Using Probabilistically Conditioned Neural Networks to achieve Speaker Adaptation

David Bijl and Frank Fallside
Cambridge University Engineering Department
Cambridge, England.

15 October 1990

Abstract

Speaker Adaptation using Neural Networks is generally difficult because network weights are adjusted in accordance to a whole training set. Introduction of new adaptation data provides a problem, because back-propagation training would converge exactly on that test data, throwing away previously learnt information.

If a neural network is formulated via a probabilistic approach, it is possible to use concepts of maximum likelihood to adapt the parameters of the network so as to accommodate changes without discarding valuable information generalized from initial training.

Here, a probabilistic approach is demonstrated which allows speaker adaptation in automatic speech recognition. The units of speech used are phonic and prosodic.

1 Introduction.

Neural networks have provided performance in speech recognition tasks that compares more than favourably with the traditional statistical technique based on Hidden Markov Models (HMMs) [7].

Given a large database of speech, a network can be trained such that the weights defining the network yield minimum error pattern classification.

Every person has a more or less distinctive voice, which makes possible the identification of a particular person from their voice, albeit prone to error [4]. In speech recognition, it is useful to exploit this property by allowing adaptation of a speech recognizer to each speaker.

In order to achieve an understanding of what network adaptation might achieve, it is useful to relate a traditional minimum risk classifier to a neural network. Consider a set of independent classifiers, whose output $g()$ is some function of an input vector x and a set of learnt parameters p. An input is deemed to belong to a class if the classifier modelling that class outperforms other classifiers.

$$g_i(x,p_i) = max_j\left(g_j(x,p_j)\right) \Rightarrow \text{ x belongs to class i}$$

Consider the output as being connected to a set of internal nodes and inputs. If a node is only connected to the nodes closer to the input than itself (ie to lower nodes), the network is said to be feed forward. Such networks may be trained by the back-propagation algorithm [8]. The input to hidden and output nodes may include both the actual network inputs and lower node values. Most commonly, only two layers are used. The first layer is a set of N_{hd} hidden nodes connected only to the N_{in} inputs. The second layer is a set of output nodes, which connect only to the hidden nodes. Connection is conventionally achieved by taking a linear weighted sum of the inputs.

$$\begin{aligned} f(p_i.x) &= f\left(\sum_j^{N_{hd}} p_{ij}x_j\right) \\ &= \sum_{j=0}^{N_{hd}} p_{ij}^{hd} \sum_{k=0}^{N_{in}} f\left(x_j p_{jk}^{hd}\right) \end{aligned}$$

A network need not be constructed by a linear sum. Indeed more appropriate connection may be considered to be a functional approximator, represented by a polynomial or even transcendental function. A quadratic or higher order polynomial function provides a more general input space descriptor than a linear function, though orders higher than quadratic are not usually practicable because the number of terms required becomes very large.

A quadratic operator is easily described by a matrix notation. Let a_{ij} weight the contribution of the product of the i^{th} and j^{th} node, and define these weights by a real symmetric matrix A^{out}. Similarly, let the i^{th} hidden output be controlled by weights defined by matrix A_i^{hd}.

$$h_i = f\left(x^t A_i^{hd} x\right) \tag{1}$$

$$output_i = f\left(h^t A^{out} h\right) \tag{2}$$

Back propagation is easily adapted for this type of network [7], making training of the network possible.

2 The Aim of Adaptation

Consider the i^{th} output of a multi-layer perceptron. Let it be connected to the outputs of a set of lower nodes h_j, some of which may be input nodes, some of which may be hidden.

$$output_i = f\left(\sum_j w_j.h_j\right)$$

From a probabilistic viewpoint, it is useful to view this as a Bayesian decision approximator. If the network has learned, or generalised a pattern, the probability of a sub-pattern, represented by h_j, affecting the output should be proportional to its probability of occurrence. This will indeed be true if the normal error criterion is minimized with a sufficiently large training set if there is a small overlap between classes.

The sub-patterns, denoted $output_j$, may be input values themselves, or input "constellations" defined by the space partitioning achieved by hidden node weights. The latter case is not possible in two layer linear networks, though it becomes so if super-linear nodes or mode than three layer networks are used.

