Training HMMs with shared parameters

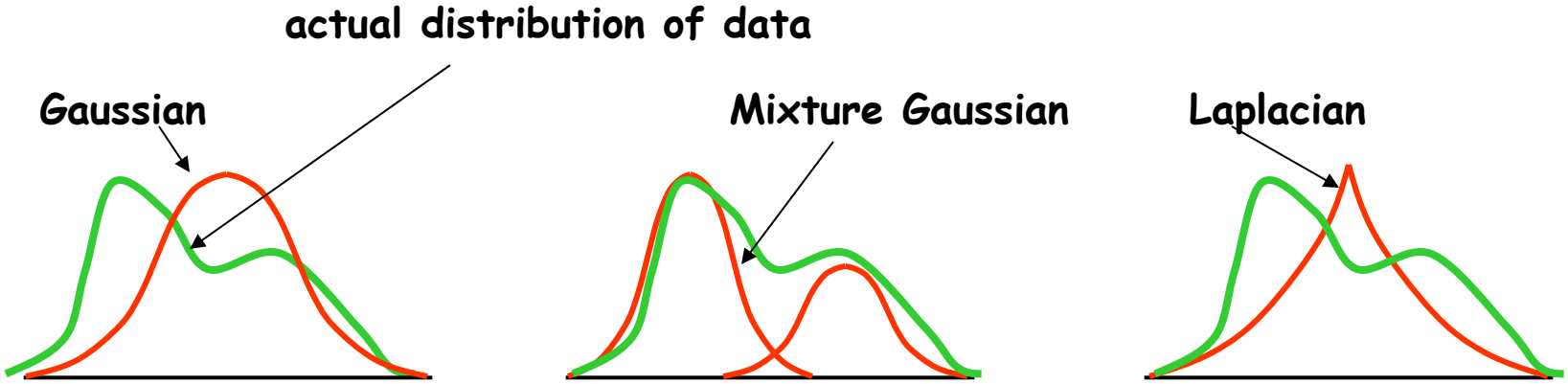
Class 25, 22 apr 2013

The problem of estimating state output distributions

- In a sub-word unit based ASR system, we may have to learn the HMMs for several thousand sub-word units
  - Each HMM has multiple states
  - Each HMM has a transition matrix
- As a result, we my have have to learn the state output distributions for tens or hundreds of thousands of HMM states
  - And also several thousand transition matrices
- The performance of the speech recognition system depends critically on how well state output distributions are modeled
  - And on how well the model parameters are learned

Modeling state output distributions

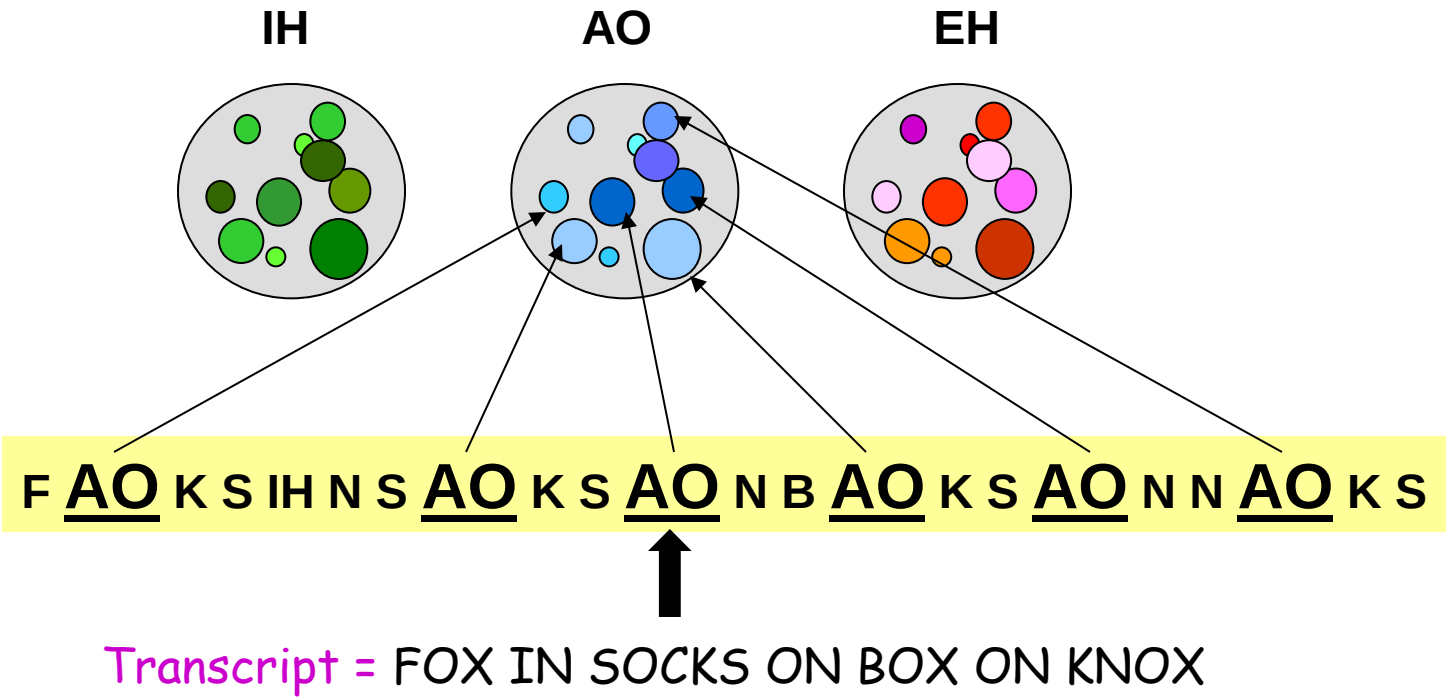
- The state output distributions might be anything in reality. We model these using various simple densities
  - Gaussian
  - Mixture Gaussian
  - Other exponential densities
- The models must be chosen such that their parameters can be easily estimated.
  - If the density model is inappropriate for the data, the HMM will be a poor statistical model
  - Gaussians are imperfect models for the distribution of cepstral features
    - Gaussians are very poor models for the distribution of power spectra
- Empirically, the most effective model has been found to be the mixture Gaussian density



