EXPERIMENTS WITH A LEARNING NETWORK FOR A SIMPLE PHONETIC RECOGNITION TASK

S.M. Peeling and J.S. Bridle

Speech Research Unit, RSRE, Mavern, Worcs, UK

INTRODUCTION

Recent developments in adaptive network algorithms [1] offer the possibility of application to various speech pattern processing areas [2]. Most applications so far have been to carefully contrived artificial problems (the only application to a significant pre-existing problem is for letter-to-sound transformation for speech synthesis [3]). We believe that the application of "Parallel Distributed Processing" [4] to speech deserves to be thoroughly explored, but the choice of initial applications needs to be made carefully. In the work reported here we have chosen a speech recognition problem which is attractive as a learning experience because of its scale, and because of previous experience on it using other techniques. The data, the specification, and a program with creditable performance were all available. Quite apart from the speech content, we can regard this experience as an attempt to replicate the function of an existing "expert system". Although the problem may be simple compared with the full phonetic recognition problem, it is quite difficult enough to be getting on with.

THE LUPINS PROBLEM

Peter and Helen Roach of Leeds University put together a broad-class phonetic recognition system (LUPINS) several years ago, as a step towards automatic language classification, funded via JSRU [5].

Using their knowledge and insight as phoneticians, and their experience of the particular task, they chose a few functions of the signal spectrum (acoustic parameters, eg smoothed energy below 500Hz), then found a set of rules for interpreting the pattern of these measurements in terms of 6 or more broad phonetic segment classes (eg vowel, fricative).

The system was developed and tested on a corpus of conversational recordings in 6 different languages, including male and female speakers. The LUPINS process is in .several phases, which were originally separate programs. The acoustic analysis process converts from waveform to raw acoustic parameters (this was originally done in hardware, but is now available in software). The initial parameter processing phase modifies the parameters by "normalising" and "boosting", and computes extra "traces" such as heavily smoothed versions of the originals. The main recognition program (MRP) then decides on a tentative label for each 10ms "frame" (vector) of modified and augmented acoustic parameters, based on data in that frame and 5 adjacent ones. The final phase groups the tentative lables using context and further expertise, to produce the required sequence of phonetic segment labels.

The work reported here is an attempt to copy the function computed by the main recognition program of LUPINS. Since this phase looks only at a small "window" around the current frame, and only labels that central frame, we can think of the function computed by this program as a (non-linear) transformation.

We have not attempted to learn the initial parameter processing, or the final "clean up" phase. MLP should be suitable for the first phase, but for the final phase something more like a dynamic programming algorithm would be more appropriate.

ALTERNATIVE APPROACHES TO LABELLING

We discuss here only the "instantaneous labelling problem": moving from a set of measurements associated with an instant of time to a label appropriate to that time.

The reference LUPINS system applies a series of tests such as comparisons of basic and derived measurements with thresholds and with one another, and logical combinations of such conditions. Each step is easy to understand, but the resulting system can be very difficult to understand and modify.

At the other extreme, consider a classic statistical pattern recognition approach. Each class (corresponding to a label) is considered to have a characteristic distribution in the space of measurements (or of "features" derived from them). An approximation to each class distribution is derived from the "training" data, and the labelling decision at each instant is made on the basis of the likelihood of each distribution giving rise to the current measurements. In this case the principles behind the decision making are very clear, but their validity is open to question, and the kinds of explanations that can be given for particular decisions is rather unsatisfactory. If the distribution for each class is assumed to be Gaussian, with equal variance in all directions for all classes, then the maximum likelihood decision is given by the class with the minimum (Euclidean squared) distance to its class mean [6].

The geometric pattern classification approach, on the other hand, concentrates on the decision making process rather than modelling distributions. Classification boundaries are placed so as to deal with the training data, and it is assumed that the same boundaries will be satisfactory for the unknown data. The special case of a linear (hyperplane) boundary is often chosen: it is easy to implement (with a linear threshold function), there is a simple and effective "learning" algorithm, and the linear case is appropriate for an important sub-set of Gaussian maximum likelihood two class parametric classification (specifically, when the two classes have equal covariance) [6].

The multi-layer logistic soft perceptron (abbreviated MLP) generalises the linear threshold unit. With an MLP having 2 hidden layers we can construct arbitrarily complex decision boundaries [6]. The RHW BP procedure offers a learning method, but it is not guaranteed to find the best set of weights. For more information on the algorithms, see [1,2].

THE DATA

For work reported here we have used a single excerpt from the LUPINS recordings: 50 seconds from the start of the "Jacket" passage (adult male southern British English). This is just an exploratory data set.

The data consists of about 5000 consecutive speech frames representing 7 different classes - Silence, Vowel, Nasal, Low, Fricative, Burst and a "don't know" class, Unclassed. Unfortunately the classes are not evenly distributed throughout the data. In particular,there are only 9 examples of the burst class whilst the silence and vowel classes together account for 86% of the data. This makes it impractical to train on, say, the first 3000 frames since the network could obtain 66% correctness by classing everything as silence. (This did happen in an early run!) Instead, the network is trained on 36 examples of each class, chosen randomly from the first 3000 frames. In the case of the Burst class the data for the first five examples of the class is replicated to make up the 36. The weights are updated after presenting the network with one example of each class.

