Applying Divide and Conquer to Large Scale Pattern Recognition Tasks

Jürgen Fritsch, Michael Finke
fritsch@ira.uka.de, finkem@cs.cmu.edu

Interactive Systems Laboratories

University of Karlsruhe
Am Fasanengarten 5
76128 Karlsruhe, Germany

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA

Abstract. Rather than presenting a specific trick, this paper aims at providing a methodology for large scale, real-world classification tasks involving thousands of classes and millions of training patterns. Such problems arise in speech recognition, handwriting recognition and speaker or writer identification, just to name a few. Given the typically very large number of classes to be distinguished, many approaches focus on parametric methods to independently estimate class conditional likelihoods. In contrast, we demonstrate how the principles of modularity and hierarchy can be applied to directly estimate posterior class probabilities in a connectionist framework. Apart from offering better discrimination capability, we argue that a hierarchical classification scheme is crucial in tackling the above mentioned problems. Furthermore, we discuss training issues that have to be addressed when an almost infinite amount of training data is available.

1 Introduction

The majority of contributions in the field of neural computation deal with relatively small datasets and, in case of classification tasks, with a relatively small number of classes to be distinguished. Representatives of such problems include the UCI machine learning database [16] and the Proben [20] benchmark set for learning algorithms. Research concentrates on aspects such as missing data, model selection, regularization, overfitting vs. generalization and the bias/variance trade-off. Over the years, many methods and 'tricks' have been developed to optimally learn and generalize when only a limited amount of data is available.

On the other hand, many problems in human computer interaction (HCI) such as speech and handwriting recognition, lipreading and speaker and writer identification require comparably large training databases and also often exhibit a large number of classes to be discriminated, such as (context-dependent) phones, letters and individual speakers or writers. For example, in state-of-the-art large vocabulary continuous speech recognition, we are typically faced with an inventory of several thousand basic acoustic units and training databases consisting of several millions of preprocessed speech patterns. There is only a

limited amount of publications available on the sometimes very different problems concerning the choice of learning machines and training algorithms for such tasks and datasets.

This article addresses exactly the latter kind of learning tasks and provides a principled approach to large scale classification problems, exemplifying it on the problem of connectionist speech recognition. Our approach is grounded on the powerful *divide and conquer* paradigm that traditionally has always been applied to problems of rather large size. We argue that a hierarchical approach that modularizes classification tasks is crucial in applying statistical estimators such as artificial neural networks. In that respect, this paper presents not just a single 'trick of the trade', it offers a methodology for large scale classification tasks. Such tasks have traditionally been addressed by building generative models rather than focussing on the prediction of posteriors without making strong assumptions on the distribution of the input.

The remainder of the paper is organized as follows. Section 2 presents the general approach to soft hierarchical classification. Section 3 then discusses methods to design the topology of hierarchical classifiers - a task that is of increasing importance when dealing with large numbers of classes. Finally, section 4 demonstrates in detail the application of hierarchical classification to connectionist statistical speech recognition. Section 5 concludes this paper with a summary.

2 Hierarchical Classification

Consider the task of classifying patterns $\mathbf{x}$ as belonging to one of N classes ω_k. Given that we have access to the class conditional probability densities $p(\mathbf{x}|\omega_k)$, Bayes theory states that the optimal decision should be based on the a-posteriori probabilities

$$p(\omega_k|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_k)p(\omega_k)}{\sum_i p(\mathbf{x}|\omega_i)p(\omega_i)}.$$

Given that equal risks are associated with all possible misclassifications, the optimal decision is to choose the class with maximum a-posteriori probability given a specific pattern $\mathbf{x}$. Two distinct approaches have to be considered when applying Bayes theory to a learning from examples task with generally unknown distributions. In the first approach, one tries to estimate class-conditional likelihoods $p(\mathbf{x}|\omega_k)$ and prior probabilities $p(\omega_k)$ from a labeled dataset which are then used to calculate posterior probabilities according to Bayes rule. In principle, this approach can be applied to tasks with an arbitrary large number of classes since the class-conditional likelihoods can be estimated independently. However, such an approach focuses on the modeling of the class-conditional densities. For classification accuracy however, it is more important to model class boundaries.

The second approach accomodates this perspective by directly estimating posterior class probabilities from datasets. It was shown (e. g. [6]) that a large

class of artificial neural networks such as multi-layer perceptrons and recurrent neural networks can be trained to approximate posterior class probabilities. The degree of accuracy of the approximation however depends on many factors, among them the plasticity of the network. Comparing the two approaches, the discriminative power of methods that estimate posterior probabilities directly is generally higher, resulting in better classification accuracy especially when the class-conditional distributions are very complex. This fact (among others) explains the success and popularity of neural network classifiers on many learning from examples tasks.

However, when the number of classes to be distinguished increases to say several thousand, neural network estimators of posterior probabilities fail to provide good approximations mainly because of two reasons: First, real-world problems involving such a large number of classes often exhibit an extremely non-uniform distribution of priors [28]. Many learning algorithms for neural networks (especially stochastic on-line gradient descent) have difficulties with non-uniformly distributed classes. Particularly the distribution of posteriors of infrequent classes tend to be approximated poorly. Second, and more important, one of the prerequisites for training neural networks to estimate posteriors, the 1-out-of-N coding of training targets, implies that the number of output neurons matches the number of classes. It is unfeasible to train a neural network with thousands of output neurons. Also, with increasing number of classes, the complexity of the optimum discriminant functions also increases and the potential for conflicts between classes grows. Thus, from our point of view, typical monolithic neural network classifiers are not applicable because of their limitation to tasks with relatively few classes.

2.1 Decomposition of Posterior Probabilities

Applying the principle of *divide and conquer*, we can break down the task of discriminating between thousands of classes into a hierarchical structure of many smaller classification tasks of controlled size. This idea underlies the approaches to decision tree architectures [5, 21, 23]. Decision trees classify input patterns by asking categorical questions at each internal node. Depending on the answer to these questions a single path is followed to one of the child nodes and the process repeats until a leaf node is reached and a (winner) class label is emitted. Therefore, decision tree classifiers can only supply us with hard decisions. No information about the confusability of a specific input pattern is given to us. Rather, we are often interested in the posterior class probabilities because we wish to have a measure of the ambigueness of a decision. Furthermore, we are sometimes required to feed a measure of the degree of membership for all potential classes into a superior decision making process. As we will see in section 4, statistical speech recognition is a typical example for the latter scenario.

Adhering to the *divide and conquer* approach but generalizing the decision tree framework, the statistical method of factoring posteriors can be applied to design *soft* classification trees [24, 25]. For now, we assume, that optimal posterior probabilities are available. Let S be a (possibly large) set of classes ω_k

to be discriminated. Consider we have a method at our disposition which gives us a partitioning of S into M disjoint and non-empty subsets S_i such that members of S_i are almost never confused with members of S_j $(\forall j \neq i)$. A particular class ω_k will now be a member of S and exactly one of the subsets S_i. Therefore, we can rewrite the posterior probability of class ω_k as a joint probability of the class and the corresponding subset S_i and factor it according to

$$\begin{aligned} p(\omega_k|\mathbf{x}) &= p(\omega_k, S_i|\mathbf{x}) \qquad \text{with} \qquad \omega_k \in S_i \\ &= p(S_i|\mathbf{x})\ p(\omega_k|S_i, \mathbf{x}). \end{aligned}$$

Thus, the global task of discriminating between all the classes in S has been converted into (1) discriminating between subsets S_i and (2) independently discriminating between the classes ω_k remaining within each of the subsets S_i. Recursively repeating this process yields a hierarchical tree-organized structure (Fig. 1).

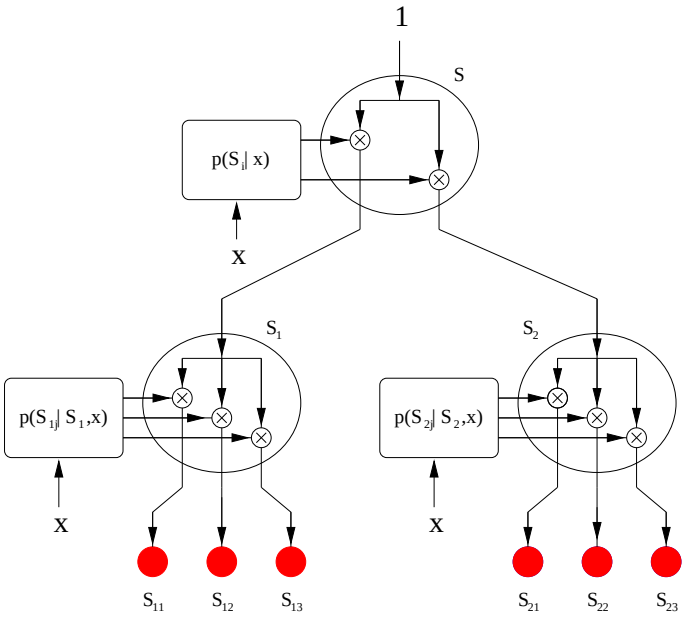

Fig. 1. Hierarchical decomposition of posteriors

Note, that the number of subclasses S_i of each node does not need to be constant throughout the classifier tree and might be subject to optimization during the tree design phase. In order to compute the posterior probability for a specific class, we have to follow the path from root node to the leaf corresponding to the class in question, multiplying all the conditional posteriors along the way. Both the design of the tree structure (*divide*) and the estimation and multiplication (*conquer*) of conditional posteriors at each node are important aspects in this architecture, that have to be considered thoroughly because in practice, only approximations to the conditional posteriors are available.