The problem of estimating state output distributions

- The parameters required to specify a mixture of K Gaussians includes K mean vectors, K covariance matrices, and K mixture weights
  - All of these must be learned from training data
- A recognizer with tens (or hundreds) of thousands of HMM states will require hundreds of thousands (or millions) of parameters to specify all state output densities
  - If state output densities are modeled by mixture Gaussians
- Most training corpora cannot provide sufficient training data to learn all these parameters effectively
  - Parameters for the state output densities of sub-word units that are never seen in the training data can never be learned at all

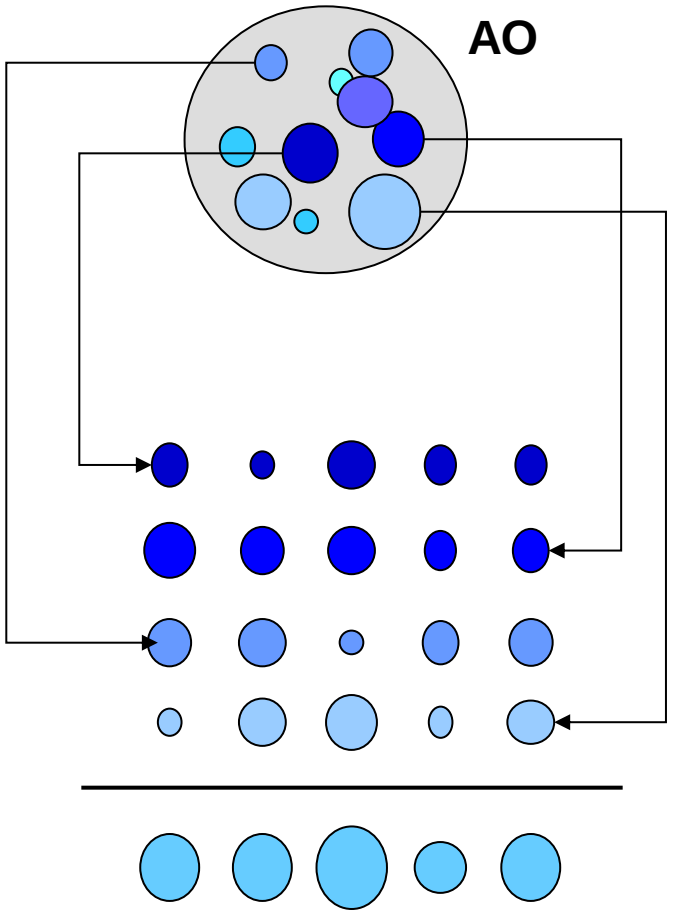
Training models for a sound unit



To train the HMM for a sub-word unit, data from all instances of the unit in the training corpus are used to estimate the parameters

Training CI models for a sound unit

Schematic example of data usage for training a 5-state HMM

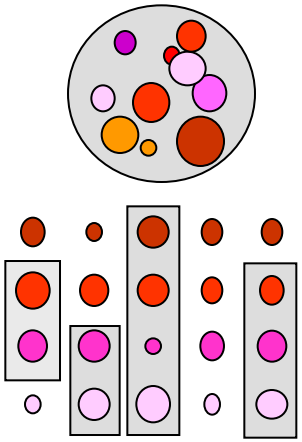


HMM for the CI unit AO

- Indiscriminate grouping of vectors of a unit from different locations in the corpus results in Context-Independent (CI) models

Gather data from separate instances, assign data to states, aggregate data for each state, and find the statistical parameters of each of the aggregates