THE STRUCTURE

All experiments used the same basic structure. There is one layer of "hidden units", and two layers of modifiable weights. All the input units are connected to all the hidden units. We have some experience of different numbers of hidden units. The inputs were either just the original 4 "traces" (short-term power in 4 fequency bands) or these plus the corresponding 4 averages over one second. We used 7 output units: one for each phonetic label (including "Unclassed").

THE LEARNING

The targets were 1 for the output unit corresponding to the correct class (according to the MRP of LUPINS), and 0 for the other six. When reporting performance we take the largest output as the class decision.

While we were still correcting the program and experimenting with learning parameters we made some unsuccessful attempts to use 6 consecutive input frames as the LUPINS main recognition program does. We decided to simplify matters by using only one frame, consisting of the local power outputs of filters, plus an optional set of the 4 corresponding 1 second averages. This 'single frame input' is the condition reported below.

We found it necessary to normalise the input values to lie approximately between 0 and 1, otherwise the initial random weight values had too much effect.

The learning parameters are the Learning Rate (gradient multiplier), and the Momentum Scaling Factor (step accumulator decay factor). A momentum factor of 0.5 was used for all the runs, but we experimented with the 'learning rate' factor.

Each class is represented by 36 training frames. The network is usually presented with 1000-1400 sets of 7 examples during the training phase. After this the weights which have been generated are tested on 385 unseen (consecutive) frames of data containing 112 segments and with the following distribution:

Number	Type
57	Low
168	Vowel
71	Fricative
53	Silence
14	Nasal
20	Unclassed
2	Burst

RESULTS AND INTERPRETATION

Based on a few experimental runs for each of a few conditions we can say that there seems to be a definite advantage in using 8 rather than 4 input values. Performance increased from around 70% correct on test data to around 80% correct. There was no advantage in using 15 rather than 8 hidden units.

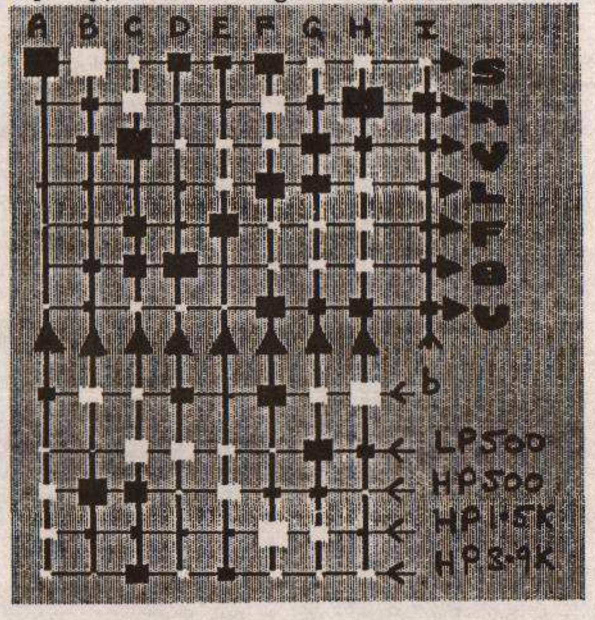
The weights were also tested on the training data and gave 68% - 84% correct classification. Most of the errors lay in the "unclassed" class. The errors were fairly randomly distributed between the classes.

The squared error with respect to the targets (averaged over the 36 training tokens for each class) is the measure of performance which the learning algorithm is designed to optimise. We found that this was a useful indicator of relative performance on different runs, but as expected it did not correlate perfectly with recognition error rate, even on the training data.

AN EXAMPLE LEARNT WEIGHT PATTERN

To illustrate the solutions found by the RHW "learning" rule, the figure shows a set of weights for a small network, with only the 4 traces as inputs, and only 8 hidden units. This set of weights produces 73% correct recognition. Each square represents a weight, joining a horizontal "wire" and a vertical wire. The width of each square is proportional to the magnitude of a weight. Black squares are negative weights, white squares are positive. The four inpur measurements which enter the network along the four horizontal wires at the bottom of the figure are short term power: below 500 Hz, above 500 Hz, above 1.5 KHz and above 3.9 KHz. Biasses are shown as weights from a notional input held permanently at value "1".

An inspection of the figure is rewarded by insights of various depths. The Silence class is picked out by columns (hidden units) A and B. The basic idea is that Silence is indicated by lack of energy (B), and contra-indicated by the presence of energy (A) at any frequency, which is no great surprise!
The Nasal (consonant) class needs mainly low frequencies (C), or rather higher frequencies (F). It is distinguished from Vowel by the former, and from Low by the latter. Much of the logic is negative, for reasons discussed by Rumelhart et. al. [1]. Burst must not have much low frequency energy (D). Fricative must not have a balance well towards low frequency (E). Lows are difficult with these inputs: when we provide local averages as well then a few HUs usually develop a characteristic pattern of weights which compare the instantaneous values with the local averages. Bursts really need more contextual information, but we can see here that middle-to-high frequencies are important.