2.2 Hierarchical Interpretation

The presented architecture can be interpreted as a probability mass distribution device. At the root node, an initial probability mass of 1 is fed into the architecture. At each node, the incoming probability mass is multiplied by the respective conditional posterior probabilities and fed into the child nodes. Eventually, the probability mass is distributed among all the leaves (classes) rendering their posterior probabilities. In contrast, classifier trees are mostly used as hard-switching devices, where only a single path from root node to one of the leaves is taken.

A hierarchical decomposition of posterior probabilities through a soft classification tree offers several advantages. If one of the nodes in the tree, for example the root node fails to provide good estimates of conditional posteriors, a hard decision tree will produce many classification errors. In a soft classification tree, such shortcomings will influence the decision process less dramatically. Also, recovery from errors is often possible through a superior decision process.

Another aspect of soft classification trees that can be exploited for various purposes is the sum-to-unity property observable in any horizontal cross-section at any level of the tree. The tree can be cut off at a certain level and still be used as a soft classification tree that computes posterior class probabilities. This is equivalent to creating a new (smaller) set of classes by clustering and merging the original classes according to the tree topology. In general, the resulting classification task will be easier to solve than the original one.

Related to the sum-to-unity property of cross-sections is that the partial posteriors computed on a path from the root node to a leaf are decreasing monotonically. This in turn allows to close paths whenever a suitable threshold is reached, pruning whole subtrees with classes that would otherwise receive posteriors smaller than the threshold. This property yields the possibility to smoothly trade off classification accuracy against computational complexity. In the limit, when only a single path with highest conditional posterior is followed, the soft classification tree transmutes into a hard decision tree.

2.3 Estimation of Conditional Node Posteriors

Given a hierarchical decomposition of posterior class probabilities, it remains to instantiate the tree nodes with estimators for the required conditional posteriors. Conditioning a posterior on a subset of classes S_i can be accomplished by restricting the training set of the corresponding learning device to the patterns with a class label from S_i. According to this setting, the available training data in each node is distributed among all its child nodes according to the class partitioning. While the root node receives all available training data, nodes further down the tree receive less data than their predecessors. On the other hand, specialization increases from root node to leaves. This fact has important consequences on learning speed and model selection when training whole hierarchies.

One of the important issues in hierarchical decompositions of posterior probabilities are the unavoidable inaccuracies of practical estimators for the conditional posteriors that have to be provided in each tree node. Neural networks

can only be trained to *approximate* the true distribution of posterior class probabilities and the degree of accuracy depends on both the inherent difficulty of the task as given by the training set and the network structure and training schedule being used. Inaccurate approximations to the true distribution of posteriors hurt most in the upper layers of a classification tree - a fact that has to be taken into account by tree design procedures, which we will discuss next.

3 Classifier Tree Design

When it comes to the design of soft classifier trees, or equivalently to the design of hierarchical decompositions of class posteriors, the choice of algorithm depends mostly on the number of initial classes. We will first discuss optimal tree structures before we will turn to heuristic design algorithms necessary when dealing with the large number of classes that we have to deal with.

3.1 Optimality

The optimal soft classification tree for a given task and given type and structure of estimators for the conditional node posteriors is the one which results in minimum classification error in the Bayes setting. If all the node classifiers would compute the true conditional posteriors, the tree structure would have no influence on the classifier performance because any kind of factoring (through any kind of tree structure) yields an *exact* decomposition of the class posteriors. However, in practice, approximation errors of node classifiers render the choice of tree structure an important issue. For small numbers of classes, the optimal tree can in principle be found by exhaustively training and testing all possible partitionings for a particular node (starting with the root node) and chosing the one that gives the highest recognition accuracy. However, even if restricting the tree structure to binary branching nodes and balanced partitionings, the number K of partitionings that have to be examined at the root node

$$K = \frac{N!}{(\frac{N}{2}!)^2}$$

quickly brings this algorithm to its limits, even for a moderate number of classes N. Therefore, we have to consider heuristics to derive near optimal tree structures. For example, one valid possibility is to assume that the accuracy of achievable approximations to the true posteriors is related to the separability of the corresponding sets of classes.

3.2 Prior Knowledge

Following the above mentioned guideline, prior knowledge about the task in question can often be applied to hierarchically partition the global set of classes into reasonable subsets. The goal is to partition the remaining set of classes in

a way that intuitively maximizes the separability of the subsets. For example, given a set of phones in a speech recognizer, a reasonable first partitioning would be to build subsets consisting of voiced and unvoiced phones. In larger speech recognition systems where we have to distinguish among multi-state context-dependent phones, prior knowledge such as state and context identity can be used as splitting criterion. In tasks such as speaker or writer identification, features such as gender or age are potential candidates for splitting criteria.

The advantage of such knowledge driven decompositions is a fast tree design phase which is a clear superiority of this approach when dealing with large numbers of classes. However, this method for the design of hierarchical classifiers is subjective and error prone. Two experts in a specific field might disagree strongly on what constitutes a reasonable hierarchy. Furthermore, it is not always the case that *reasonable* partitionings yield good separability of subsets. Expert knowledge can be misleading.

3.3 Confusion Matrices

In case the number of classes is small enough to allow the training of a single classifier, the design of a soft classifier tree can be based on the confusion matrix of the trained monolithic classifier. Indicating the confusability of each pair of classes, the confusion matrix yields relatively good measures of the separability of pairs of classes. This information can be exploited for designing a tree structure using a clustering algorithm. For instance, we can define the following (symmetric) distance measure between two disjunct sets of classes S_k and S_l

$$d(S_k, S_l) = - \sum_{\omega_i \in S_k} \sum_{\omega_j \in S_l} C(\omega_i, \omega_j | \mathcal{T}) + C(\omega_j, \omega_i | \mathcal{T})$$

where $C(\omega_i, \omega_j | \mathcal{T})$ denotes the number of times class ω_i is confused with class ω_j as measured on a set of labeled patterns $\mathcal{T}$. The distance $d(S_k, S_l)$ can now be used as a replacement for the usual Euclidean distance measure in a standard bottom-up clustering algorithm. Unfortunately, once the number of classes increases to several thousand, training of a monolithic classifier becomes increasingly difficult.

3.4 Agglomerative Clustering

Assuming that separability of classes correlates with approximation accuracy of estimators for the posterior class probabilities, we can go further and assume that separability of classes can be measured by a suitable distance between the class conditional distributions in feature space. We already introduced such a distance measure in form of the elements of a class confusion matrix. Other, computationally less expensive distance measures would be the Euclidean distance between class means or the Mahalanobis distance between the classes second order statistics. Irrespective of the chosen distance measure, the goal always is to group the set of classes in a way that results in maximum inter- and minimum intra-group

distances. Solutions to this problem are known as agglomerative clustering algorithms and a large pool of variations of the basic algorithm is available in the literature [7].

4 Application to Speech Recognition

In this section, we will demonstrate the main ideas and benefits of the hierarchical classifier approach on the task of large vocabulary continuous speech recognition (LVCSR). More specifically, we will focus on acoustic modeling for statistical speech recognition using hidden Markov models (HMM) [27]. To give an impression of the complexity of such a task: training databases typically consist of tens of millions of speech patterns, the number of acoustic classes being distinguished ranges from ca. 50 (monophones) to over 20000 (context-dependent polyphones).

4.1 Statistical Speech Recognition

The basic statistical entity in HMM based speech recognition is the posterior probability of word sequences $W_1, \ldots, W_N$ given a sequence of acoustic observations $\mathbf{X_1}, \ldots, \mathbf{X_M}$ and a set of model parameters Θ

$$P(W_1, \ldots, W_N | \mathbf{X_1}, \ldots, \mathbf{X_M}, \Theta)$$

During training, we are seeking parameters Θ that maximize this probability on the training data

$$\hat{\Theta} = \arg\max_{\Theta} \prod_{t=1}^{T} P(W_1, \ldots, W_{N(t)} | \mathbf{X_1}, \ldots, \mathbf{X_{M(t)}}, \Theta)$$

and during recognition, we want to find the sequence of words that maximizes this probability for a given acoustic observation and fixed model parameters Θ

$$\hat{W_1}, \ldots, \hat{W_N} = \arg\max_{W_1, \ldots, W_N} P(W_1, \ldots, W_N | \mathbf{X_1}, \ldots, \mathbf{X_M}, \Theta)$$

In order to simplify the process of maximizing the posterior probability of word sequences, Bayes rule is usually applied

$$P(W_1, \ldots, W_N | \mathbf{X_1}, \ldots, \mathbf{X_M}) = \frac{P(\mathbf{X_1}, \ldots, \mathbf{X_M} | W_1, \ldots, W_N)\ P(W_1, \ldots, W_N)}{P(\mathbf{X_1}, \ldots, \mathbf{X_M})}$$

This rule separates the estimation process into the so called *acoustic model (AM)* consisting of terms that depend on the acoustic observations $\mathbf{X_1}, \ldots, \mathbf{X_M}$ and the *language model (LM)* consisting of terms that depend only on the sequence of words $W_1, \ldots, W_N$. In this paper we will focus on acoustic modeling using connectionist estimators as a typical example of a task involving the discrimination of thousands of classes. For a review on other important aspects of

LVCSR such as pronunciation modeling, language modeling and decoding algorithms we refer the reader to [27].