Training sub-word unit models with shared parameters

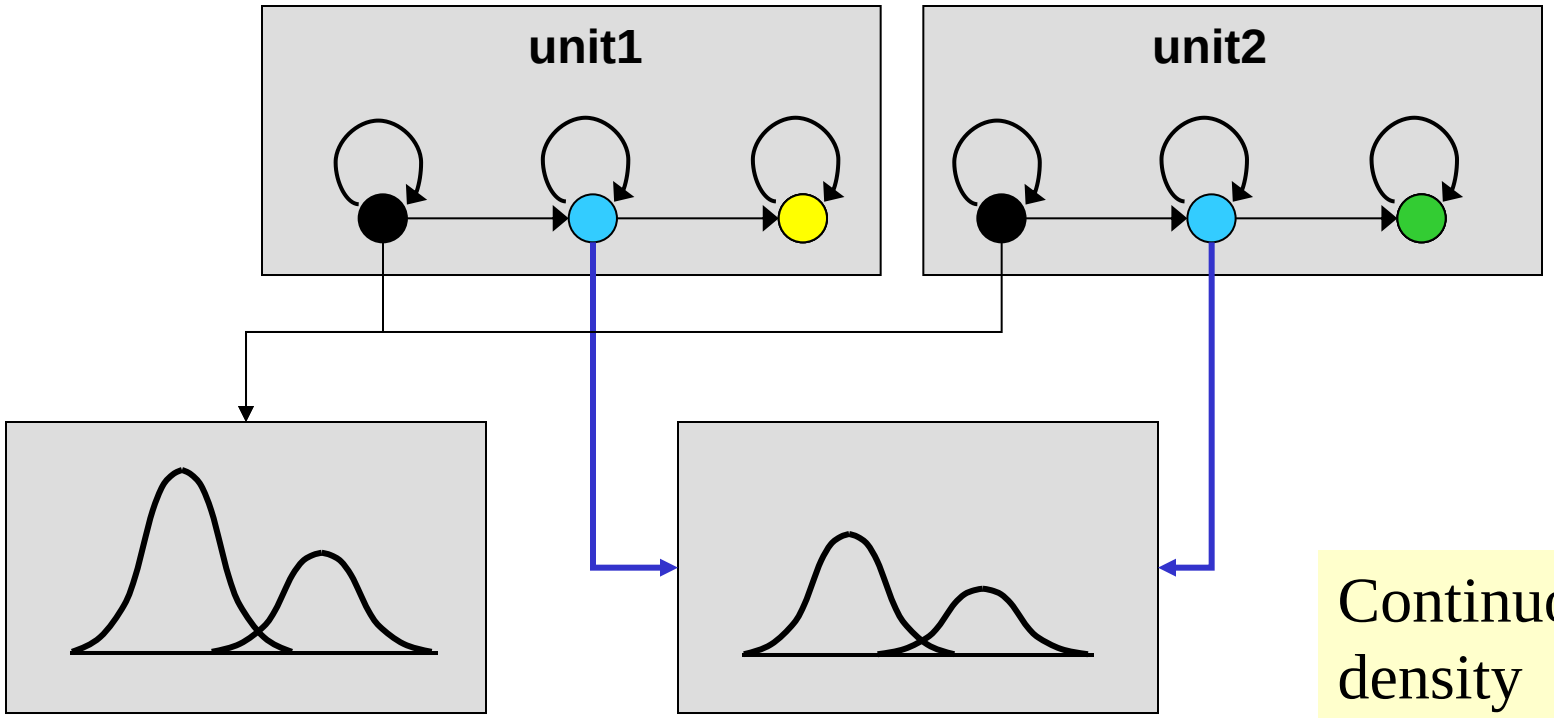


- Context based grouping of observations results in finer, Context-Dependent (CD) models
- CD models can be trained just like CI models, if no parameter sharing is performed
  - The number of subword units in a language is usually very large
- Usually insufficient training data to learn all subword HMMs properly (*Typically subword units are triphones*)
  - Parameter estimation problems

- Parameter sharing is a technique by which several similar HMM states share a common set of HMM parameters
- Since the shared HMM parameters are now trained using the data from all the similar states, there are more data available to train any HMM parameter
  - As are result HMM parameters are well trained
  - This tradeoff is that the estimated parameters cannot discriminate between the “tied” states

Sharing parameters

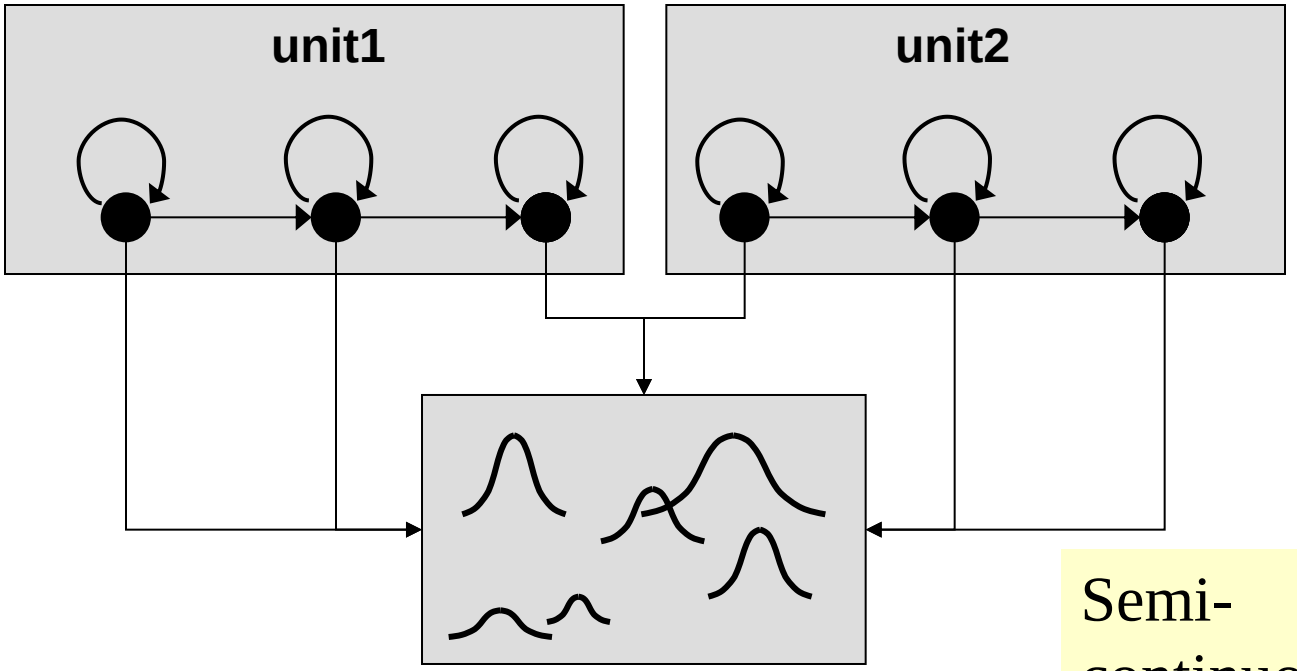
Individual states may share the same mixture distributions