In a traditional two layer MLP, network outputs are connected to hidden nodes, which themselves are connected to input nodes. The sub-patterns represented by these hidden nodes in the case of speech patterns are shapes representing underlying data cluster concentrations. It is the re-positioning of these clusters that is the principal aim of speaker adaptation.

3 Adaptation using back-propagation

Back-propagation adjusts weights most quickly for output nodes. As one moves away from the output, learning occurs through the back-propagation of errors, which is a second order effect.

This is observed in practice by the fact the network updates are orders of magnitude smaller for hidden weights than for output weights. That it stagnates as one moves from the output is a significant reason for why networks were historically limited to two layers, and why fully inter-connected network have been found to be more prone to local minima problems [8]. If a local minima exists based primarily on a linear mapping of the input to the output then the greater speed of "learning" of the weights connecting the inputs to the outputs will cause this local minima to be a likely stable attractor for the system.

If back-propagation is used for speaker adaption, one is obliged to adapt very slowly. In practice, this tends to be unsatisfactory, because the amount of calibration data then needed corresponds to a very lengthy time.

Modification schemes to a basic network are possible. One scheme, based on a quadratic perceptron node, attempts jointly to minimize the classification error and the description volume of space encapsulated by a node. My experiments have found this to be unsuccessful.

Ultimately one is obliged to realize where back-propagation works most quickly. High-level information, stored in weights nearer the output than the input, is absorbed more quickly than low level information. A trained network if perturbed in the output node weights would re-train quite quickly. A perturbation in the hidden weights, however, is much slower to correct because the back-propagated errors provide second generation knowledge.

It will be seen, however, that weights that define nodes which connect to the inputs must be modified to achieve adaptation.

Consider the language adaptation problem. A new speaker has different sound constellations, because his or her sound for a given vowel or consonant has a different and new quality [6]. For a different speaker, the language spoken is still the same and the statistics of that language, being more a property of the language rather than the speaker, are largely invariant.[1] Given a Bayesian approximator, where output nodes connect and weight input constellations, the output node weights will remain relatively fixed, while the hidden node weights, which describe the sound sub-group statistics, will vary from speaker to speaker.

Back propagation will most quickly adapt the output weights and therefore, in any quick adaptation process will tend to model the a priori statistics of the adaptation phrase more quickly than of the sound sub-group constellation distributions that codify the underlying phone units. This leads to a reduction in performance, which I have observed in previous experiments. An extreme case of this occurs if a unit does not occur in an adaptation set: the unit will, quite sensibly, be turned "off". In this case, one models or adapts the output statistics rather than inputs statistics.

Variations on back-propagation, such as volume minimizing in the nodes, suffer from the this same problem.

4 A Probabilistic Framework for a Network

A alternative approach to back-propagation is to manipulate the hidden units such that their weights are varied to accommodate a new speaker. Such an approach allows adaptation to the input rather than the output.

A solution, as will be shown, is to recognize the duality of a gaussian mixture and a constrained quadratic perceptron [1] and use the statistical insight to initialise and adapt a neural model.

Consider also a gaussian mixture probability density function. A single gaussian N in n dimensions is parametrized by a mean vector μ and a co-variance matrix Σ. Letting t superscript denote the matrix transpose operator and $|\Sigma|$ the determinant of matrix Σ, then the probability of input vector o is

$$N(o|\mu,\Sigma) = \frac{e^{-\frac{1}{2}(o-\mu)^t\Sigma^{-1}(o-\mu)}}{(2\pi)^{n/2}\sqrt{|\Sigma|}} \tag{3}$$

A gaussian mixture $\aleph$ is obtained by adding a weighted set of gaussian together, scaled such that the function integrates to unity. The weights w_m affect the mixture of these gaussian. The functions parameters, comprised of w, μ and σ, are denoted by

[1]Speaker idiosyncrasies will induce slight alteration to the overall personal phoneme statistics of a language, but most current day speech recognition tasks constrain or dictate the grammar removing room from this small possibility of variation.

