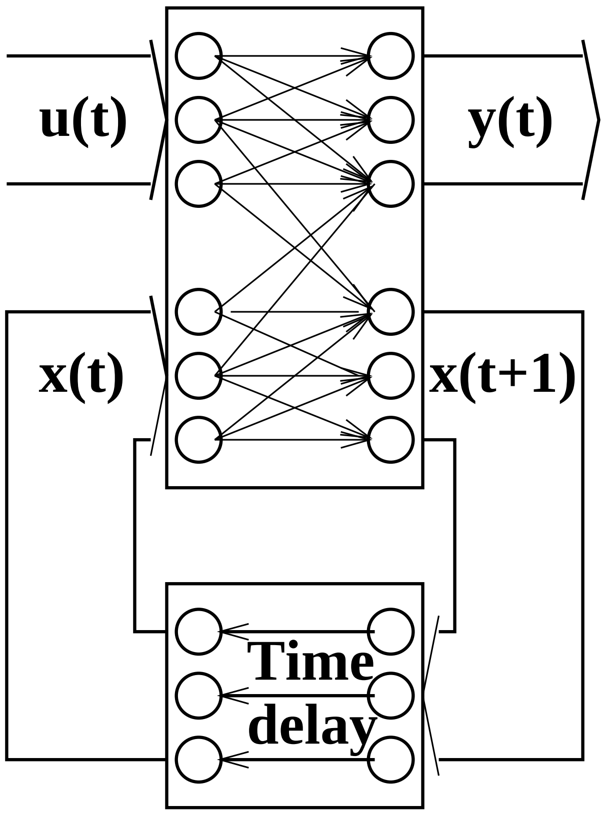
University of Cambridge

MPhil in Computer Speech Text & Internet Technology

Module: Speech Processing II

Lecture 16: Connectionist/Hybrid Systems



Lent 2003

Hybrid Speech Recognition Systems

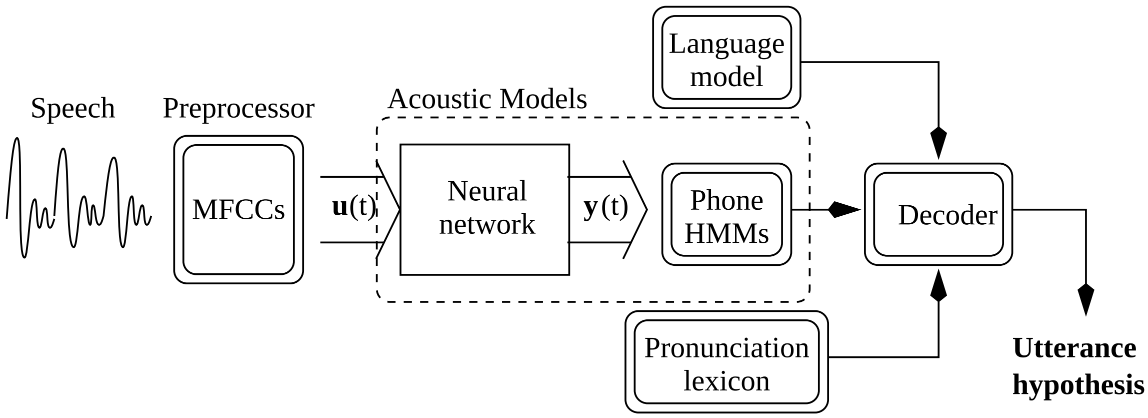
In this lecture we will look at various topics on how neural networks are used in speech recognition systems.

- The Connectionist/HMM Hybrid System
- Multi-layer Perceptrons
- Recurrent Neural Network
- Context Dependent Modelling
- Speaker/Environmental Adaptation

Connectionist/HMM hybrids

In the previous lecture the general attributes of multi-layer perceptrons and how they may be trained were discussed. The topology that has been most successfully used to apply neural networks to speech recognition system is a hybrid scheme. This has two parts.

1. A Markov chain (the states of our standard HMM). This handles the variable length nature of the speech data.
2. One (or possible many) neural network that is used to compute "likelihoods" for each state.



The acoustic model now consists of a neural network and a finite state model.

Neural Nets and Probabilities

Neural Nets may be used to model *conditional probability* or posterior probabilities (e.g. $P(\omega_i|\mathbf{x}) = y_i(\mathbf{x})$) provided

1. The training data set is sufficiently large that it approximates an infinite data set;
2. The size of the network is sufficiently large that a "good" solution exists;
3. The optimisation of the network parameters finds an "appropriate" minimum.

