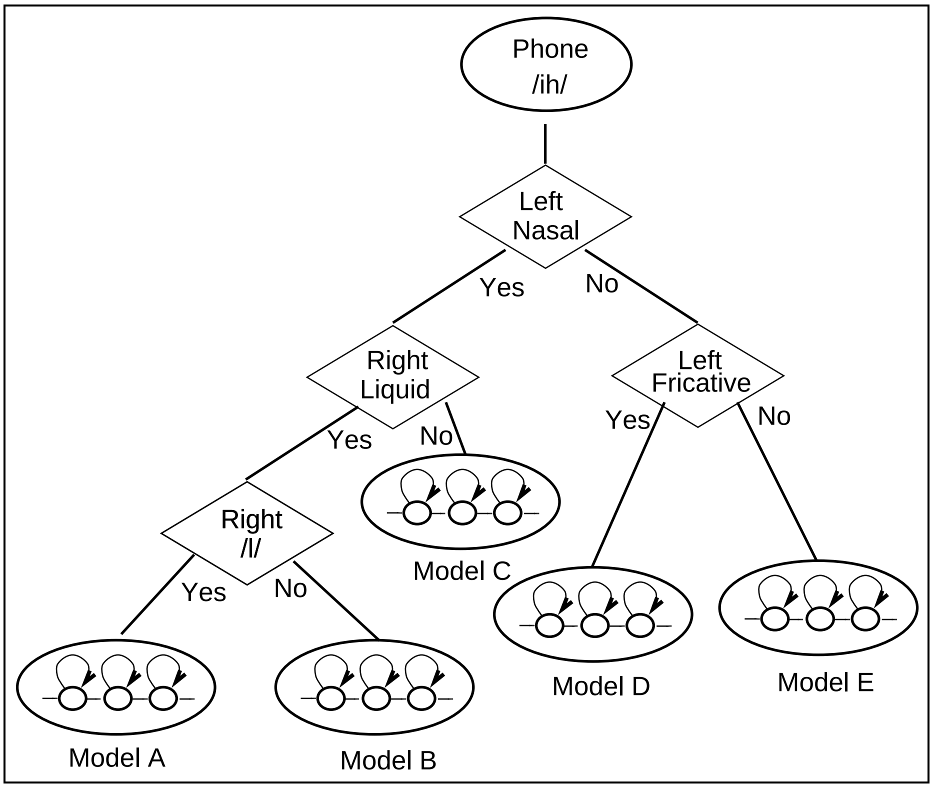
University of Cambridge

MPhil in Computer Speech Text & Internet Technology

Module: Speech Processing II

Lecture 4: Issues in HMM Training



Lent 2003

Issues in HMM Training

Discriminative Training

System Trainability

- maximum a-posteriori training
- parameter tying

HMM Topology

Number of State Components

Building an HMM System

Limitations of MLE

So far only Maximum Likelihood (ML) training has been examined. It maximises

$$\mathcal{R}_{ML} = \sum_{r=1}^{R} \log p_\theta(\boldsymbol{O}^r|\mathcal{M})$$

where $p_\theta(\boldsymbol{O}^r|\mathcal{M})$ is the likelihood of generating the training data $\boldsymbol{O}^r$ with the model $\mathcal{M}$ having the parameters θ.

Thus ML training maximises the likelihood of the data for a particular class. It does not take into account the data from the other classes, ie it does not guarantee to increase the discrimination between classes (words).

ML training may be shown to be a *good* estimation scheme (a consistent minimum-variance estimator), if:

1. The observations are from the assumed family of distributions.
2. The true language model is used in recognition.
3. Training sample set is very large.
4. As the model parameters move towards the *true* model parameters the performance of the decoder improves.

In general none of these are true. It is therefore worthwhile examining other criteria.