$$\aleph_j(o) = \sum_m w_{j,m} N(o|\mu_{j,m}, \Sigma_{j,m}) \quad (4)$$

Consider a two-layer quadratic perceptron, as described in equation 1. Equation 4 may be written as a simplification of the network equation. Firstly, the inputs must be augmented by a constant to achieve a threshold and to allow the mean and variance to be condensed into a single matrix. Setting the weights that describe the output, A^{out}, such that they operate in a linear fashion requires that all element are set to zero except those in the last row, and only those connecting to the corresponding mixture are used, as defined by weights $w_{j,m}$. Also, the functional operator $f(x)$ must be set $exp(-x)$.

This simplification causes all mutual information to be lost. In deciding how much evidence exists that input x belongs to output class i, instead of utilising information from all hidden nodes, ie all gaussian estimators, only within class information is used. Furthermore, the output weights are also constrained by their stochastic nature.

One can assign the hidden nodes as specified by a conventional maximum likelihood training scheme to define the nature of the network hidden nodes. New weight values can be derived for the output weights by a one pass process as long as output values are available.

In the case of speech recognition, output values may be assigning the noisy values of 0 or 1, according to whether or not the input is deemed to correspond to the output class. If the hidden node values are augmented by their squared value and a constant, a least mean squared error solution may be computed. It may be shown that this is a least mean square error solution to a quadratic approximation to a Bayesian classifier [5]. Arguably better criterion exist than MSE, but a good solution is computable in one pass of the data. Back-propagation network solutions require hundreds of passes of the training data, making this approach very attractive.

Given a network with hidden nodes that are derived from a within class information training method, one can easily adapt a network for a new speaker by instigating the same process (maximum likelihood) for re-estimation. One is not hampered by a lack of data for certain classes: such classes need not be modified.

When sufficient in-class information exists, the relevant mixture functions may be re-iterated, and the set of of hidden nodes pertaining to that class updated. By using this process during training and during running, adaptation is achieved.

5 Experiments.

Experiments conducted were based on the TIMIT database [3]. The raw speech was preprocessed to produce the following parametrization. FFT spectrum were used for the cepstral generation, because these have been found to be superior to linear predictive cepstrae for the recognition of stop consonants [1]. This is presumably due to the presence of zeros in stop consonant spectra [2].

- 10 cepstral coefficients, based on the FFT spectrum.
- zero crossings
- total energy, up to 8000Hz
- sonorant energy, 60-5000Hz
- low frequency energy, 60-400Hz
- high frequency energy 650-3000Hz
- the ratio of low frequency to high frequency energy.

The labels of the TIMIT database were each used to train a gaussian mixture function by a maximum likelihood method. The mixtures were trained on all speakers of the training set, and then adapted to each speaker where data permitted.

A network classifier was then assigned the task of broad band labelling. More precisely, each TIMIT label was assigned to one of vowel, nasal, stop, fricative or silence. Three hidden nodes were assigned to each TIMIT class, and five outputs were connected to all hidden nodes and the inputs. Inputs are presented, with the potential of context frames, to allow the utilization of spectral change.

Output classification was computed on the basis of the maximum scoring classifier, as previously described.

Two experiments have been conducted, and the last two are being conducted.

No.	Non-Adaptive	Adaptive
1	Male, one dialect	Same speakers, different sentences
2	Optimize the minimum number	
	of adaptation frames.	
3	Male and Female, same dialect.	Same speakers, different sentences
4	Male, one dialect	Male speakers, different dialect

5.1 Results and Discussion

The results some improvement. Prosodic labelling is probably not an ideal task for adaptation, and better results might be expected on a phonetic labelling task. Experiment 1 may be seen as the limiting case of allowing adaptation only if an infinite number of adaptation frame are available. Results for experiments 3 and 4, along further results, will be presented.

Minimum Frames Required.	Recognition Performance
5	64.8
10	65.3
20	65.9
∞	65.1

The results show some improvement, though further experiment are required to find the best possible number of adaptation frames required before a hidden node is re-estimated.