It is not necessary to use neural nets for this task, but they do offer a practical framework for approximating arbitrary non-linear multivariate mappings.

A **cross-entropy** training criterion,

$$E = -\sum_{p=1}^{n}\sum_{i=1}^{K} t_i(\mathbf{x}_p)\log(y_i(\mathbf{x}_p))$$

and a *one-of-K* coding of the target is used, i.e. $t_i(\mathbf{x}_p) = 1$ if the observation at p belongs to class ω_i, is appropriate for the multi class problem.

A **softmax** activation functions should be used for the output layer

$$y_j = \frac{\exp(z_j)}{\Sigma_{i=1}^{K}\exp(z_i)}$$

MLPs for Speech Recognition

The classes become states, one state per class. (Sometimes the state is duplicated for duration modelling).

For standard recognition with an HMM we need $p(\mathbf{u}_t|q_i(t))$ where $q_i(t)$ denotes occupation of state i at time t. Use Bayes

$$p(\mathbf{u}_t|q_i(t)) = \frac{P(q_i(t)|\mathbf{u}_t)p(\mathbf{u}_t)}{P(q_i(t))}$$

$p(\mathbf{u}_t)$ is independent of the state sequence, so ignored. $P(q_i(t))$ is the prior probability of the particular state, class, $q_i(t)$, and may be obtained from the frequency counts in the training data (note independent of time).

$p(\mathbf{u}_t|q_i(t))$ may then be used in a standard HMM decoding procedure.

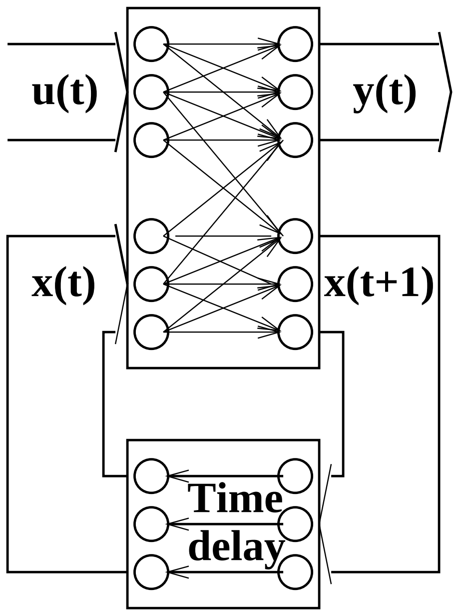
There is still a need to model inter-frame correlation. Common to pad input vector with left and right context (upto +/- 4 frames)

$$\mathbf{u}_t = \begin{bmatrix} \mathbf{o}_{t-w} \\ \vdots \\ \mathbf{o}_t \\ \vdots \\ \mathbf{o}_{t+w} \end{bmatrix}$$

Delta parameters have also been appended to the input feature vector.

Recurrent Neural Network

Recurrent Neural Networks (RNNs) have also been used for the conditional probability modelling.



$\mathbf{u}_t$ is the **input** vector; $\mathbf{x}_t$ is the **state** vector and $\mathbf{y}_t$ is the **output**, "probability", vector.

$$\begin{aligned} \mathbf{y}_t &= \mathcal{F}(\mathbf{u}_t, \mathbf{x}_t) \\ \mathbf{x}_{t+1} &= \mathcal{G}(\mathbf{u}_t, \mathbf{x}_t) \end{aligned}$$

The state vector thus holds a "representation" of the previously seen observations.

RNNs are may be trained using **Error Back Propagation Through Time**.

RNNs for Speech Recognition

The softmax activation function is used for the output layer and the training criterion is cross-entropy. The form of these expressions used are

$$y_{ti} = \frac{\exp(\mathbf{w}_i'\mathbf{z}_t)}{\sum_{j=1}^{K}\exp(\mathbf{w}_j'\mathbf{z}_t)}$$
$$x_{(t+1)i} = \frac{1}{1+\exp(-\mathbf{v}_i'\mathbf{z}_t)}$$

where $\mathbf{v}_i$ is the weight vector for the i^{th} node of the state vector, $\mathbf{w}_i$ is the weight vector for the i^{th} node of the output layer,

$$\mathbf{z}_t = \begin{bmatrix} 1 \\ \mathbf{u}_t \\ \mathbf{x}_t \end{bmatrix}$$