CURRENT WORK

Currently we are extending the input measurements to include a few frames of context, as used in the LUPINS main program. The current method of treating Unclassed as a class will be changed so that we train on frames actually labelled by LUPINS, and we shall have a reject criterion based on the output values of the MLP. We also have to check the performance of a system with no hidden units, and the use of simple parametric statistical classifiers. When we are satisfied that we are using the right methods we shall train on a more representative data set, and test using new speakers and languages.

We distinguish between two goals: to replicate the performance of an "expert system" for speech recognition, and to do as well as possible for the same problem. Here we have used, as "targets" for the learning, the outputs of the relevant LUPINS program. When we can use the same input data (4 local and 4 average channels, at 6 consecutive frames) then we can test the ability of the MLP to learn the same non-linear transformation as the program implements. However, when we want to tackle the underlying speech problem we shall use as targets the phonetician's best labels, and we shall feel free to use as input measurements anything which might be useful and compatible with our methods.

It is known that for continuous-valued inputs a MLP needs 2 hidden layers (with enough units) to be sure to be able to compute any classification [7]. Although there is reason to expect the current problem to be amenable to a single layer, we should try the more complicated system.

One of our aims is to develop tools to help us understand the weight patterns developed automatically by the learning algorithm, and thus shed light on the potential and limitations. The example weight pattern, in the figure, was produced using a simple system which allows the order of the various types of unit (rows and columns) to be changed interactively, to show orderings and groupings. We are working on automatic methods for arranging the hidden units into "meaningful" one- and two-dimensional patterns, and also modifications of the learning rule which encourage such order to develop.

One very exciting possibility, which is in some ways a converse of better methods for weight interpretation, is a method for non-random assignment of initial weights. We know that the learning algorithm is liable to get stuck in local optima, and it is clear that, particularly for multi-layer systems, a good starting point for the weights could be an enormous help. We are working on a system for "compiling" expertise (specified in terms of the kinds of rules in LUPINS) into a pattern of weights in a MLP network. The BP learning method would then adjust the weights to "tune" performance on training data. The potential for automatic "discovery" of qualitatively different superior solutions would depend on the initial weight magnitudes and the extra degrees of freedom provided.

The method of measuring the performance of a phonetic labelling system is a significant problem in its own right. It has been suggested [8] that a phonetic feature based scoring method be used. A 'componential' representation has advantages for such networks in any case [3,9].

CONCLUSIONS

Although there is still much to do before we have a system worth taking seriously as a broad class phonetic labeller, we are pleased with the experience gained so far. The LUPINS problem and data seem suitable as the basis for further work.

There is the possibility that this work could lead to an effective method for constructing speech sound classifiers, and/or a methodology for applying adaptive network techniques to various "pattern recognition" problems.

Adaptive network methods have the potential to combine some of the strengths of "knowledge based" methods and mathematically-based statistical methods. They can have the richness and complexity of the former and the advantage of the latter of being "tuned" by exposure to large amounts of example data- more than a person can remember the detail of. We need better methods of setting up networks to include, in their topology and initial weights, as much existing speech knowledge as may be useful, we need methods of finding out what is going on as the learning progresses, and we need methods for designing "training courses" to encourage early important and generic distintinctions.

ACKNOWLEDGEMENTS

We are most grateful for the collaboration and help provided by the LUPINS team at Leeds University Department of Linguistics and Phonetics: Peter and Helen Roach, Andrea Dew and Paul Rowlands.

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton and R.J. Williams, 'Learning internal representations by error propagation', *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, D.E. Rumelhart and J.L. McClelland (eds.), MIT Press, (1986).

[2] S.M. Peeling, R.K. Moore and M.J. Tomlinson, 'The multi-layer perceptron as a tool for speech pattern processing', Proc IoA Autumn Conf on Speech and Hearing, (1986).

[3] T.J. Sejnowski and C.R. Rosenburg, 'NETtalk:A parallel network that learns to read aloud', Johns Hopkins Univ Tech Report JHU/EECS-86/01, (1986).

[4] D.E. Rumelhart and J.L. McClelland (eds), 'Parallel Distributed Processing: Explorations in the microstructure of Cognition. Vol 1:Foundations', MIT Press, (1986).

[5] H.N. Roach and P.J. Roach, 'Automatic identification of speech sounds from different languages', Working Papers in Linguistics and Phonetics, Vol 1, pp91-96, Leeds University, (1983).

[6] J.S. Bridle, 'Pattern Recognition techniques for speech recognition', *Spoken Language Generation and Understanding*, C. Simon (Ed.), Reidel, (1980).

[7] I.D. Longstaff and R.J.F. Cross, 'Pattern recognition in the multi-layer perceptron', submitted to Pattern recognition Letters, (1986).

[8] P.J. Roach and H.N. Roach, 'Assessing accuracy in automatic identification of phonetic segments', Proc. IEE Int. Conf Speech Input/Output, (1986).

[9] G.E. Hinton, J.C. McClelland and D.E. Rumelhart, 'Distributed representations', *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations*, D.E. Rumelhart and J.L. McClelland (eds.), MIT Press, (1986).