Discriminative Training

The best choice would be to minimise the word error rate of the system.

$$\mathcal{R}_{DISC}(\theta) = 1 - \frac{1}{R}\sum_{w=1}^{W} 1_{w_\theta = w}$$

where $1_{w_\theta=w}$ returns 1 if the recogniser says the word is w. Unfortunately this is highly complex, as the function $1_{w_\theta=w_r}$ is not differentiable! Instead **Discriminative** training criteria may be used.

Recently there has been interest in a variety of training criteria:

1. Minimum Empirical Error (MEE);
2. Maximum Mutual Information (MMI);
3. Conditional Maximum Likelihood (CML);
4. Minimum Discrimination Information (MDI);
5. Minimum Phone Error (MPE);
6. Corrective Training.

In all cases the same model structure and units are used. However the different criteria will result in different parameter values.

MMI Training

One approximation is Maximum Mutual Information training. MMI training is defined as

$$\hat{\theta} = \arg\max_{\theta}(\mathcal{R}_{MMI}(\omega;\boldsymbol{O}))$$

where $\mathcal{R}_{MMI}(\omega;\boldsymbol{O})$ is the **mutual information** between the training data and the model set. The language model is normally optimised independently of the acoustic model data, so what is optimised is

$$\mathcal{R}_{MMI}(\omega;\boldsymbol{O}) = \sum_{r=1}^{R}\left\{\log p_{\theta}(\boldsymbol{O}^r|\mathcal{M}) - \log p_{\theta}(\boldsymbol{O}^r)\right\}$$

or $p(\mathcal{M}^w|\boldsymbol{O})$ is optimised, which is identical to CML training. The second term may be written as

$$p_{\theta}(\boldsymbol{O}^r) = \sum_{w=1}^{W} p_{\theta}(\boldsymbol{O}^r|\mathcal{M}^w)p_{\theta}(\mathcal{M}^w)$$

The first term of $\mathcal{R}_{MMI}(\boldsymbol{\omega};\boldsymbol{O})$ is identical to the standard ML training term. The difference lies in the second term, which may be interpreted as trying to minimise the likelihood of the wrong model sequences.

Implementation Issues

There are implementation problems. The main problem is that in contrast to ML training, there is no closed-form solution to the problem of maximising the objective function given the complete dataset. It is therefore either optimised using standard optimisation techniques, e.g gradient descent or using a set of re-estimation formulae have been proposed that, slowly converge but don't guarantee an increase in objective function at each step. There is still the problem of computing $\log p_\theta(\boldsymbol{O}^r)$ which should include all possible word sequences. One possibility that scales to large datasets is to use word lattices to compactly encode the most probable set of word sequences.

Closely related to MMI training is frame discrimination (FD) training in which the term $\log p_\theta(\boldsymbol{O}^r)$ is computed using all the Gaussians in the HMM system. With care this can be computed rapidly and can be as effective as lattice-based MMI but more computationally efficient.

System Trainability

In the first lecture of this module various problems associated with using context-dependent phone models were mentioned, *robust parameter estimation*, *unseen contexts* and *flexibility in number of model*. In this lecture, solutions to these problems will be described and related to estimating the parameters of a CDHMM systems. The main techniques are:

1. **Backing-Off**. If there is insufficient training data to train, for example a particular triphone, *back-off* to the appropriate biphone. If insufficient data for the biphone, back-off to the monophone.
2. **Parameter Smoothing**. Average the parameters from specific, poorly trained models, with more general, well trained, models. For example smoothing context dependent models with context independent models.
3. **Maximum A-Posteriori Training**. Prior information on parameter values is used in the training of the models.
4. **Parameter Tying**. Use the same parameter values in different distributions, models etc and pool the training data.

MAP Estimation

So far in the estimation process no prior information about the form of the probability distributions has been used. If prior knowledge is used, this leads to **Maximum A-Posteriori** estimation. This may be viewed as a smoothing process. For example context-independent models may be used as priors for context-dependent models. For the CDHMM case, the means, variances and mixture weights may all have priors on them.