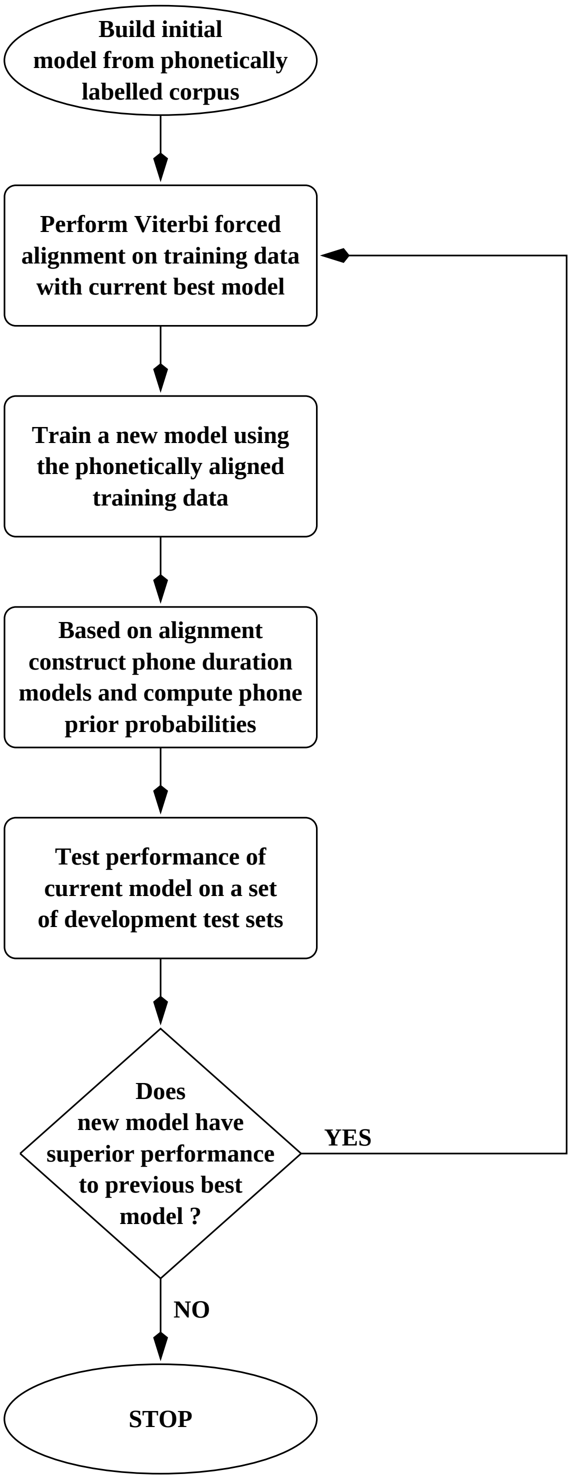
the input vector now typically only consists of the current observation, i.e. $\mathbf{u}_t = \mathbf{o}_t$. The inter-frame correlation is modelled by the state vector. If implemented directly this would allow no future context to be modelled.

In order to build a representation of future acoustic context within the state vector the estimate of the posterior probability is delayed by typically 4 frames

$$y_{ti} = P(q_i(t)|\mathbf{x}_{t+4}, \mathbf{u}_{t+4}) \approx P(q_i(t)|\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{t+4})$$

For non real-time situation models are trained both **forward** and **backward** in time and then merged.

Training Hybrid Systems



Training Hybrid Systems (cont)

In many ways the training of hybrid systems is similar to standard HMM training. Unfortunately the estimation of the model parameters is not as straight forward as HMM training. In standard HMM training the model estimation is **guaranteed** to find the optimal solution given the current frame/state (component) alignment. This is not true for NNs. Here the weights ($\mathbf{W}$) are optimised using iterative techniques of the following form (at iteration $\tau + 1$)

$$\mathbf{W}^{(\tau+1)} = \mathbf{W}^{(\tau)} + \Delta\mathbf{W}^{(\tau)}$$

A consequence of this is that the parameters are not guaranteed to be optimal for the given alignment and many iterations of the optimisation may be required. There is also the standard *stability* problem. For very large networks with large amounts of data it may be hard to train the network and may take a long time (this is particularly true of the RNN).

To "help" the training procedure it is common to also normalise the input vector such that

1. It has zero mean
2. Unit standard deviation

This means that the dynamic range of all the parameters are approximately the same and speeds up the training time.

Training on Large Databases

As the amount of training data increases, so the number of hidden units tends to be increased. Very rapidly the training time becomes very large. Schemes for handling theses large databases are

1. **Special Purpose Hardware**: Due to the simple structure of NNs special purpose hardware can be constructed that significantly increases the network size that can be trained at a particular point in time
2. **Speaker Clustering**: The training speakers are clustered into N groups, in the same fashion as standard speaker clustering. N speaker-dependent recognisers are trained.
3. **"Modified" Boosting**. Instead of using expert knowledge, the current recogniser may be used to split the data. The procedure is
   (a) Train a network on a randomly selected subset of the training data.
   (b) Recognition the training data using current network (either on a frame or word level).
   (c) Select those sentences with the highest error rate to train a second network.