The task of acoustic modeling (ignoring the denominator) is to estimate parameters $\Theta^{\mathbf{AM}}$ which maximize

$$P(\mathbf{X_1}, \ldots, \mathbf{X_M} | W_1, \ldots, W_N, \Theta^{\mathbf{AM}}).$$

Words W_i are modeled as sequences (or graphs) of phone models. The mapping from words to phone models is usually accomplished by means of a pronunciation dictionary. Phone models in turn are usually modeled as m-state left-to-right hidden Markov models (HMM) to capture the temporal and acoustic variability of speech signals. The following figure shows the process of converting a sequence of words into (1) a pronunciation graph (possibly with pronunciation variants) and (2) an HMM state graph.

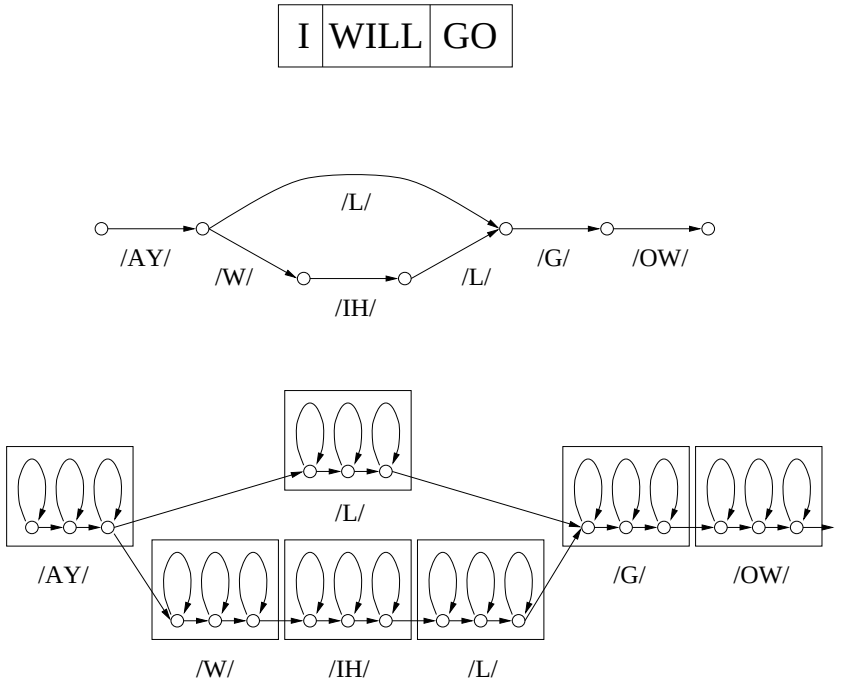

Fig. 2. Typical hidden Markov model in speech recognition

In this framework, where word sequences are represented as directed acyclic graphs of HMM states, the likelihood of an acoustic observation can be rewritten as

$$P(\mathbf{X_1}, \ldots, \mathbf{X_M}|W_1, \ldots, W_N) = \sum_{s_1,\ldots,s_M} P(\mathbf{X_1}, \ldots, \mathbf{X_M}|s_1, \ldots, s_M)\; p(s_1, \ldots, s_M)$$

where the summation extends over all possible state sequences $s_1, \ldots, s_M$ in the HMM model for the word sequence $W_1, \ldots, W_N$. In the Viterbi approximation, the above likelihood is approximated by the probability of the most likely state sequence

$$P(\mathbf{X_1}, \ldots, \mathbf{X_M}|W_1, \ldots, W_N) \approx \max_{s_1,\ldots,s_M} P(\mathbf{X_1}, \ldots, \mathbf{X_M}|s_1, \ldots, s_M)\; p(s_1, \ldots, s_M).$$

Given a specific state sequence, the likelihood of the acoustic observations given that sequence can be factored as follows

$$P(\mathbf{X_1}, \ldots, \mathbf{X_M}|s_1, \ldots, s_M) \approx \prod_{i=1}^{M} p(\mathbf{X_i}|X_1, \ldots, X_{i-1}, s_1, \ldots, s_M)\; p(s_1, \ldots, s_M).$$

In the application of first-order hidden Markov models to the estimation of such likelihoods one usually makes two simplifying assumptions:

– Independence of Observations:

$$P(\mathbf{X_1}, \ldots, \mathbf{X_M}|s_1, \ldots, s_M) \approx \prod_{i=1}^{M} p(\mathbf{X_i}|s_1, \ldots, s_M)\; p(s_1, \ldots, s_M)$$

– First-order Assumption:

$$P(\mathbf{X_1}, \ldots, \mathbf{X_M}|s_1, \ldots, s_M) \approx \prod_{i=1}^{M} p(\mathbf{X_i}|s_i)\; p(s_i|s_{i-1})$$

4.2 Emission and Transition Modeling

Mainstream LVCSR systems follow the above approach by modeling emission probability distributions $p(\mathbf{X_i}|s_i)$ and transition probabilities $p(s_i|s_{i-1})$ separately and independently. Emission probability distributions are usually modeled using mixture densities from the exponential family, such as the mixture of Gaussians

$$p(\mathbf{X_i}|s_i) = \sum_{k=1}^{n} \gamma_k N_k\,(\mathbf{X_i}|s_i)$$

where the γ_k denote *mixture coefficients* and the N_k *mixture component densities* (here: normal distributions). Transition probabilities on the other hand are modeled by simple multinomial probabilities since they are conditioned on a discrete variable only (not on the input vector).

The advantage of this approach is a decoupled estimation process that separates temporal and acoustic modeling. As such, it allows to easily vary HMM state topologies after training in order to modify temporal behaviour. For instance, state duplication is a popular technique to increase the minimum duration constraint in phone models. Having separated emission and transition probability estimation, state duplication consists of simply sharing acoustic models among multiple states and adapting the transition probabilities.

However, the disadvantage of the above approach is a mismatch in the dynamic range of emission and transition probabilities. The reason is that transition probabilities are modeled separately as multinomial probabilities, constrained by the requirement to sum to one. This leads to a dominant role of emission probabilities with transition probabilities hardly influencing overall system performance.

4.3 Phonetic Context Modeling

So far we have assumed that only one HMM is required per modeled monophone (see Fig. 2). Since the English language can be modeled by approximately 45 monophones, one might get the impression that only that number of HMM models need to be trained. In practice however, one observes an effect called coarticulation that causes large variations in the way specific monophones actually sound, depending on their phonetic context.

Usually, explicit modeling of phones in phonetic context yields great gains in recognition accuracy. However, it is not immediately clear how to achieve robust context-dependent modeling. Consider, for example, so called *triphone* models. A triphone essentially represents the realization of a specific monophone in a specific context spanning one phone to the left and right. Assuming an inventory of 45 monophones, the number of (theoretically) possible triphones is $45^3 = 91125$. Many of these triphones will occur rarely or never in actual speech due to the linguistic constraints in the language. Using triphones therefore results in a system which has too many parameters to train. To avoid this problem, one has to introduce a mechanism for sharing parameters across different triphone models.

Typically, a CART like decision tree is adopted to cluster triphones into *generalized triphones* based on both their a-priori probability and their acoustic similarity. Such a top-down clustering requires the specification of viable attributes to be used as questions on phonetic context in order to split tree nodes. Mostly, linguistic classes such as vowel, consonant, fricative, plosive, etc. are being employed. Furthermore, one can generalize triphones to *polyphones* by allowing dependence on a wider context (and not just the immediate left and right neighboring phones). Fig. 3 shows a typical decision tree for the clustering of polyphonic variations of a particular monophone state.