The general MAP estimation for Gaussian mixture CDHMMs is complex, but for a single dimensional Gaussian the MAP estimate for the mean is

$$\mu_{MAP} = \frac{\sigma^2 \mu_p + \sigma_p^2 \Sigma_{t=1}^{T} L_j(t) o_t}{\sigma^2 + \sigma_p^2 \Sigma_{t=1}^{T} L_j(t)}$$

where σ^2 is the assumed known variance, μ_p and σ_p^2 are the prior mean and prior variance of the mean. This is often written as

$$\mu_{MAP} = \frac{\tau \mu_p + \Sigma_{t=1}^{T} L_j(t) o_t}{\tau + \Sigma_{t=1}^{T} L_j(t)}$$

Values of τ in the range 2-20 have been used. The smaller the value of τ the greater the influence of the observed data.

The MAP estimate is thus a weighted average of the prior mean and the mean of the training data. It is interesting to see what happens at the limits.

$$\begin{array}{ll} \Sigma_{t=1}^{T} L_j(t) = 0 & \mu_{MAP} = \mu_p \\ \Sigma_{t=1}^{T} L_j(t) \rightarrow \infty & \mu_{MAP} = \frac{\sum_{t=1}^{T} L_j(t) o_t}{\sum_{t=1}^{T} L_j(t)} \end{array}$$

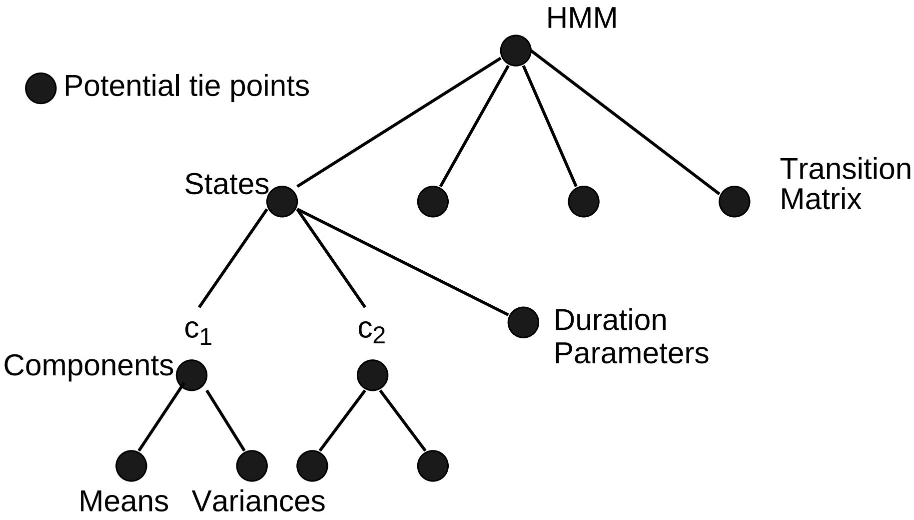
Thus with enough data MAP training becomes ML training.

Obtaining good estimates for the priors is a complex task. The simplest approach is to use the CI model parameters as priors and empirically set τ to an appropriate value.

Note for those of a mathematical bias: MAP estimates are usually based on the use of *conjugate priors*. A conjugate prior is defined such that the posterior distribution of the parameters will belong to the same family of distributions as the prior distribution for any sample size and any observation. For the case of the Gaussian mean the conjugate prior is simply a Gaussian distribution.

Parameter Tying

As previously mentioned parameter tying may be performed at a variety of levels. For example in HTK:



1. **Generalised Triphones**. Different triphone contexts share the same model.
2. **State-Clustered Triphones**. States of different triphones share the same distributions.
3. **Tied Mixture** (or SCHMMs). All output probability distributions (Gaussians) are shared across all HMMs.
4. **Grand Variance**. The same variance matrix is shared over all Gaussians.

Generalised tying is implemented by simply pooling the data from all the data examples for any particular tied parameter.

Bottom-Up Parameter Tying

Assume that all contexts are distinct but to ensure reliable estimates smoothing or sharing is required. The basic procedure is:

1. Models are built for all observed contexts.
2. Merge "models" that are acoustically similar.
3. If sufficient data available stop.