Context Modelling

As connectionist hybrid systems tend to be built on **phones** as the base unit there is the same problem as standard HMMs. That is the acoustic realisation of the current phone is highly context dependent.

It is not feasible to simply increase the number of output nodes to cover all possible phones in all possible contexts (consider the number free parameters required).

One possible factorisation for context dependent modelling is

$$P(q_i(t), c_j^{(l)}, c_k^{(r)} | \mathbf{U}_1^{t+4}) \;=\; P(q_i(t)|\mathbf{U}_1^{t+4}) P(c_j^{(l)}, c_k^{(r)} | q_i(t), \mathbf{U}_1^{t+4})$$

where $\mathbf{U}_1^{t+4} = \mathbf{u}_1, \ldots, \mathbf{u}_{t+4}$.

Again this must be further modified to allow it to be used in a hybrid system

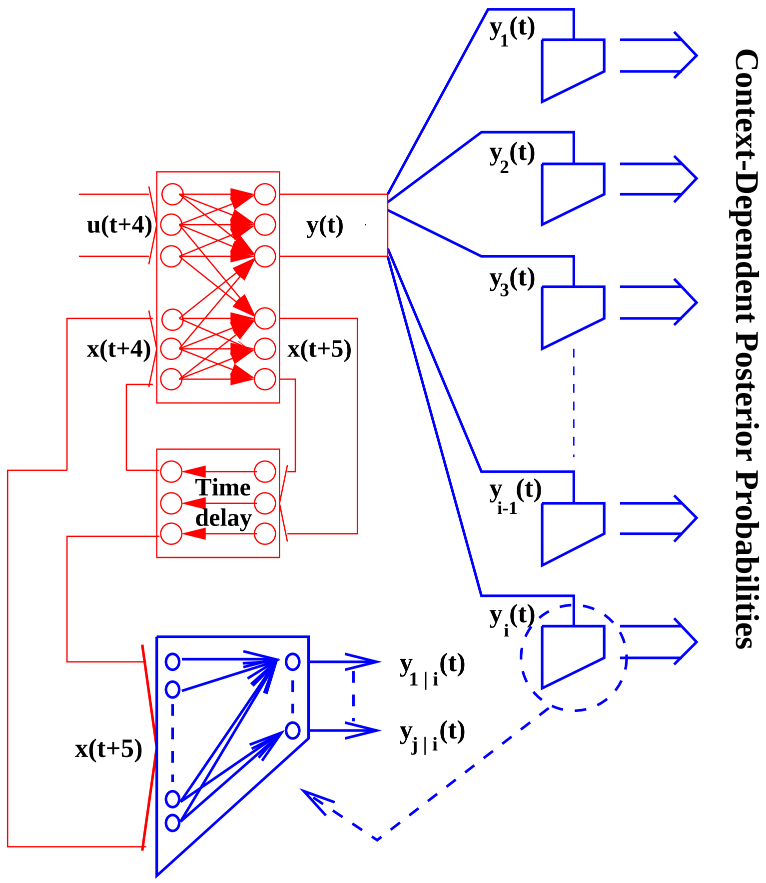
$$p(\mathbf{u}_{t+4}|q_i(t), c_j^{(l)}, c_k^{(r)}, \mathbf{U}_1^{t+3}) = \frac{P(q_i(t), c_j^{(l)}, c_k^{(r)} | \mathbf{U}_1^{t+4}) p(\mathbf{u}_{t+4}|\mathbf{U}_1^{t+3})}{P(q_i(t), c_j^{(l)}, c_k^{(r)})}$$

The denominator is again found from the counts in the training data.

The choice of context classes may be made in exactly the same way as for standard HMMs, e.g. using decision trees.

Context Modelling (cont)

One implementation shown here for a RNN achitecture (applicable to MLPs as well) is to use the following architecture.



For each of the context-independent classes a separate network is built. This is used to discriminate between the various contexts for that class.

