable word recognition. In *Proc. of ICSLP*, pages 1519–1522, Yokohama.

Matassoni, M., Omologo, M., Cristoforetti, L., Giuliani, D., Svaizer, P., Trentin, E., and Zovato, E. (1999). Some results on the development of a hands-free speech recognizer for car-environment. In *Proceedings of the 1999 international workshop on Automatic Speech Recognition and Understanding (ASRU)*, Keystone, Colorado, USA.

McDermott, E. and Katagiri, S. (1989). Shift-invariant, multi-category phoneme recognition using Kohonen's LVQ2. In *International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 81–84, Glasgow, Scotland. IEEE.

McDermott, E. and Katagiri, S. (1991). LVQ-based shift-tolerant phoneme recognition. *IEEE Transactions on Signal Processing*, 39(6):1398–1411.

Menendez-Pidal, X., de Cordoba, R., Ferreiros, J., and Pardo, J. M. (1995). Incorporating fuzzy modelling in a hybrid HMM-ANNs system for CSR tasks. In *Proc. of EUROSPEECH*, pages 1689–1692, Madrid.

Menendez-Pidal, X., Ferreiros, J., de Cordoba, R., and Pardo, J. M. (1994). Recent work in hybrid neural networks and HMM systems in CSR tasks. In *Proc. of ICSLP*, pages 1515–1518, Yokohama.

Merhav, N. and Lee, C. H. (1993). A minimax classification approach with application to robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1:90–100.

Minsky, M. and Papert, S. (1969). *Perceptrons.* MIT Press, Cambridge.

Mokbel, C., Jouvet, D., Monné, J., and De Mori, R. (1998). Robust speech recognition. In De Mori, R., editor, *Spoken Dialogues with Computers*, pages 435–439, London, UK. Academic Press.

Moody, J. and Darken, C. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294.

Moon, S. and Hwang, J. N. (1997). Robust speech recognition based on joint model and feature space optimization of hidden Markov models. *IEEE Transactions on Neural Networks*, 8(2):194–204.

Morgan, D. P. and Scofield, C. L. (1991). *Neural Networks and Speech Processing.* Kluwer, Norwell, Mass.

Morgan, D. P., Scofield, C. L., and Adcock, J. E. (1991). Multiple neural network topologies applied to keyword spotting. In *International Conference on Acoustics, Speech and Signal Processing*, pages 313–316, Toronto.

Morgan, N. and Bourlard, H. (1990). Continuous speech recognition using multilayer perceptrons with hidden Markov models. In *International Conference on Acoustics, Speech and Signal Processing*, pages 413–416, Albuquerque, NM.

Morgan, N., Konig, Y., Wu, S., and Bourlard, H. (1995). Transition-based statistical training for ASR. In *Proceedings of IEEE Automatic Speech Recognition Workshop (Snowbird)*, pages 133–134.

Moudenc, T., Sokol, R., and Mercier, G. (1996). Segmental phonetic features recognition by means of neural-fuzzy networks and integration in an N-best solutions post-processing. In *Proc. of ICSLP*, pages 338–341, Philadelphia.

Mozer, M. C. (1993). Neural net architectures for temporal sequence processing. In Weigend, A. and Gershenfeld, N., editors, *Predicting the future and understanding the past*, pages 243–264. Addison-Wesley, Redwood City, CA.

Mozer, M. C. and Smolensky, P. (1989). Skeletonization: A technique for trimming the fat from a network via relabance assessment. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 107–115. Morgan Kaufmann.

Neto, J., Almeida, L., Hochberg, M., Martins, C., Nunes, L., Renals, S., and Robinson, T. (1995). Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system. In *Proc. of EUROSPEECH*, pages 2171–2174, Madrid.

Neukirchen, C. and Rigoll, G. (1997). Advanced training methods and new network topologies for hybrid MMI-connectionist/HMM speech recognition systems. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3257–3260, Munich, Germany.

Niles, L. and Silverman, H. (1990). Combining hidden Markov models and neural network classifiers. In *International Conference on Acoustics, Speech and Signal Processing*, pages 417–420, Albuquerque, NM.