Continuous density HMMs with tied states

Sharing parameters

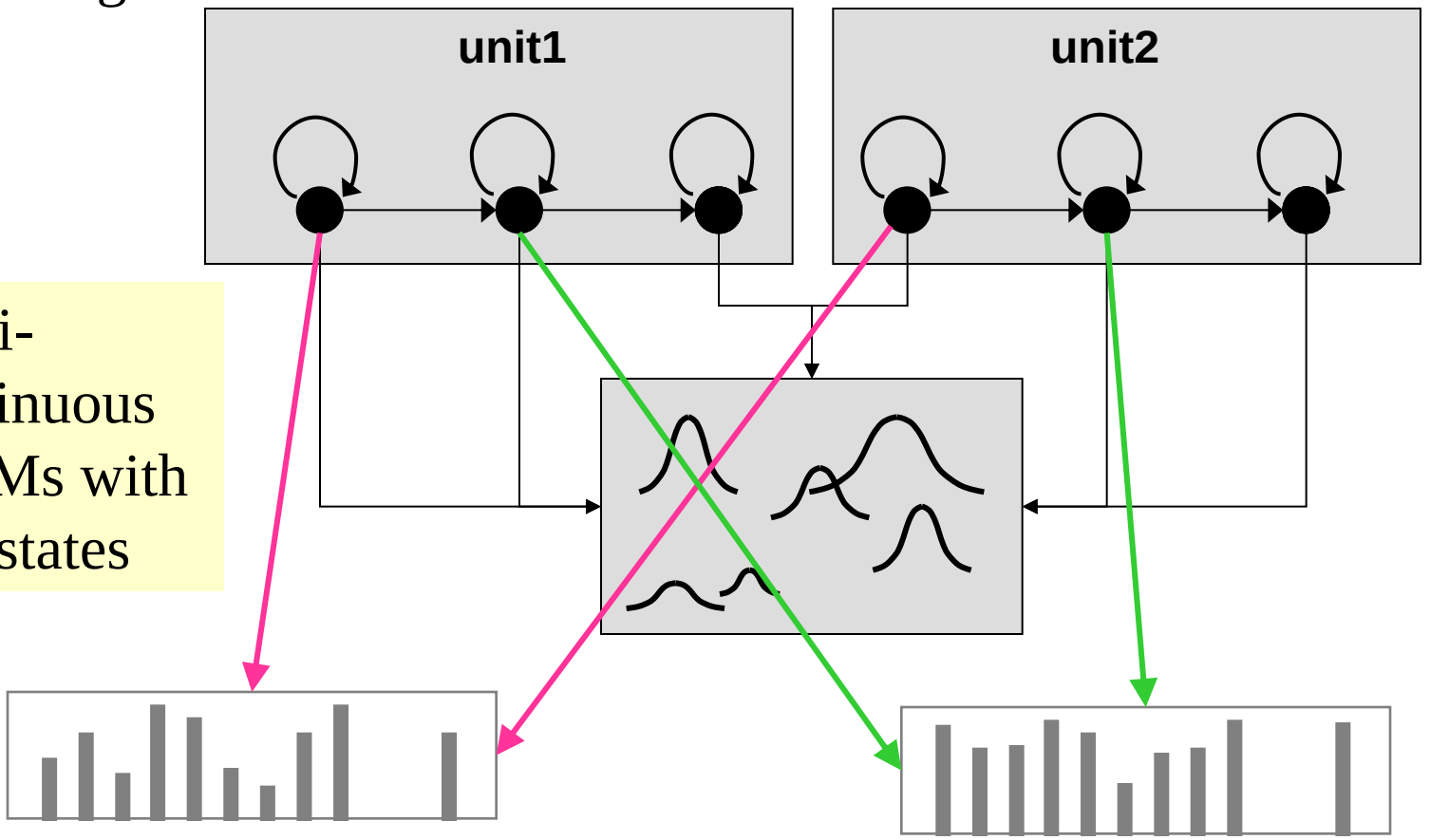
Mixture Gaussian state densities: all states may share the same Gaussians, but with different mixture weights



Semi-continuous HMMs

Sharing parameters

Mixture Gaussian state densities: all states may share the same Gaussians, but with state-specific mixture weights, and then share the weights as well



Semi-continuous HMMs with tied states

Deciding how parameters are shared

- Partly a design choice
  - Semi-continuous HMMs, vs. phonetically tied-semi-continuous HMMs vs. continuous density HMMs
- Automatic techniques
  - Data-driven Clustering
    - Group HMM states together based on the similarity of their distributions, until all groups have sufficient data
      - **The densities used for grouping are poorly estimated in the first place**
      - **Has no estimates for unseen sub-word units**
      - **Places no restrictions on HMM topologies etc.**
  - Decision trees
    - Clustering based on expert-specified rules. The selection of rule is data driven
      - **Based on externally provided rules. Very robust if the rules are good**
      - **Provides a mechanism for estimating unseen sub-word units**
      - **Restricts HMM topologies**

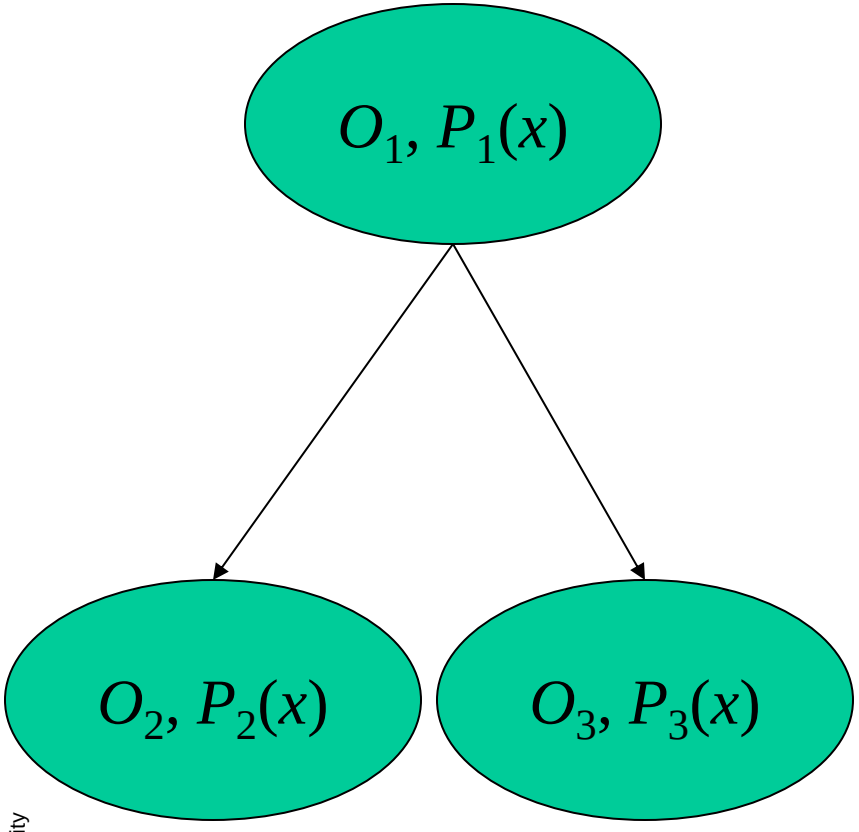
Decision Trees

- Related to Classification and Regression Trees (Leo Breiman), and ID3 and C4.5 (Ross Quinlan)
  - Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984), **Classification** and **Regression Trees**, Wadsworth Inc., Belmont, CA.
  - Quinlan, J. (1993) **C4.5: programs for Machine Learning**, Morgan Kaufmann

- Basic principle: Recursively partition a data set to maximize a pre-specified objective function
  - The actual objective function used is dependent on the specific decision tree algorithm

- The objective is to separate the data into increasingly "pure" subsets, such that most of the data in any subset belongs to a single class
  - In our case the "classes" are HMM states