Generalised Triphones
The model comparisons and merging may be done at the model level to form *Generalised Triphones*.

State-Clustered Triphones
Comparison and clustering may be performed on the state level to form *State-Clustered Triphones*. Allows left state to be clustered independently of the right state.

Limitations
Unreliable for contexts that occur rarely in training data. Unable (without using back-off) to cluster contexts not seen in training data.

Distance Measures between Models

Bottom up clustering requires a distance measure between two models (or states). A general likelihood based distance measure is

$$d(\mathcal{M}_1, \mathcal{M}_2) = \frac{1}{T^1}\Big(\log(p(\mathbf{O}^1|\mathcal{M}_1)) - \log(p(\mathbf{O}^1|\mathcal{M}_2))\Big)$$

where $\mathbf{O}^1$ is a set of data generated by "model" $\mathcal{M}_1$ and T^1 is the number of samples in that set. For simplified case of distance between two single Gaussian component states this simplifies to

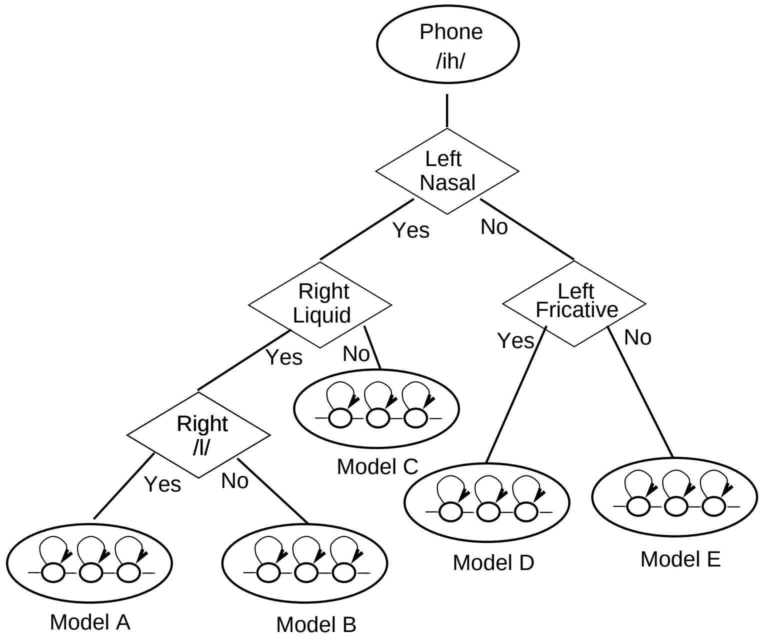
$$\begin{aligned} d(\mathcal{M}_1, \mathcal{M}_2) &= \int \log\left(\frac{p(\mathbf{o}|\mathcal{M}_1)}{p(\mathbf{o}|\mathcal{M}_2)}\right) p(\mathbf{o}|\mathcal{M}_1) d\boldsymbol{o} \\ &= \frac{1}{2}\Big(\mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1 - \boldsymbol{I}) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)'\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ &\quad + \log\left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|}\right)\Big) \end{aligned}$$

(the *trace* is defined as $\mathrm{tr}(\boldsymbol{A}) = \sum_i a_{ii}$) This is the divergence or Kullback-Leibler number. Symmetric forms are also used where

$$d'(\mathcal{M}_1, \mathcal{M}_2) = \frac{d(\mathcal{M}_1, \mathcal{M}_2) + d(\mathcal{M}_2, \mathcal{M}_1)}{2}$$

An alternative scheme is to minimise the loss in likelihood that results from merging the models (see top-down parameter tying).