The collection of all leaf nodes of decision trees for each monophone state in a given system represents a robust and general set of context-dependent subphonetic units. Since each of these units corresponds to several triphone HMM states, they are often called *tied states*. Typically, a large vocabulary continuous

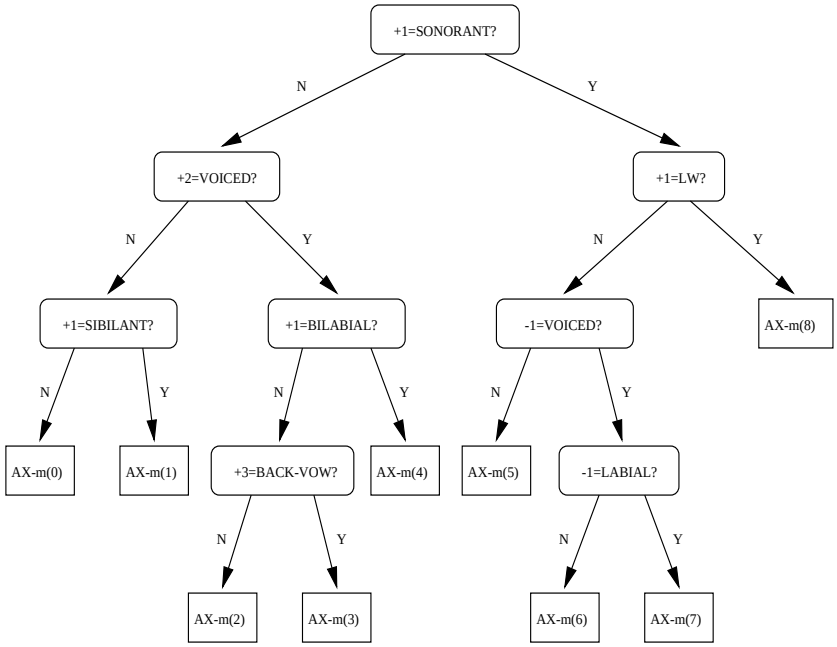

Fig. 3. Phonetic Context Modeling using Decision Trees. Shown is a decision tree modeling phonetic contexts of middle state (3-state HMM) of monophone /AX/.

speech recognizer models between 3000 and 24000 such tied states. Mainstream LVCSR systems scale to any number of context-dependent modeling units since emission and transition models are independently estimated for each tied state.

4.4 Connectionist Acoustic Modeling

Locally discriminant connectionist acoustic modeling is the most popular approach to integrate neural networks into an HMM framework [3, 4, 18]. It is based on converting estimates of local posterior class probabilities to scaled likelihoods using Bayes rule. These scaled likelihoods can then be used as observation probability estimates in standard HMMs. For a moderately small number N of HMM states, a neural network can be trained to jointly estimate posterior probabilities $p(s_i|\mathbf{X_i})$ for each state s_i given an input vector $\mathbf{X_i}$. Bayes rule yields the corresponding scaled [1] class conditional likelihoods

$$\hat{p}(\mathbf{X_i}|s_i) = \frac{p(s_i|\mathbf{X_i})}{p(s_i)}.$$

[1] The missing additional term consisting of the probability of the input vector $p(\mathbf{X_i})$ is usually omitted because it is independent of the class/state identity and therefore does not influence a Viterbi style search for the most likely state sequence.

While $p(s_i|\mathbf{X_i})$ is estimated using a neural network, prior probabilities $p(s_i)$ can be estimated by relative frequencies as observed in the training data. Several researchers (e. g. [3, 14]) have reported improvements with connectionist acoustic modeling when the technique for the estimation of emission probabilities was the only difference in comparison. Since mainstream HMMs for speech recognizers are mostly trained in a maximum likelihood framework using the Expectation-Maximization (EM) algorithm, incorporation of discriminatively trained neural networks that focus on modeling of class boundaries instead of class distributions is often observed to be beneficial. Also, compared to mixtures of Gaussians based acoustic models, connectionist acoustic models are often reported to achieve the same accuracy with far less parameters.

However, when the number of HMM states is increased to model context-dependent polyphones (triphones,quintphones), a single neural network can no longer be applied to estimate posteriors. It becomes necessary to factor the posterior state probabilities [17] and modularize the process of estimating those posteriors. In most approaches, the posteriors are factored on phonetic context or monophone identity (e.g. [4, 9, 15]). Viewing factoring as a hierarchical decomposition of posteriors, we generalized the approaches to context-dependent connectionist acoustic modeling by introducing a tree structured hierarchy of neural networks (HNN) [12, 13] corresponding to a multi-level factoring of posteriors based on a-priori knowledge such as broad sound classes (silence,noises,phones), phonetic context and HMM state identity. Fig. 4 shows the topology of such a structure.

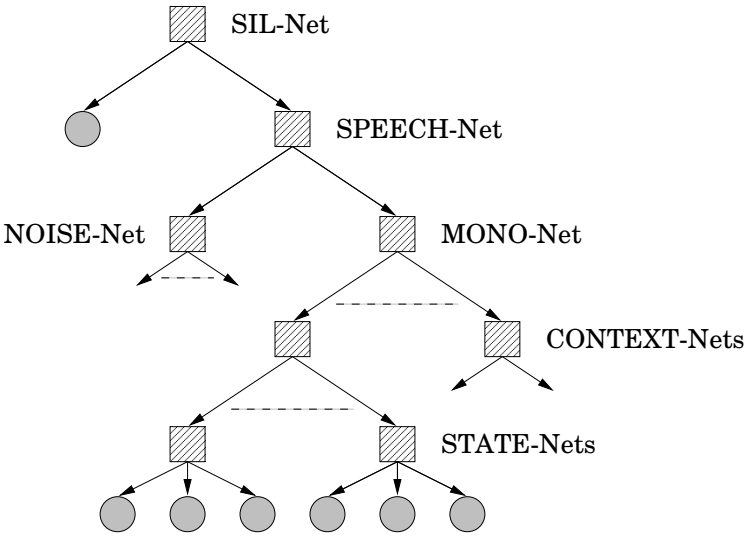

Fig. 4. Topology of a Hierarchy of Neural Networks (HNN) to estimate context-dependent posteriors, factored based on a-priori phonetic knowledge

At the top of this hierarchy, we discriminate silence, noise and speech sounds by means of two networks (SIL-Net, SPEECH-Net). The motivation for this specific partitioning comes from the observation that these three classes are easy to distinguish acoustically. The remainder of the tree structure decomposes the posterior of speech, conditioning on monophone, context and state identity as these are convenient sound classes modeled by any phone based HMM speech recognizer. The hierarchy of Fig. 4 can be decomposed even further, for instance by factoring conditional monophone posteriors (estimated by the MONO-Net) based on linguistic features (e.g. voiced/unvoiced, vowel/consonantal, fricative etc.). The motivation behind such a decomposition is twofold. First, it reduces the number of local classes in each node, improving approximation accuracy and second, it yields a decoupled and specialized set of expert networks having to handle a smaller amount of phonetic variation.

However, as mentioned in section 3, the use of prior knowledge for the design of a hierarchy of neural networks does not take into account dissimilarity of the observed classes in feature space. We therefore developed an agglomerative clustering algorithm to automatically design such hierarchies for the estimation of posteriors for a large number of classes. We termed this framework ACID/HNN [11].

4.5 ACID Clustering

ACID (**A**gglomerative **C**lustering based on **I**nformation **D**ivergence) is a bottom-up clustering algorithm for the design of tree-structured soft classifiers such as a hierarchy of neural networks (HNN) [10, 11]. Although developed for connectionist acoustic modeling, the algorithm can in principle be used for any kind of classification task. Starting from a typically very large set of initial classes, for example the set of decision tree clustered HMM states in a speech recognizer [2], the ACID algorithm constructs a binary hierarchy. The nodes in the resulting tree are then instantiated with estimators for the respective conditional posterior probabilities, for instance in form of an HNN. The clustering metric in the ACID algorithm is the symmetric information divergence [26]

$$d(s_i, s_j) = \int_{\mathbf{x}} (p(\mathbf{x}|s_i) - p(\mathbf{x}|s_j)) \log \frac{p(\mathbf{x}|s_i)}{p(\mathbf{x}|s_j)} \, d\mathbf{x}$$

between class conditional densities of clusters. In contrast to standard agglomerative clustering algorithms which mostly represent clusters by their means and employ the Euclidean distance metric, we chose to represent clusters by parametric mixture densities (mixtures of Gaussians) in the ACID algorithm. Modeling clusters with mixture densities is much more adequate than just using the mean and it still allows to cluster large amounts of classes in a reasonable time. The symmetric information divergence (also called Kullback-Leibler distance) measures the dissimilarity of two distributions and was therefore chosen

[2] In our case, we experimented with up to 24000 initial classes

as the clustering metric. Typically, each initial class (state) is modeled using a single full covariance multivariate Gaussian density

$$p(\mathbf{x}|s_i) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} \exp\{-\frac{1}{2}(\mathbf{x}-\mu_i)^t \Sigma_i^{-1}(\mathbf{x}-\mu_i)\}$$

with mean vector μ_i and covariance matrix Σ_i. Clustering then continuously merges initial classes which corresponds to building mixture densities based on the Gaussians. The symmetric information divergence between two states s_i and s_j with Gaussian distributions amounts to

$$\begin{aligned} d(s_i, s_j) = & \frac{1}{2} tr\{(\Sigma_i - \Sigma_j)(\Sigma_j^{-1} - \Sigma_i^{-1})\} \\ & + \frac{1}{2} tr\{(\Sigma_i^{-1} + \Sigma_j^{-1})(\mu_i - \mu_j)(\mu_i - \mu_j)^t\} \end{aligned}$$

The computation of this distance measure requires $O(n^2)$ multiplications and additions (assuming pre-computed inverse covariance matrices), where n is the dimensionality of the input feature space. To reduce the computational load of the ACID clustering algorithm, one can model the class conditional likelihoods with diagonal covariance matrices only. Feature space transformations such as principal component analysis and linear discriminant analysis can be used to approximate such distributions. When using diagonal covariance Gaussians, the symmetric information divergence simplifies to

$$d(s_i, s_j) = \frac{1}{2} \sum_{k=1}^{n} \frac{(\sigma_{jk}^2 - \sigma_{ik}^2)^2 + (\sigma_{ik}^2 + \sigma_{jk}^2)(\mu_{ik} - \mu_{jk})^2}{\sigma_{ik}^2 \sigma_{jk}^2}$$

where σ_{ik}^2 and μ_{ik} denote the k-th coefficient of the variance and mean vectors of state s_i, respectively. The evaluation of the latter distance measure requires only $O(n)$ multiplications and additions.