Niranjan, M. and Fallside, F. (1990). Neural networks and radial basis functions in classifying static speech patterns. *Computer Speech and Language*, 4:275–289.

Nowlan, S. J. (1991). *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures.* CMU-CS-91-126, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Nowlan, S. J. and Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4(4):473–493.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273.

Oja, E. (1989). Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1:61–68.

Oja, E. and Karhunen, J. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84.

Pao, Y. H. (1989). *Adaptive Pattern Recognition and Neural Networks.* Addison-Wesley.

Pearlmutter, B. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269.

Pican, N., Fohr, D., and Mari, J.-F. (1996). HMMs and OWE neural network for continuous speech recognition. In *Proc. of ICSLP*, pages 1309–1312, Philadelphia.

Pican, N., Mari, J.-F., and Fohr, D. (1997). Continuous speech recognition using a context sensitive ANN and HMMs. In *Proc. of EUROSPEECH*, volume 1, pages 95–98, Rhodi.

Pineda, F. (1989). Recurrent back-propagation and the dynamical approach to adaptive neural computation. *Neural Computation*, 1:161–172.

Plaut, D., Nowlan, S., and Hinton, G. (1986). Experiments on learning by back-propagation. Technical Report CMU–CS–86–126, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Poggio, T. and Girosi, F. (1989). A theory of networks for approximation and learning. Technical Report 1140, MIT AI Laboratory, Cambridge, MA.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497.

Polak, E. and Ribiere, G. (1969). Note sur la convergence de methods de directions conjures. *Revue Francaise Information Recherche Operationnelle*, 16:35–43.

Powell, M. J. D. (1987). Radial basis functions for multivariable interpolation: a review. In Mason, J. C. and Cox, M. G., editors, *Algorithms for Approximation: IMA-1985 Conference*. Clarendon Press, Oxford.

Rabiner, L. and Juang, B. (1986). An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 257–285.

Rabiner, L. and Juang, B. H. (1993). *Fundamentals of Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, USA.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, L. R., Juang, B. H., and Lee, C. H. (1996). An overview of automatic speech recognition. In Lee, C. H., Soong, F. K., and Paliwal, K. K., editors, *Automatic Speech and Speaker Recognition: Advanced Topics*, Boston, USA. Kluwer Academic Publishers.

Reichl, W. and Ruske, G. (1995). A hybrid RBF-HMM system for continuous speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3335–3338, Detroit.

Renals, S., Morgan, N., Bourlard, H., Cohen, M., and Franco, H. (1994). Connectionist probability estimators in HMM speech recognition. *IEEE Trans. on Speech and Audio Processing*, 2(1):161–1174.

Rigoll, G. (1994). Maximum mutual information neural networks for hybrid connectionist-HMM speech recognition systems. *IEEE Trans. on Speech and Audio Processing*, 2(1):175–1184.

Rigoll, G., Neukirchen, C., and Rottland, J. (1995). Large vocabulary speaker-independent continuous speech recognition with a new hybrid system based on MMI-neural networks. In *Proc. of EUROSPEECH*, pages 1659–1662, Madrid.

Riis, S. K. and Krogh, A. (1997). Hidden neural networks: a framework for HMM/NN hybrids. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3233–3236, Munich, Germany.

Robinson, A. and Fallside, F. (1988). Static and dynamic error propagation networks with application to speech coding. In Anderson, D., editor, *Neural Information Processing Systems*, pages 632–641, Denver, CO. American Institute of Physics, New York.

Robinson, T. (1992). A real-time recurrent error propagation network word recognition system. In *International Conference on Acoustics, Speech and Signal Processing*, volume I, pages 617–620.

Robinson, T. (1994). An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305.

Robinson, T., Almeida, L., Boite, J. M., Bourlard, H., Fallside, F., Hochberg, M., Kershaw, D., Kohn, P., Konig, Y., Morgan, N., Neto, J. P., Renals, S., Saerens, M., and Wooters, C. (1993). A neural network based, speaker independent, large vocabulary, continuous speech recognition system: The wernicke project. In *Eurospeech93*, Berlin, Germany.