Top-Down Parameter Tying



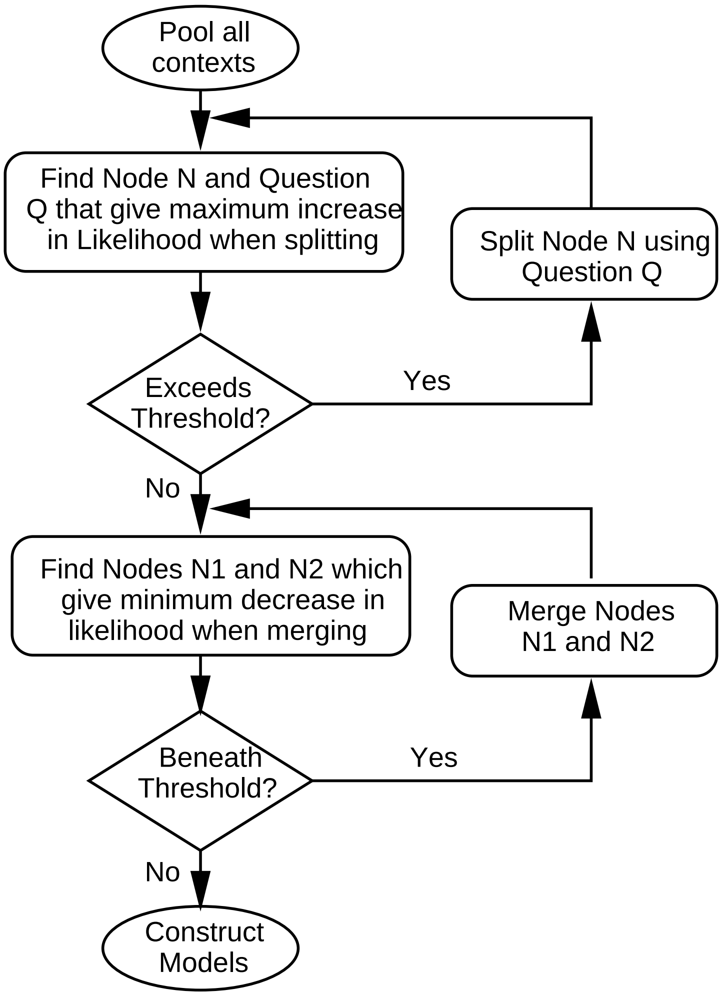
Simple Binary decision tree, at each node yes/no decision.
Advantages

1. No need to back-off, handles unseen contexts elegantly.
2. Allows *expert* knowledge to be incorporated
3. Allows any degree of context dependency to be simply incorporated.

Disadvantages

1. Only locally optimal splits are selected.
2. Not all question combinations normally asked

Constructing Decision Trees



To make decision tree generation computationally efficient

1. The frame/state alignment is not altered (hence the contribution of the transition probabilities may be ignored).
2. Single component Gaussians are accurate enough to decide on contexts.

Change in Likelihood

The change in the data likelihood of either splitting or merging contexts must be calculated to select the best question to ask at each stage of tree construction.

1. **Split** If we hypothesise a split at a particular node, p, into a set of descendents, $1 \ldots D$, the change in likelihood is

$$\begin{aligned}\Delta \log\left(p(\mathbf{O}|C')\right) = & -\sum_{d=1}^{D} \frac{1}{2} \log(|\mathbf{\Sigma}_d|) \sum_{t=1}^{T} L_d(t) \\ & + \frac{1}{2} \log(|\mathbf{\Sigma}_p|) \sum_{t=1}^{T} L_p(t)\end{aligned}$$

   C' is the new parameter tying.