Making the simplifying assumption of linearity of information divergence, we can define the following distance measure between clusters of Gaussians S_k and S_l

$$D(S_k, S_l) = \sum_{s_i \in S_k} p(s_i|S_k) \sum_{s_j \in S_l} p(s_j|S_l) d(s_i, s_j)$$

This distance measure is used in the **ACID** clustering algorithm:

1. **Initialize algorithm with N clusters S_i, each containing (1) a parametric model of the class-conditional likelihood and (2) a count C_i, indicating the frequency of class s_i in the training set.**
2. **Compute within cluster priors $p(s_i|S_k)$ for each cluster S_k, using the counts C_i**
3. **Compute the symmetric divergence measure $D(S_k, S_l)$ between all pairs of clusters S_k and S_l.**
4. **Find the pair of clusters with minimum divergence, S_k^* and S_l^***
5. **Create a new cluster $S = S_k^* \bigcup S_l^*$ containing all Gaussians of S_k^* and S_l^* plus their respective class counts. The resulting parametric model is a mixture of Gaussians where the mixture coefficients are the class priors**
6. **Delete clusters S_k^* and S_l^***
7. **While there are at least 2 clusters remaining, continue with 2.**

ACID Initialization Initialization requires to estimate class conditional likelihoods for all (tied) states modeled by the recognizer. The number N of initial classes therefore is determined by other parts of the speech recognizer, namely by the phonetic decision tree that is typically applied to cluster phonetic contexts, or aquivalently to tie HMM states [27]. Initial class conditional densities for these classes can be computed from state alignments using either the Viterbi or the Forward-Backward algorithm on training data and corresponding HMM state graphs generated from training transcriptions. Estimation of initial parametric models for the ACID algorithm therefore requires a single pass through the training data. After initial models have been estimated, the actual ACID clustering does not require any further passes through the training data. Furthermore, note that this algorithm clusters HMM states without knowledge of their phonetic identity solemnly based on acoustic dissimilarity.

ACID Dendrograms For illustration purposes, Fig. 5 shows a dendrogram of a typical ACID clustering run on a relatively small set of only 56 initial classes corresponding to the set of single-state monophone HMMs in a context-independent speech recognizer. The set of classes consists of 44 standard English phones along with 7 noise sounds (marked with a plus), 4 phones modeling interjections (marked with an ampersand) and silence (SIL).

Already the top level split separates silence, breathing and noise sounds (lower subtree) almost perfectly from phonetic sounds (upper subtree). Furthermore, clusters of acoustically similar phones can be observed in the ACID tree, for instance

- IX,IH,IY,Y
- JH,CH,SH,ZH
- Z,S,F
- ER,AXR,R

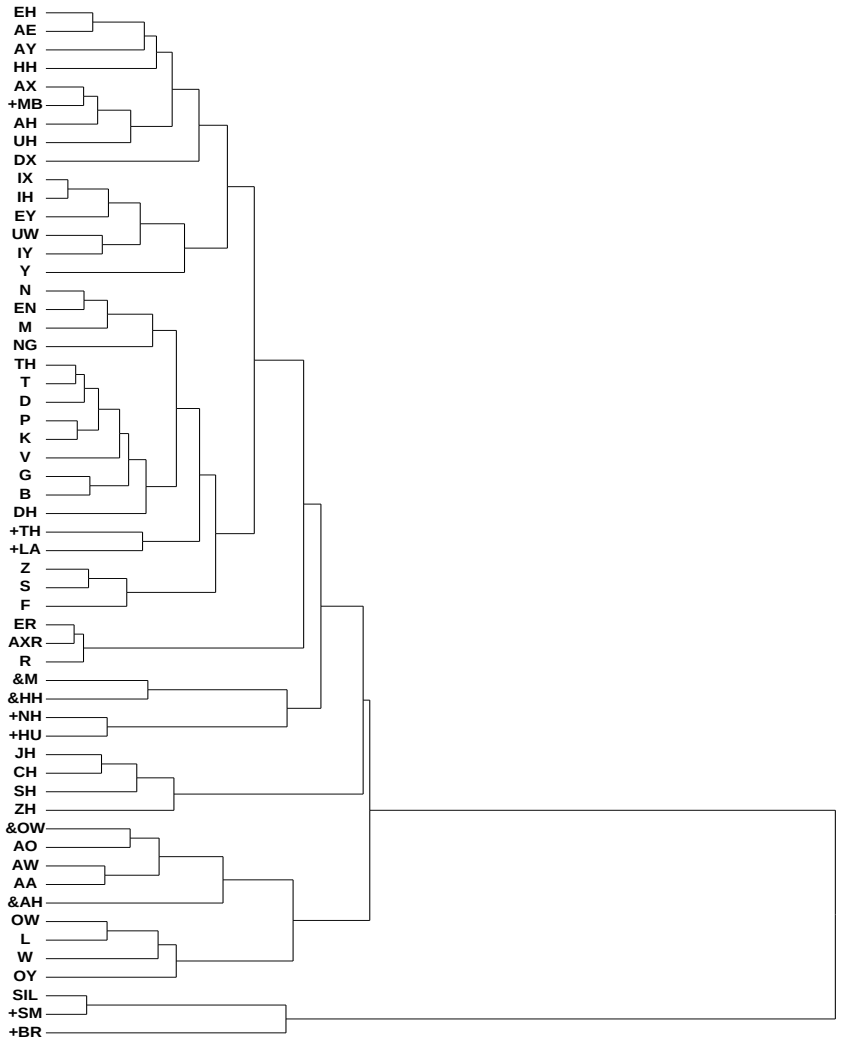

Fig. 5. Typical dendrogram of ACID clustering

ACID clustering was found to be quite effective in generating a hierarchical decomposition of a classification task into subtasks of increasing difficulty (when traversing the tree from root node to leaves). In the case of connectionist acoustic modeling for speech recognition, we observed that nodes in the upper layers of an ACID clustered HNN tree distinguish between broad phonetic classes, whereas nodes further down the tree begin to distinguish the particular phones within a broad phonetic class. Thus, ACID clustering constitutes an effective algorithm for discovering inherent hierarchical structure and exploiting it for

modular classification.

Model Selection The choice of model size and topology becomes very important in the application of hierarchical soft classifiers to tasks such as connectionist speech recognition. While the global tree topology is determined by the outcome of the ACID clustering (or any other tree design procedure), it remains to decide on local (node-internal) classifier topology. The task of a local classifier is to estimate conditional posterior probabilities based on the available training data. Since a particular local estimator is conditioned on all predecessor nodes in the tree, it only receives training data from all the classes (leaves) that can be reached from the respective node. This amounts to a gradually diminishing training set when going from root node to nodes further down the tree. Fig. 6 shows this property of HNNs with a plot of the amount of available training patterns vs. node depth for a binary hierarchy with 6000 leaves. Note the logscale on the ordinate.

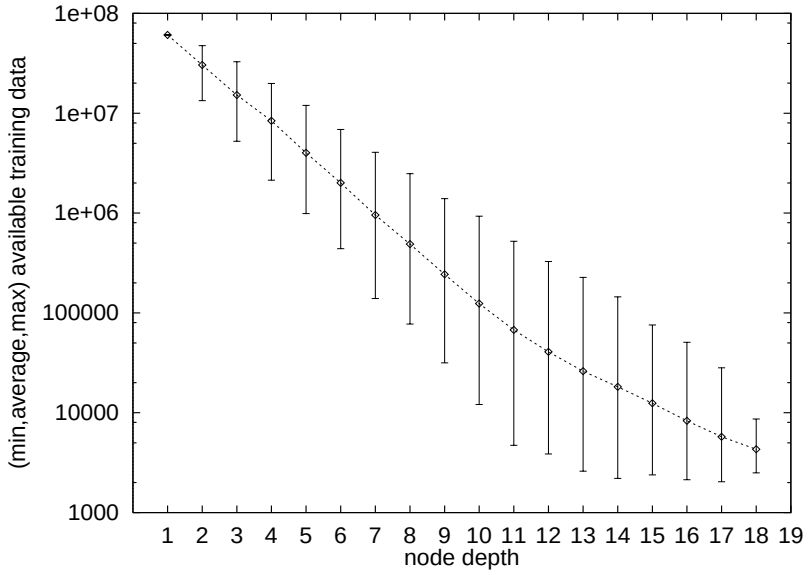

Fig. 6. Available Training Data in Different Depths of HNN Tree

When deciding on the local model complexity, we consider tree nodes as lying in a continuum between the following two extrema:

Top of the Hierarchy
- large amounts of training data available
- allows for large node classifiers
- relatively easy, general classification tasks

Bottom of the Hierarchy

- only small amounts of training data available
- requires relatively small node classifiers
- comparably hard classification tasks
- high degree of specialization

Ideally, the complexity of local node classifiers should be selected so as to maximize generalization ability of the complete hierarchy. Generalization, on the other hand, is influenced by three factors: (1) size and distribution of the training set, (2) model complexity and (3) classification complexity of the specific task at hand. Obviously, we can not alter the latter of these factors. Furthermore, in the context of our architecture, we assume that the size of the training set for each node is fixed by the tree topology, once the hierarchy has been designed. Therefore, we have to choose model complexity based on available training data and difficulty of classification task.