Robinson, T. and Fallside, F. (1991). A recurrent error propagation network speech recognition system. *Computer Speech and Language*, 5(3):259–274.

Rottland, J., Neukirchen, C., Willett, D., and Rigoll, G. (1997). Large vocabulary speech recognition with context dependent MMI-connectionist/HMM systems using the WSJ database1. In *Proc. of EUROSPEECH*, volume 1, pages 79–82, Rhodes.

Rumelhart, D., Hinton, G., and Williams, R. (1986a). Learning internal representations by error propagation. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge.

Rumelhart, D., Hinton, G., and Williams, R. (1986b). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Sanger, T. (1989). An optimality principle for unsupervised learning. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 1*, pages 11–19, Denver, CO. Morgan Kaufmann, San Mateo.

Sato, M. (1990). A real time learning algorithm for recurrent analog neural networks. *Biological Cybernetics*, 62:237–241.

Sejnowski, T. and Rosenberg, C. (1987). Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168.

Sejnowski, T. J. and Rosenberg, C. R. (1986). *NETtalk: A parallel network that learns to read aloud.* Technical report 86-01, Department of Electrical Engineering and Computer Science, Johns Hopkins University, Baltimore, MD.

Singer, E. and Lippmann, R. P. (1992). A speech recognizer using radial basis function neural networks in an HMM framework. In *International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 629–632, San Franscisco.

Takahashi, J. and Sagayama, S. (1994). Telephone line characteristic adaptation using vector field smoothing technique. In *Proc. of ICSLP*, pages 991–994, Yokohama.

Tebelskis, J., Waibel, A., Petek, B., and Schmidbauer, O. (1991). Continuous speech recognition using linked predictive networks. In Lippman, R. P., Moody, R., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 199–205, Denver, CO. Morgan Kaufmann, San Mateo.

Trentin, E. (1995). Rejection and the equal error rate: Principles and a case study. IRST Technical Report 9507-06, IRST, Povo (TN), ITALY.

Trentin, E. (1999a). Activation functions with learnable amplitude. In *Proceedings of IJCNN99, International Joint Conference on Neural Networks*, Washington, DC.

Trentin, E. (1999b). Learning the amplitude of activation functions in layered networks. In Marinaro, M. and Tagliaferri, R., editors, *Neural Nets - WIRN Vietri - 98*, pages 138 – 144, Berlin, Germany. Springer-Verlag.

Trentin, E., Bengio, Y., Furlanello, C., and De Mori, R. (1998). Neural networks for speech recognition. In Mori, R. D., editor, *Spoken Dialogues with Computers*, pages 311–361, London, UK. Academic Press.

Trentin, E., Brugnara, F., Bengio, Y., Furlanello, C., and De Mori, R. (1996a). Statistical and neural network models for speech recognition. IRST Technical Report 9610-08, IRST, Povo (TN). To appear in R. Daniloff (Ed.) Connectionist accounts of clinical and normal language, Hillsdale, N.J.: Erlbaum, 2001 (forthcoming).

Trentin, E. and Cattoni, R. (1997). An hybrid HMM/recurrent neural networks approach to indoor robot navigation. In *Proceedings of RWC97 - Real World Computing Symposium*, Tokyo, Japan.

Trentin, E. and Cattoni, R. (1999a). A hybrid framework for indoor robot navigation. In Marinaro, M. and Tagliaferri, R., editors, *Neural Nets - WIRN Vietri - 98*, pages 255 – 263, Berlin, Germany. Springer-Verlag.

Trentin, E. and Cattoni, R. (1999b). Learning perception for indoor robot navigation with a hybrid HMM/Recurrent neural networks approach. *Connection Science*, 11(3/4):243–265. Special Issue on *Adaptive Robots*.

Trentin, E., Furlanello, C., and Falavigna, D. (1995). An evaluation criterion for connectionist systems in classification tasks. In *Proceedings of First Workshop on Evaluation criteria of neural nets efficiency in industrial applications*, IIASS, Vietri sul Mare (SA).