Decision trees: splitting nodes by increase in log likelihood



$$O_1 = O_2 \bigcup O_3$$

- The parent set O_1 has a distribution $P_1(x)$
  - The total log likelihood of all observations in O_1 on the distribution of O_1 is

$$L(O_1) = \sum_{x \in O_1} \log(P_1(x))$$

- The child set O_2 has a distribution $P_2(x)$
  - The total log likelihood of all observations in O_2 on the distribution of O_2 is

$$L(O_2) = \sum_{x \in O_2} \log(P_2(x))$$

- The child set O_3 has a distribution $P_3(x)$
  - The total log likelihood of all observations in O_3 on the distribution of O_3 is

$$L(O_3) = \sum_{x \in O_3} \log(P_3(x))$$

- The total increase in set-conditioned log likelihood of observations due to partitioning O_1 is

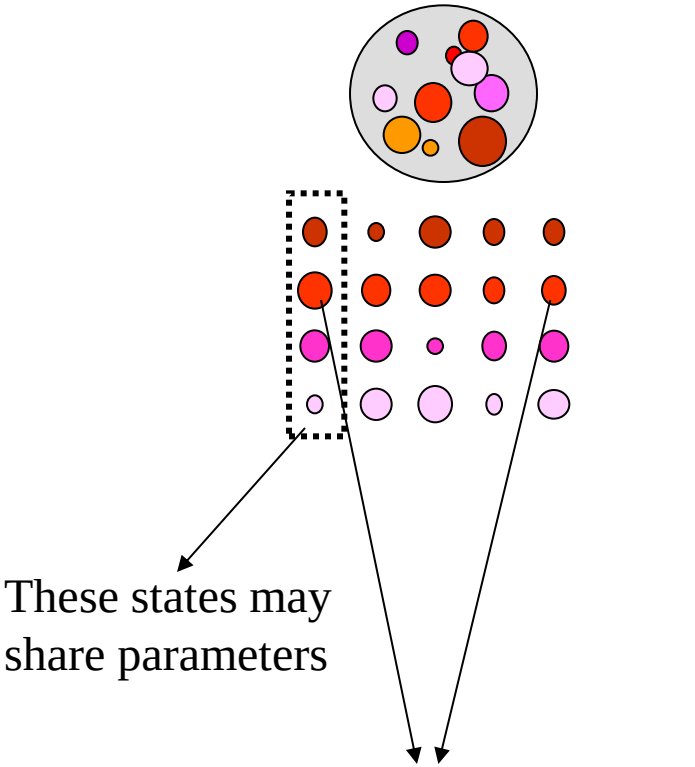
$$L(O_1) + L(O_2) - L(O_3)$$

- Partition O_1 such that the increase in log likelihood is maximized
  - Recursive perform this partition of each of the subsets to form a tree

Decision trees for parameter sharing

- Identify set of states that can potentially be merged
  - Based on rules, rather than data, in order to enable prediction of distributions of unseen sub-word units
- Partition the union of the data in all the states recursively using a decision tree procedure
  - Ensuring that entire states go together during the partitioning
  - Terminate the recursion at any leaf when partitioning the data at the leaf will result in children with insufficient data for parameter estimation
  - Alternately, grow the tree until each state is at a separate leaf, and prune the tree backwards until all leaves have sufficient data for parameter estimation
- The leaves of the resulting tree will include many states. All states in a leaf will share distribution parameters.

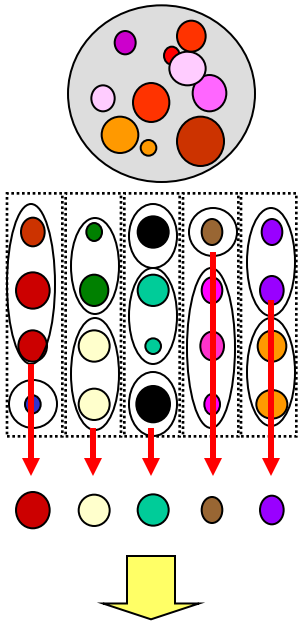
Heuristic for deciding which states can be grouped at the root of a tree



- A common heuristic is to permit states with identical indices in the HMMs for different N-phones for the same base phone to share parameters.
  - E.g., the first state of all triphones of the kind AA(*,*) are allowed to share distribution parameters
- Within any index, states are be further clustered using a decision tree
  - All states within each cluster share parameters
  - In the worst case, where all N-phone states with a common index share a common distribution results in simple CI phonemes
- States with different indices are not allowed to share parameters
- This heuristic enables the "synthesis" of HMMs for N-phones that were never seen in the training data
  - This only works for N-phones for which base phones can be identified

Synthesizing HMMs for unseen subword units

HMMs for seen subword units

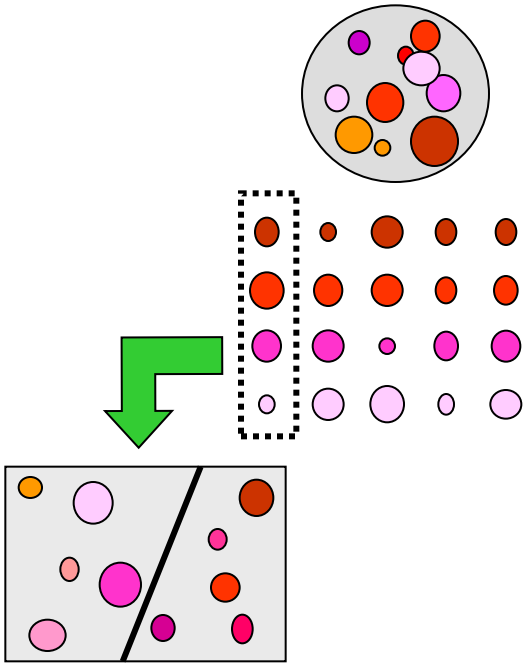


HMM for unseen subword unit. The state output distribution for any state is identical to the distribution of a cluster of observed states

All N-phones with a common basephone are assumed to have identical transition matrices