In our experiments in connectionist acoustic modeling, we typically use multi layer perceptron (MLP) nodes with a single hidden layer and control model complexity by varying the number of hidden units. We use standard projective kernels with tanh activations for the hidden units and a task dependent non-linearity for the output units (sigmoid for binary and softmax for multiway classification). The overall number of weights in such a network depends linearily on the number of hidden units. According to [1] and with some approximations, a rule of thumb is to choose the number of hidden units M to satisfy

$$N > \frac{M}{\epsilon}$$

where N is the size of the training set and ϵ is the expected error rate on the test set. In our case, the variation in the number of training patterns in the different nodes dominates the above formula. Therefore, we set the number of hidden units proportional to b^{-n}, where b is the branching factor of the classification tree and n is the node depth. As long as the tree is approximately balanced in terms of the prior distribution of child nodes, this strategy leads to hidden layers with size proportional to the number of available training patterns. A more fundamental treatment of model complexity using multiple training runs and cross validation is desirable. However, in case of large-scale applications such as speech recognition, such a strategy is not realizable because of the high computational cost resulting from very large training databases. Less heuristic approaches to select model complexity still have to be explored.

4.6 Training Hierarchies of Neural Networks on Large Datasets

For the demonstration of various aspects of training large and complex structures such as hierarchies of neural networks on typical datasets, we report on experiments on the Switchboard [19] speech recognition database. Switchboard is a large corpus of conversational American English dialogs, recorded in telephone quality all over the US. It consists of about 170 hours of speech which typically

corresponds to about 60 million training samples. The corpus currently serves as a benchmark for the official evaluation of state-of-the-art speech recognition systems. Switchboard is a comparably hard task, current best systems achieve word error rates in the vicinity of 30-40%. Fig. 7 shows the structure of an HNN based connectionist acoustic model for an HMM based recognizer, in our case the Janus recognition toolkit (JanusRTk) [8].

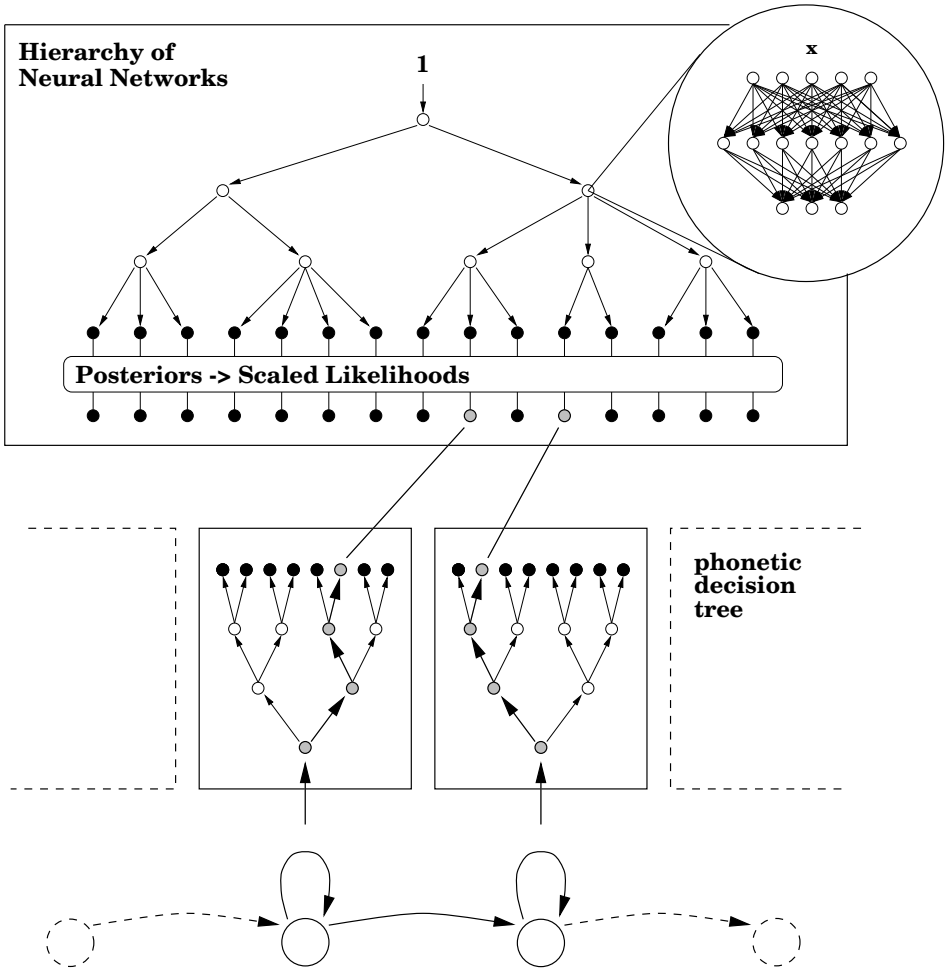

Fig. 7. Hierarchy of Neural Networks for Connectionist Acoustic Modeling: The upper part shows an ACID clustered HNN after node merging. This architecture computes posterior probabilities for a set of generalized polyphones. To allow for integration into the HMM framework, these posteriors are converted to scaled likelihoods. The correspondence to actual HMM states is accomplished by means of phonetic decision trees.

Due to the inherent variability and complexity of the task and the large amount of training data, typical speech recognition systems model several thousand distinct subphonetic units (HMM states) as base classes. This requires to train an estimator for posterior probabilities of thousands of distinct acoustic classes based on millions of training samples, in order to take advantage of the full modeling granularity of the speech recognizer.

In the following, we will discuss several aspects of training a hierarchical soft classifier on large datasets such as Switchboard. Due to the modular structure of the classifier, the size of the model inventory and the training database, the following discussion leads to rather unique problems and solutions. However, it is important to emphasize that these properties stem from the structure of the classifier and the size of the task - not from the specific task of acoustic modeling for speech recognition. Thus, they are transferable to comparably large tasks, e. g. handwriting, speaker or face recognition.

Classifier Tree Topology Depending on the number of classes to be modeled, tree design algorithm, branching factor and size and structure of local node classifiers have to be chosen. For Switchboard, we were experimenting with three systems consisting of 6000, 10000 and 24000 distinct classes, respectively. We used the ACID clustering algorithm to design an initial tree structure from the set of base classes for the 6k and 24k systems. As a second step of the tree design phase, we applied a greedy node merging algorithm on the ACID clustered hierarchy. Node merging decreases the number of internal nodes while increasing the average branching factor (arity) of the tree. Training of such hierarchies is less problematic than training of the original binary tree structure since the difference among nodes (in terms of the number of available training patterns) is somewhat extenuated and the overall number of networks is reduced. However, local classification tasks change from 2-way (binomial) to more complex multi-way (multinomial) problems which might have an impact on the accuracy of estimating conditional posteriors. Therefore, we constrain the node merging algorithm to produce nodes with a maximum branching factor of 8-12. This value was found to improve training speed while not affecting overall classifier accuracy. Considerably larger branching factors are not reasonable in our case as we would gradually loose the advantage of the hierarchical structure by flatening the tree.

For the 10k system, we were using the architecture of Fig. 4 that was designed by prior knowledge, not taking into account any measure of class similarity. This structure exhibits a larger average branching factor and less depth than the ACID clustered trees. Although we could decrease the branching factor at the MONO node by introducing linguistic classes as mentioned earlier, we still have large branching factors at the context nodes which are much harder to resolve with prior knowledge only.

The resulting tree nodes were instantiated with MLPs of varying size of the (single) hidden layer. The local MLPs output layer were parameterized with the softmax non-linearity for two reasons. First, it complies to the property of the

level	# nodes = # networks	# hidden units/network
1	1	256
2	1	256
3	1	256
4	3	192
5	19	128
6	121	64
7	816	32
total	962	

level	# nodes = # networks	# hidden units/network
1	1	128
2	10	128
3	77	64
4	524	32
5	3434	16
total	4046	

Fig. 8. Overview of ACID clustered HNNs for 6k (left) and 24k (right) classes

modeled probability distribution to sum up to one, and second, the softmax function implements the expected value of the multinomial probability density. Fig. 8 gives an overview of the structure of the ACID/HNN systems. Tree compactification reduced the number of internal nodes of the 24k system from 24k to about 4k by increasing the average number of local classes (average branching factor) from 2 to about 8. Especially when dealing with large numbers of classes, we found that moderate tree compactification improved classifier performance. The overall numbers of parameters of the tree classifiers were 2M for the 6k system, 2.4M for the 10k system and 3.1M for the 24k system.