Training the CD "Experts"

There are again some choice about the form of the expert training. The training of the one-layer MLP is simple, but the input may be either

1. **Multi-frame Input** In the same way as the MLP training the input vector consists of stacked vectors. Thus

$$P(c_j^{(l)}, c_k^{(r)}|q_i(t), \mathbf{U}_1^{t+4}) \approx P(c_j^{(l)}, c_k^{(r)}|q_i(t), \mathbf{U}_{t-4}^{t+4})$$

2. **State-Vector training**. For the RNN the expert may be trained on the state vector, thus assuming that $\mathbf{x}_{t+5}$ is a good representation of $\mathbf{U}_1^{t+4}$,

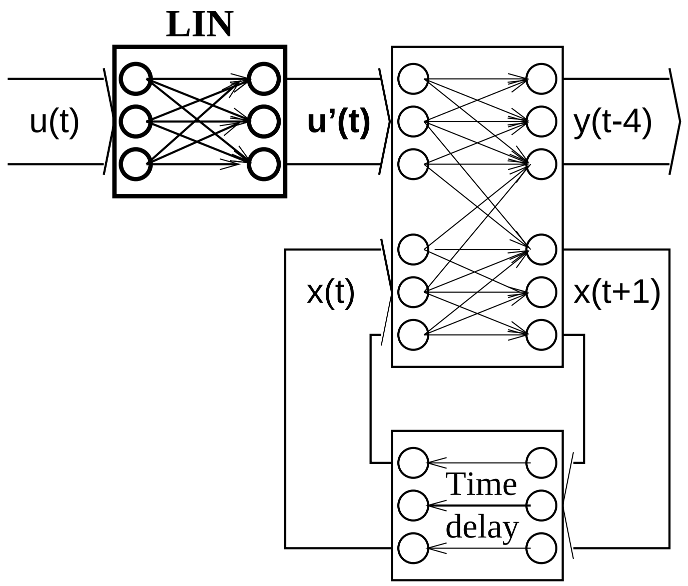
$$P(c_j^{(l)}, c_k^{(r)}|q_i(t), \mathbf{U}_1^{t+4}) \approx P(c_j^{(l)}, c_k^{(r)}|q_i(t), \mathbf{x}_{t+5})$$

So far the "targets" have not been defined. It is not possible to have a separate target for every possible context (standard problem). The context-clustering used is similar to the decision tree clustering previously described.

Adaptation with Neural Nets

Unlike HMMs, it is not possible to associate a "meaning" with each of the parameters of a Neural Network.

Standard enhancement schemes, or speaker normalisation schemes, may be used. Alternatively a matrix (linear input network or LIN) may be used for the adaptation/enhancement.



The LIN training procedure starts by initialisation with an identity matrix (typically only 14×13 parameters including the bias). Then for each iteration:

1. Propagate the input to the output layer of the RNN.
2. Back-propagate the error through the RNN. Note the weights of the RNN are kept constant.
3. Update the LIN's (linear activation function) weights.

The ABBOT System

1. **Training Data**: Wall Street Journal training data.
   (a) **System-1**: Speaker clustering - SI-284 split into 5 speaker groups.
   (b) **Systems-2 and 3**: SI-84 subset - only 84 of the speakers used.
2. **Parameterisation**: Two basic parameterisations:
   (a) **PLP** Perceptual Linear Prediction - 13 features.
   (b) **MEL+** (inc. pitch, voicing and energy) - 23 features.

   System-1 also used delta parameters.
3. **Acoustic Models**: Two models trained for each condition - one *forward* in time, one *backward*.
   (a) **System-1 and 2**: 54 target phones.
   (b) **Systems-3**: 54 target phones + 527 CD targets.
4. **Vocabulary**: 65,000 word vocab, multiple prons
5. **Language Model**: Trigram language model.

Performance:

System	WER (%)	
	Dev. Data	Eval Data
System-1 (Clustered SI-284)	11.2	12.4
System-2 (CI SI-84)	11.5	13.3
System-3 (CD SI-84)	10.0	11.2

Comparing NNs and HMMs

1. **Speed**: NNs are typically faster (all posteriors calculated simultaneously), BUT, in LVCSR a significant amount of time is spent in *search*, HMMs can use fast Gaussian computation.

2. **Simple Training**: HMMs have simple training schemes, NN training is not guaranteed to reach the local maximum.

3. **No. Parameters**: NNs can have an order of magnitude fewer parameters than HMMs, BUT, in LVCSR the language model typically dominates the number of parameters and MMI training of HMMs may be used (& tying schemes can also similarly reduce HMM parameters).

4. **Parameter Meaning**: HMM parameters have a "meaning" so model-based adaptation, either environment or speaker, may be performed, BUT, NNs can still use enhancement/front-end schemes for robustness.

5. **Performance**: The best HMM systems give better performance on large training data sets e.g. 1997 DARPA broadcast news evaluation best HMM system (HTK) 16.2% word error, only NN system 27%.