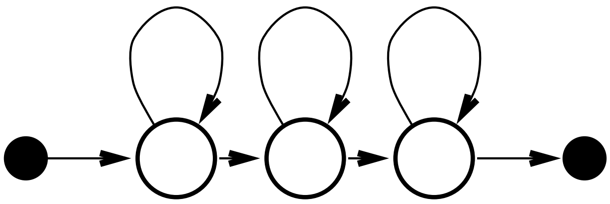
2. **Merge** If we hypothesise a merge of descendents $1 \ldots D$ into parent node, p, the change in overall data likelihood is

$$\begin{aligned}\Delta \log\left(p(\mathbf{O}|C')\right) = & \sum_{d=1}^{D} \frac{1}{2} \log(|\mathbf{\Sigma}_d|) \sum_{t=1}^{T} L_d(t) \\ & - \frac{1}{2} \log(|\mathbf{\Sigma}_p|) \sum_{t=1}^{T} L_p(t)\end{aligned}$$

Note that if the mean vector and covariance matrix is stored for each state along with an occupation count the required covariance for any combination of states (i.e. any point in the tree) can be very simply computed.

HMM Topology

One of the first design decisions to be made is the HMM topology. Often the same model structure is used for each HMM, e.g.



Standard Phone Model

This is almost certainly sub-optimal. Some labs/systems use a similar left-to-right topology but with the number of states proportional to the average phone duration. (Note this increases the number of parameters in the system.)

Good techniques for determining the appropriate HMM topology is an area of active research.

Number of Gaussian Components

A variety of schemes have been used for determining the number of components for each state.

- **Equal numbers** : In the same fashion as fixing the state topology the number of components may be set to be the same.
- **Dependent on state occupancy** : Make the number of components proportional (upto a maximum number of components) to the number of observations in the state.
- **Bayesian information criterion** : Use BIC to determine the number. Thus increase components until
$$BIC(\mathbf{O}, \mathcal{M}) = p(\mathbf{O}|\mathcal{M}) - \frac{k}{2}\log(T)$$
(k is the number of parameters in $\mathcal{M}$) becomes negative. Penalised BIC has aso been used where the likelihood is penalised with $\lambda\frac{k}{2}\log(T)$.
- **State posteriors** : Increase the components of states for which the state posterior (here state j) is small. One possible implementation is to examine
$$\frac{\exp(\Sigma_{t=1}^{T} L_j(t)\log(b_j(\mathbf{o}_t)))}{\Sigma_{i=1}^{N_T}\exp(\Sigma_{t=1}^{T} L_j(t)\log(b_i(\mathbf{o}_t)))}$$
where N_T is the total number of states.

Building an HMM System

This describes how a large vocabulary cross-word triphone system may be built.

1. Using best previous models (eg TIMIT models) to obtain phone alignments.
2. Build single Gaussian monophone models (typically 4 re-estimation iterations).
3. "Clone" monophones for every cross-word triphone context seen in the training data.
4. Build single Gaussian unclustered triphone models (typically 2 re-estimation iterations).
5. Perform state-level decision-tree clustering - generates initial single-component state-clustered triphones
6. Train single-component models (typically 4 re-estimation iterations)
7. Mix-up and train (typically 4 re-estimation iterations at each level)
   ($1 \to 2$, $2 \to 3$, $3 \to 5$, $5 \to 7$, $7 \to 10$, $10 \to 12$)

Final system is a **12-component state-clustered cross-word triphone** system.

A Wall Street Journal System

System details:

1. **Training Data**: Wall Street Journal training data. The full training data set contains 284 speakers, each uttering between 50 and 150 sentences, for a total of 36,000 sentences (about 66 hours of speech).
2. **Parameterisation**: 12 MFCCs, normalised log-energy, delta and delta-delta parameters. Cepstral Mean Normalisation performed on a per-sentence level.
3. **Acoustic Models**: 12-component state-clustered cross-word triphones (6399 distinct speech states - about 6 million parameters). Two forms were used *Gender-Independent* and *Gender-Dependent*.
4. **Vocabulary**: 65,000 word vocabulary, selected by frequency count (including names). Multiple pronunciations used.
5. **Language Model**: Trigram language model.
6. **Test Set**: Unlimited vocabulary, two sets of 20 unknown speakers uttering about 15 sentences each (the 1994 DARPA H1 development and evaluation sets), "clean" environment.

System	WER (%)	
	Dev. Data	Eval Data
Gender-Independent	9.43	9.94
Gender-Dependent	9.06	9.39