Training Algorithm and Parameters Training a distributed, hierarchically organized collection of neural networks on different amounts of training data is a challenging task. Our training criterion is maximum likelihood, assuming a multinomial probability model (1-out-of-N) over all base classes. A target class label is associated with each training pattern, indicating the correct base class. All networks in nodes along the path from root node to the current target class' leaf receive the current pattern for training. Because of the large amount of training data, we use on-line (stochastic) gradient ascent in log-likelihood with small batches (10-100 patterns) to train the individual networks. More elaborate training algorithms which apply second order methods in optimizing the objective function are too costly in our scenario - a single epoch of training, processing all 60 million patterns in the training database takes 3-5 days on a Sparc Ultra workstation. A practical training algorithm therefore must not take longer than 1-4 epochs to converge. Furthermore, because of the large number of networks that have to be trained, a potential training algorithm can not be allowed to use large amounts of memory - which could be the case with second order methods. Of course, training of individual node classifiers is independent and can therefore easily be parallelized for shared memory multi-processors which alleviates the latter requirement.

Since we are relying on *stochastic* gradient ascent in our training method, we additionally use a simple momentum term to smooth gradients. Also, we use local learning rates for each network that are initialized with a global learning

rate and adapted individually to the specific learning task. The global learning rate is annealed in an exponentially decaying scheme:

$$\eta_G^{n+1} = \eta_G^n * \gamma_G.$$

Typically, we use an initial global learning rate η_G between 0.001 and 0.01, a momentum constant of 0.5 and a global annealing factor γ_G of $0.999 \ldots 0.9999$ applied after each batch update.

In order to accomodate the different learning speeds of the node classifiers due to the different amount of available training data, we control individual learning rates using the following measure of correlation between successive gradient vectors g_{n-1} and g_n:

$$\alpha_n = \arccos\left(\frac{g_n^t g_{n-1}}{||g_n|| ||g_{n-1}||}\right)$$

α_n measures the angle between the gradients g_{n-1} and g_n. Small angles indicate high correlation and therefore steady movement in weight space. Therefore, we increase the learning rate linearily up to the current maximum (as determined by initial learning rate, annealing factor and number of updates performed) whenever $\alpha_n < 90°$ for several batch updates M. Large angles, on the other hand, indicate random jumps in weight space. We therefore decrease the learning rate exponentially whenever $\alpha_n > 90°$ for several batch updates M. In summary, we obtain the following update rule for local learning rate η_i of network i:

$$\eta_i^{n+1} = \min\left\{\eta_G^{n+1}, \left\{\begin{array}{c}\eta_i^n + \delta \\ \eta_i^n * \gamma \\ \eta_i^n\end{array}\right\}\right\} \qquad \text{if} \left\{\begin{array}{c}\frac{1}{M}\left(\sum_{k=0}^{M} \alpha_{n-k}\right) < 90° - \epsilon \\ \frac{1}{M}\left(\sum_{k=0}^{M} \alpha_{n-k}\right) > 90° + \epsilon \\ \text{else}\end{array}\right\}$$

with linear increase $\delta = 0.001 \ldots 0.01$ and exponential annealing factor $\gamma = 0.5 \ldots 0.9$. The number of batch updates M controls smoothing of α whereas ϵ controls the influence of the global learning rate. For $\epsilon \to 90°$, local learning rates are forced to follow the global learning rate, whereas low values of ϵ allow local learning rates to develop individually. Typical values that have been used in our experiments are $M = 10$ and $\epsilon = 20°$.

Adapting individual learning rates to the training speed is a critical issue in hierarchical classifiers. Networks at the top of the tree have to be trained on very large amounts of training data. Therefore, learning rates must be allowed to become relatively small in order to benefit from all the data and not reach the point of saturation too early. On the other hand, networks at the bottom of the tree have to be trained with comparably small amounts of data. In order to train these networks within the same small number of passes through the overall data, we have to apply comparably large learning rates to reach a maximum in local likelihood as fast as possible. However, unconstrained adaptation of learning rates with aggressive optimization of learning speed may result in failure to converge. In our experiments with global initialization of all networks using the

same maximum learning rate, global annealing of the maximum learning rate and local adaptation of individual learning rates that are constrained to never become larger than the global learning rate gives best results.

Generalization/Overfitting Simply speaking, we did not observe any overfitting in our experiments. Taking a look at the training of a large hierarchy in terms of performance on an independent cross-validation set (Fig. 9), we can see that the likelihood on this data levels off, but never starts to decrease again, as is often observed on smaller tasks. In the plots of Fig. 9, the vertical lines indicate multiple epochs (passes) through the training data (consisting of 87000 utterances). Obviously, the large amount of available training data allows for excellent generalization, early stopping was not necessary. This behaviour is surprising at first sight, because we did not use any kind of explicit regularization of the local MLPs. At second sight, however, we can identify several reasons for the good generalization of HNNs on this task:

- Training data can be considered very noisy in our case, since samples come from a large variety of different speakers and recording conditions. Training with noisy data is similar to regularization and therefore improves generalization [2].
- Consider the hierarchy for the 6k classes system (Fig. 8). Some of the 816 networks at the bottom of the tree probably have not seen enough training patterns to generalize well to new data. Although all of these networks together constitute 85% of the total number of networks, they contribute just as one of 7 networks to any particular posterior probability. The networks in the upper part of the hierarchy have the largest influence on the evaluation of posterior probabilities. For those networks, the amount of available training data can be considered so abundant that test set error approaches training set error rate. In other words, optimal generalization can be achieved.

Results We evaluate the proposed hierarchical classifiers as connectionist acoustic models in a speech recognition system. Performance of speech recognizers is usually measured in terms of the word error rate on a reasonably large set of test utterances. In our case, we test the different acoustic classifiers with the Janus [8] recognizer on the first 30 seconds of each speaker in the official 1996 Switchboard evaluation test set, consisting of 366 utterances not present in the training set.

acoustic classifier	# classes	# parameters	word error rate
HNN	10000	2.0 M	37.3 %
ACID/HNN	6000	2.4 M	36.7 %
ACID/HNN	24000	3.1 M	33.3 %

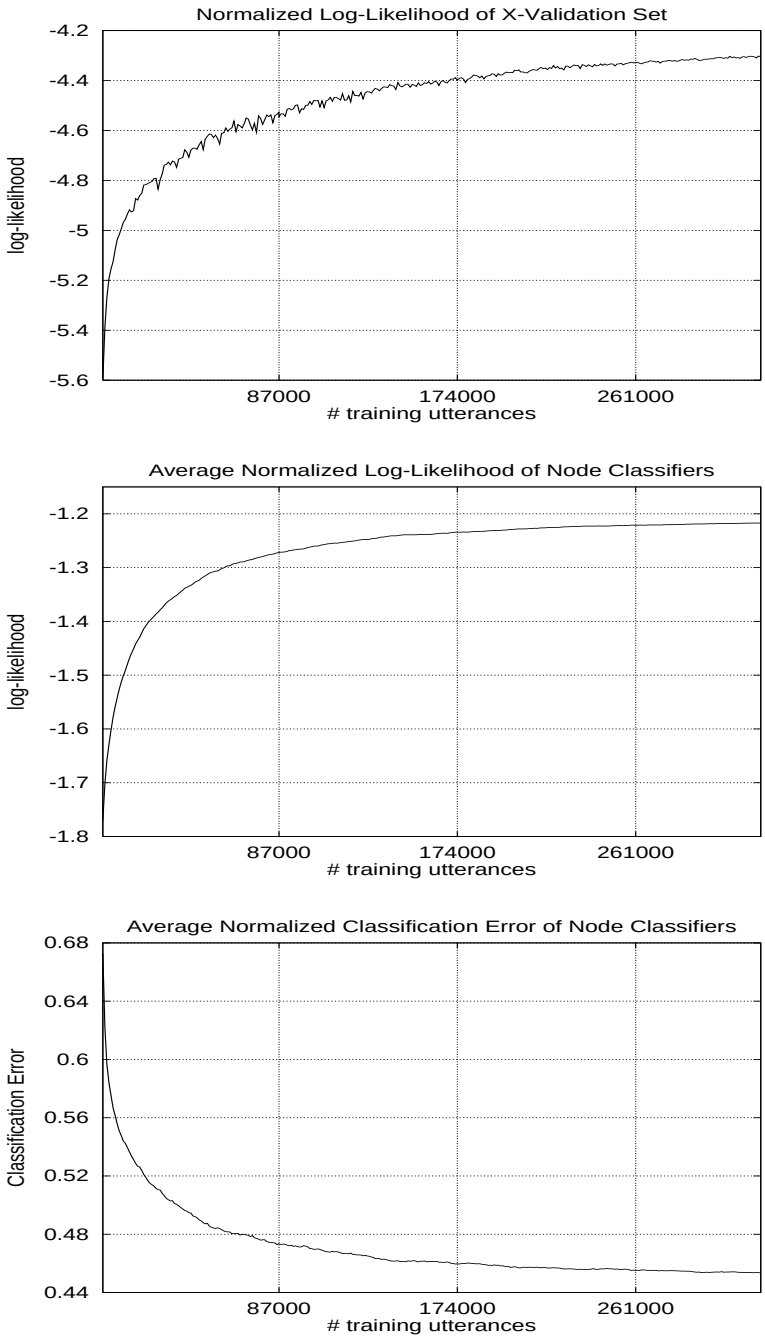

Fig. 9. Cross-Validation during training of 24k ACID/HNN architecture

The above results are competitive with those of state-of-the-art systems and indicate a clear advantage of the ACID clustered over the pre-determined hierarchical classifiers. We suspect, that the reason for the better performance of automatically clustered hierarchies of neural networks is the difference in tree topology. Automatically clustered HNNs such as the presented ACID/HNN trees exhibit small and comparably uniform average branching factors that allow to robustly train estimators of conditional posterior probabilities. In contrast, hand-crafted hierarchies such as the 10k HNN tree contain nodes with rather large branching factors. Fig. 10 shows the branching factors for all the networks in the 10k tree structure.