Trentin, E. and Giuliani, D. (1997). Speaker normalization with a mixture of recurrent networks. In *Proceedings of ESANN97, European Symposium on Artificial Neural Networks*, Bruges, Belgium.

Trentin, E. and Giuliani, D. (2001). A mixture of recurrent neural networks for speaker normalization. *Neural Computing & Applications*. Accepted for publication (Forthcoming).

Trentin, E., Giuliani, D., and Furlanello, C. (1996b). Spectral mapping: A comparison of connectionist approaches. In Marinaro, M. and Tagliaferri, R., editors, *Neural Nets. WIRN Vietri-96*, pages 270–277, Berlin. Springer - Verlag.

Trentin, E. and Gori, M. (1999a). Combining neural networks and hidden Markov models for speech recognition. In Marinaro, M. and Tagliaferri, R., editors, *Neural Nets - WIRN Vietri - 98*, pages 63 – 79, Berlin, Germany. Springer-Verlag.

Trentin, E. and Gori, M. (1999b). A tutorial on connectionist and hybrid HMM/connectionist systems for speech recognition. In Kasabov, N., editor, *IJCNN99 Tutorials Track 8 (Book on CD-ROM, Chapter 3): Speech and Language Processing*, Dunedin, New Zealand. University of Otago.

Trentin, E. and Gori, M. (2000). A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*. Accepted for publication (forthcoming). Also available as IRST Technical Report Ref. No. 9803-03.

Trentin, E. and Matassoni, M. (1999). Robust segmental-connectionist learning for recognition of noisy speech. In *Proceedings of the first Workshop on Robust Methods for Speech Recognition in Adverse Conditions*, pages 159 – 162, Tampere, Finland.

Trentin, E. and Matassoni, M. (2000). The regularized SNN-TA model for recognition of noisy speech. In *Proceedings of IJCNN2000 (International Joint Conference on Neural Networks)*, pages V 97–102, Como (Italy).

Waibel, A. (1989). Modular construction of time-delay neural networks for speech recognition. *Neural Computation*, 1:39–46.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. (1989a). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:328–339.

Waibel, A., Sawai, H., and Shikano, K. (1989b). Modularity and scaling in large phonemic neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37:1888–1898.

Werbos, P. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.

Werbos, P. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1:339–356.

Wilinski, P., Solaiman, B., Hillion, A., and Czamecki, W. (1998). Toward the border between neural and markovian paradigms. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(2):146–159.

Williams, R. and Zipser, D. (1989a). Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1:87–111.

Williams, R. and Zipser, D. (1989b). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.

Winters, J. and Rose, C. (1989). Minimum distance automata in parallel networks for optimum classification. *Neural Networks*, 2:127–132.

Yamada, T. and Yabuta, T. (1992). Neural network controller using autotuning method for nonlinear functions. *IEEE Transactions on Neural Networks*, 3:595–601.

Yan, Y., Fanty, M., and Cole, R. (1997). Speech recognition using neural networks with forward-backward probability generated targets. In *International Conference on Acoustics, Speech and Signal Processing*, pages 3241–3244, Munich, Germany.

Yu, D., Huang, T., and Chen, D. W. (1994). A multi-state NN/HMM hybrid method for high performance speech recognition. In *Proc. of ICSLP*, pages 1503–1506, Yokohama.

Zavaliagkos, G., Austin, S., Makhoul, J., and Schwartz, R. (1993). A hybrid continuous speech recognition system using segmental neural nets with hidden Markov models. *Int. Journal of Pattern Recognition and Artificial Intelligence*, pages 305–319. Special Issue on Applications of Neural Networks to Pattern Recognition (I. Guyon Ed.).

Zavaliagkos, G., Schwartz, R., and Makhoul, J. (1995). Batch, incremental and instantaneus adaptation techniques for speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, pages I–676–679, Detroit.

Zavaliagkos, G., Zhao, Y., Schwartz, R., and Makhoul, J. (1994). A hybrid segmental neural net/hidden Markov model system for continuous speech recognition. *IEEE Trans. on Speech and Audio Processing*, 2(1):151–160.