- Each HMM state of every unseen N-phone is assumed to be clustered with some subset of states of the same index, that belong to other N-phones of the same base phone
- The state output distribution for each HMM state of the unseen Nphone is set to be identical to the distribution for the cluster it belongs to
- Estimating HMMs for unseen Nphones simply involves identifying the state clusters that their states belong to
  - The clusters are identified based on expert-specified rules
- For this technique to work, the HMMs for all Nphones of a given basephone must have identical numbers of states and identical topologies

Clustering states with decision trees



- All states with a common index are initially grouped together at the root node
- Each node is then recursively partitioned
- All states in the leaves of the decision tree share parameters
- A separate decision tree is built for every state index for every base phone
- The decision tree procedure attempts to maximize the loglikelihood of the training data

The assignment of vectors to states can be done using previously trained CI models or with CD models that have been trained without parameter sharing

The expected log-likelihood of a vector drawn from a Gaussian distribution with mean μ and variance C is

$$E\left[\log\left(\frac{1}{\sqrt{(2\pi)^d\,|C|}}e^{-0.5(x-\mu)^T C^{-1}(x-\mu)}\right)\right]$$

Expected log-likelihood of a Gaussian random vector

Expected log-likelihood of a vector drawn from a Gaussian distribution

$$E\left[\log\left(\frac{1}{\sqrt{(2\pi)^d\,|C|}}e^{-0.5(x-\mu)^T C^{-1}(x-\mu)}\right)\right]=$$

$$E\left[-0.5(x-\mu)^T C^{-1}(x-\mu)-0.5\log\left((2\pi)^d\,|C|\right)\right]=$$

$$-0.5E\left[(x-\mu)^T C^{-1}(x-\mu)\right]-0.5E\left[\log\left((2\pi)^d\,|C|\right)\right]=$$

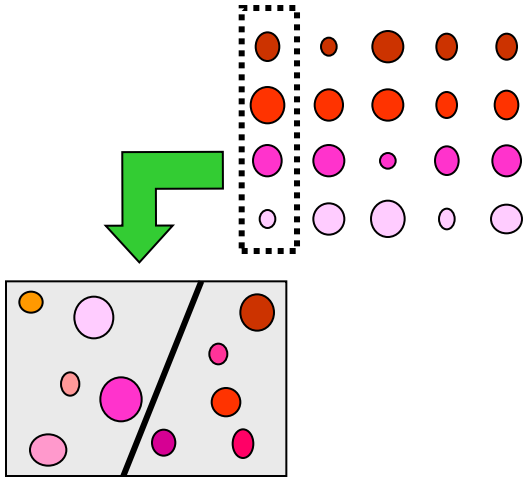
$$-0.5d-0.5\log\left((2\pi)^d\,|C|\right)$$

- **This is a function only of the variance of the Gaussian**
- **The expected log-likelihood of a set of N vectors is**

$$-0.5Nd-0.5N\log\left((2\pi)^d\,|C|\right)$$

Partitioning each node of the decision tree

Observation vectors are partitioned into groups to maximize within class likelihoods



If we partition a set of ***N*** vectors with mean **m** and variance ***C*** into two sets of vectors of size ***N*1** and ***N*2** , with means **m1** and **m2** and variances ***C*1** and ***C*2** respectively, the total expected log-likelihood of the vectors after splitting becomes

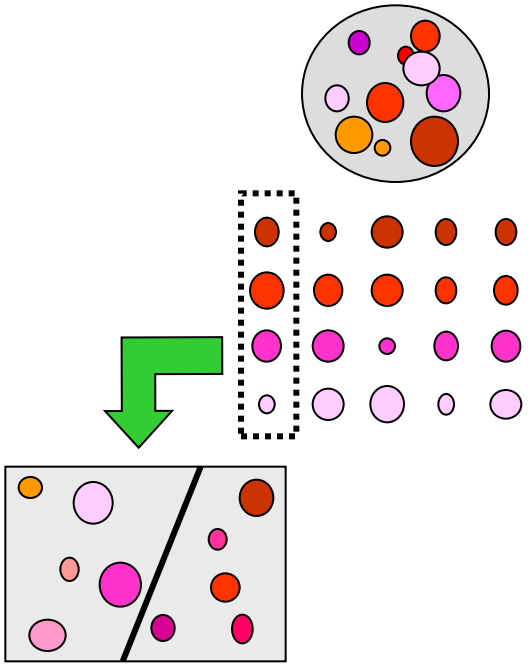
$$-0.5N_1 d - 0.5N_1 \log\left((2\pi)^d \mid C_1 \mid\right) - 0.5N_2 d - 0.5N_2 \log\left((2\pi)^d \mid C_2 \mid\right)$$

The total log-likelihood has increased by

$$0.5N \log\left((2\pi)^d \mid C \mid\right) - 0.5N_1 \log\left((2\pi)^d \mid C_1 \mid\right) - 0.5N_2 \log\left((2\pi)^d \mid C_2 \mid\right)$$

Partitioning each node of the decision tree

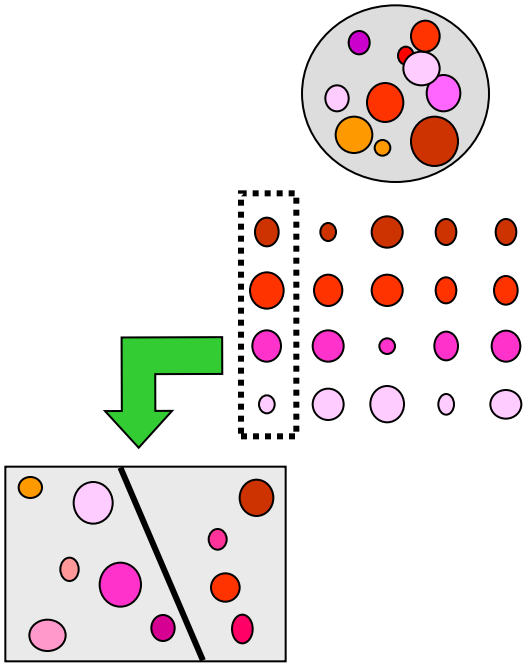
Ideally, every possible partition of vectors into two clusters must be evaluated



Partitioning is performed such that all vectors belonging to a single Nphone must together