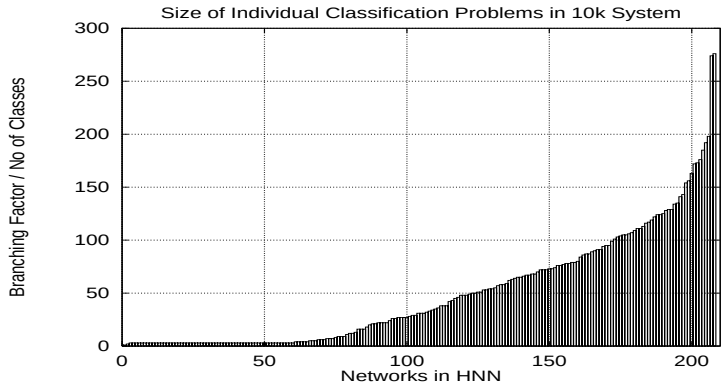

Fig. 10. Branching Factors of Individual Nodes in 10k HNN

The largest observed branching factor in this tree was 276. This requires the joint estimation of conditional posterior probabilities for as many as 276 classes which may result in rather poor approximations to the true posterior probabilities for some of the networks in the tree.

Furthermore, the superior performance of both ACID/HNN classifiers over the hand-crafted 10k tree, demonstrates the full scalability of the hierarchical approach and justifies the increase in the number of parameters. Earlier attempts to train hand-crafted hierarchies for 24k classes failed to provide classifiers that could be used as acoustic models in a speech recognizer. Poor approximations to the real posterior probabilities led to instabilities in decoding when dividing by priors in this case. Apart from that, we do not know of any other non-parametric approach capable of directly and discriminatively estimating posterior probabilities for such a large amount of classes.

5 Conclusions

We have presented and discussed a methodology for the estimation of posterior probabilities for large numbers of classes using a hierarchical connectionist

framework. The aim of the paper is to demonstrate the necessity of hierarchical approaches to modularize classification tasks in large-scale application domains such as speech recognition, where thousands of classes have to be considered and millions of training samples are available. The *divide and conquer* approach proves to be a versatile tool in breaking down the complexity of the original problem into many smaller tasks. Furthermore, agglomerative clustering techniques can be applied to automatically impose a suitable hierarchical structure on a given set of classes, even in the case this set contains tens of thousands of classes. In contrast to the relatively small standard benchmarks for learning machines, aspects such as choice of training method, model selection and generalization ability appear in different light when tackling large-scale probability estimation problems.

References

1. E. B. Baum, D. Haussler (1989) *What Size Net Gives Valid Generalization?*, Neural Computation 1, pp 151-160.
2. C. M. Bishop (1995) *Training with Noise is Equivalent to Tikhonov Regularization*, Neural Computation 7, issue 1, Jan 1995, pp 108-116.
3. H. Bourlard, N. Morgan (1994) *Connectionist Speech Recognition – A Hybrid Approach*, Kluwer Academic Press, 1994.
4. H. Bourlard, N. Morgan (1992) *A Context Dependent Neural Network for Continuous Speech Recognition*, IEEE Proc. Intl. Conf. on Acoustics, Speech and Signal Processing, volume 2, pp 349-352, San Francisco, CA.
5. L. Breiman, J. H. Friedman, R. A. Olshen & C. J. Stone (1984) *Classification and Regression Trees*, Wadsworth International Group, Belmont, CA.
6. J. Bridle (1990) *Probabilistic Interpretation of Feed Forward Classification Network Outputs, with Relationships to Statistical Pattern Recognition*, In Neurocomputing: Algorithms, Architectures, and Applications, F. Fogelman-Soulie and J. Hérault, eds. Springer Verlag, New York.
7. R. Duda, P. Hart (1973) *Pattern Classification and Scene Analysis*, John Wiley & Sons, Inc.
8. M. Finke, J. Fritsch, P. Geutner, K. Ries & T. Zeppenfeld (1997) *The JanusRTk Switchboard/Callhome 1997 Evaluation System*, Proceedings of LVCSR Hub5-e Workshop, May 13-15, Baltimore, Maryland.
9. H. Franco, M. Cohen, N. Morgan, D. Rumelhart & V. Abrash (1994) *Context-Dependent Connectionist Probability Estimation in a Hybrid Hidden Markov Model – Neural Net Speech Recognition System*, Computer Speech and Language, Vol. 8, No 3, pp 211-222, July 1994.
10. J. Fritsch, M. Finke (1997) *ACID/HNN: Clustering Hierarchies of Neural Networks for Context-Dependent Connectionist Acoustic Modeling*, In Proceedings of International Conference on Acoustics, Speech and Signal Processing, May 1998, Seattle, Wa.
11. J. Fritsch (1997) *ACID/HNN: A Framework for Hierarchical Connectionist Acoustic Modeling*, In Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding, December 1997, Santa Barbara, Ca.

12. J. Fritsch, M. Finke & A. Waibel (1997) *Context-Dependent Hybrid HME/HMM Speech Recognition using Polyphone Clustering Decision Trees*, Intl. Conf. on Acoustics, Speech and Signal Processing, volume 3, pp 1759, Munich, Germany.
13. J. Fritsch (1996) *Modular Neural Networks for Speech Recognition*, Tech. Report CMU-CS-96-203, Carnegie Mellon University, Pittsburgh, PA.
14. M. M. Hochberg, G. D. Cook, S. J. Renals, A. J. Robinson, & R. S. Schechtman (1995) *The 1994 ABBOT Hybrid Connectionist-HMM Large-Vocabulary Recognition System*, In Spoken Language Systems Technology Workshop, pp 170-176, ARPA, Jan. 1995.
15. D. J. Kershaw, M. M. Hochberg & A. J. Robinson (1995) *Context-Dependent Classes in a Hybrid Recurrent Network-HMM Speech Recognition System*, Tech. Rep. CUED/F-INFENG/TR217, Cambridge University Engineering Department, Cambridge, England.
16. C. J. Merz, P. M. Murphy (1996) *UCI Repository of Machine Learning Databases*, http://www.ics.uci.edu/ mlearn/MLRepository.html, University of California, Department of Information and Computer Science.
17. N. Morgan, H. Bourlard (1992) *Factoring Networks by a Statistical Method*, Neural Computation 4, No. 6, pp 835-838, 1992.
18. N. Morgan, H. Bourlard (1995) *An Introduction to Hybrid HMM/Connectionist Continuous Speech Recognition*, Signal Processing Magazine, pp 25-42, May 1995.
19. NIST (1997) *Conversational Speech Recognition Workshop, DARPA Hub-5E Evaluation*, May 13-15/1997, Baltimore, Maryland.
20. L. Prechelt (1994) *Proben1 – A Set of Neural Network Benchmark Problems and Benchmarking Rules*, Technical Report 21/94, University of Karlsruhe, Germany.
21. J. R. Quinlan (1986) *Induction of Decision Trees*, Machine Learn. 1, pp 81-106.
22. L. R. Rabiner (1989) *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE 77, pp 257-285.
23. S. R. Safavian, D. Landgrebe (1991) *A Survey of Decision Tree Classifier Methodology*, IEEE Transactions on Systems, Man and Cybernetics, Vol.21, No.3, pp 660-674.
24. J. Schürmann, W. Doster (1984) *A Decision Theoretic Approach to Hierarchical Classifier Design*, Pattern Recognition 17 (3), pp 359-369.
25. J. Schürmann (1996) *Pattern Classification: A Unified View of Statistical and Neural Approaches*, John Wiley & Sons, Inc., New York, 1996.
26. J. T. Tou, R. C. Ganzales (1974) *Pattern Recognition Principles*, Addison Wesley, Reading, Massachusettes.
27. S. Young (1996) *Large Vocabulary Continuous Speech Recognition: a Review*, CUED Technical Report, Cambridge University.
28. S. Lawrence, I. Burns, A. Back, A. C. Tsoi & C. L. Giles (1998) *Neural Network Classification and Prior Class Probabilities*, published in this volume.

This article was processed using the LaTeX macro package with LLNCS style