Partitioning each node of the decision tree

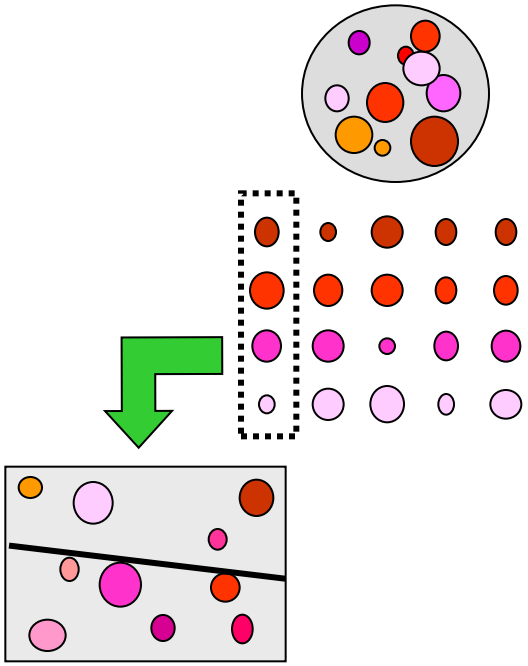
Ideally, every possible partition of vectors into two clusters must be evaluated



Partitioning is performed such that all vectors belonging to a single Nphone must together

Partitioning each node of the decision tree

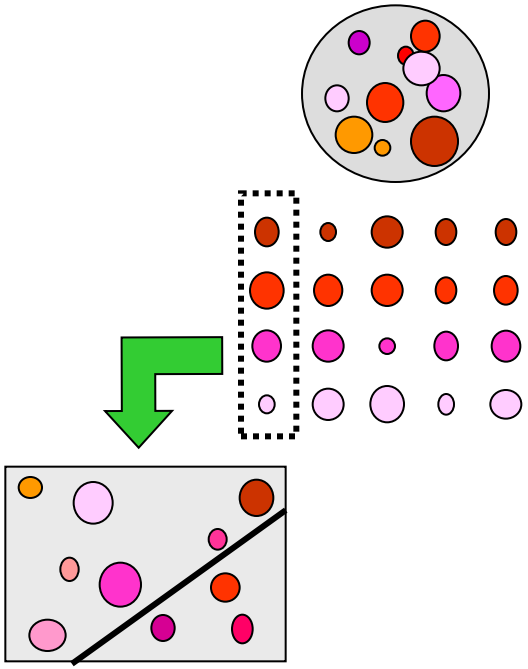
Ideally, every possible partition of vectors into two clusters must be evaluated



Partitioning is performed such that all vectors belonging to a single Nphone must together

Partitioning each node of the decision tree

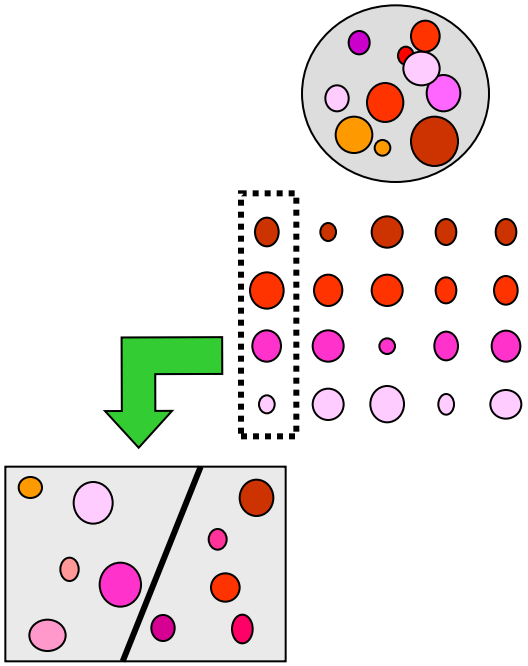
Ideally, every possible partition of vectors into two clusters must be evaluated



Partitioning is performed such that all vectors belonging to a single Nphone must together

Partitioning each node of the decision tree

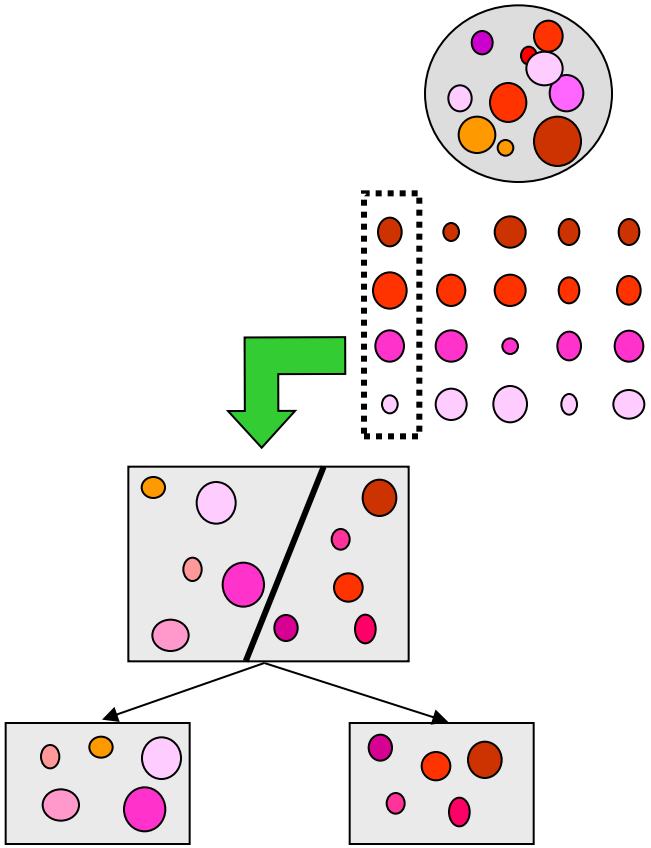
Ideally, every possible partition of vectors into two clusters must be evaluated



Partitioning is performed such that all vectors belonging to a single Nphone must together

Partitioning each node of the decision tree

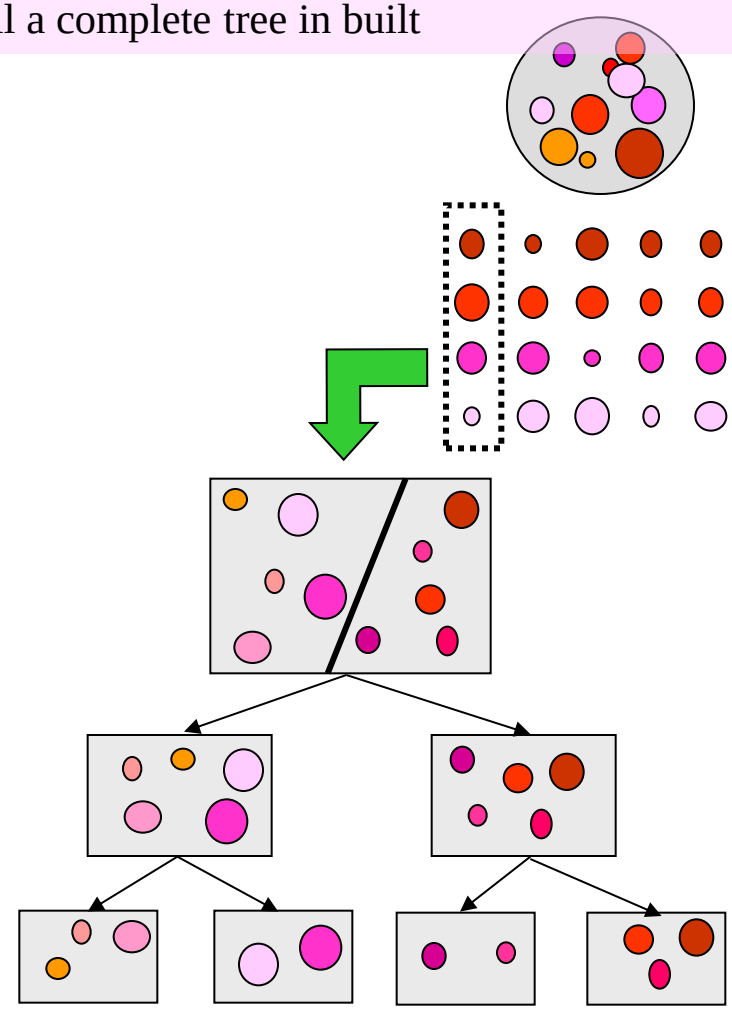
Ideally, every possible partition of vectors into two clusters must be evaluated, the partition with the maximum increase in log likelihood must be chosen



Partitioning is performed such that all vectors belonging to a single Nphone must together

Partitioning each node of the decision tree

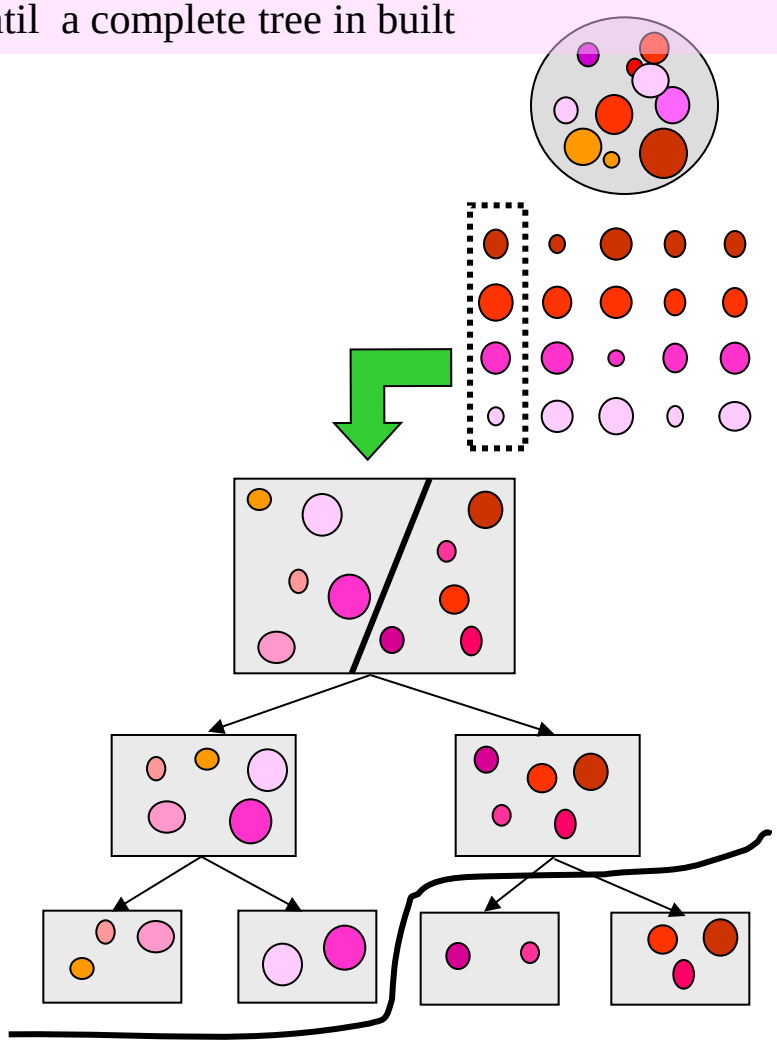
Ideally, every possible partition of vectors into two clusters must be evaluated, the partition with the maximum increase in log likelihood must be chosen, and the procedure must be recursively continued until a complete tree in built



Partitioning is performed such that all vectors belonging to a single Nphone must together

Partitioning each node of the decision tree

Ideally, every possible partition of vectors into two clusters must be evaluated, the partition with the maximum increase in log likelihood must be chosen, and the procedure must be recursively continued until a complete tree in built



Partitioning is performed such that all vectors belonging to a single Nphone must together

The trees will have a large number of leaves

All trees must then be collectively pruned to retain only the desired number of leaves. Each leaf represents a tied state (sometimes called a *senone*)

All the states within a leaf share distribution parameters. The shared distribution parameters are estimated from all the data in all the states at the leaf

Sharing parameters: evaluating the partitions

- 2^{n-1} possible partitions for n vector groups. Exhaustive evaluation too expensive
- Linguistic questions are used to reduce the search space
- Linguistic questions are pre-defined phone classes. Candidate partitions are based on whether a context belongs to the phone class or not
  - Example:

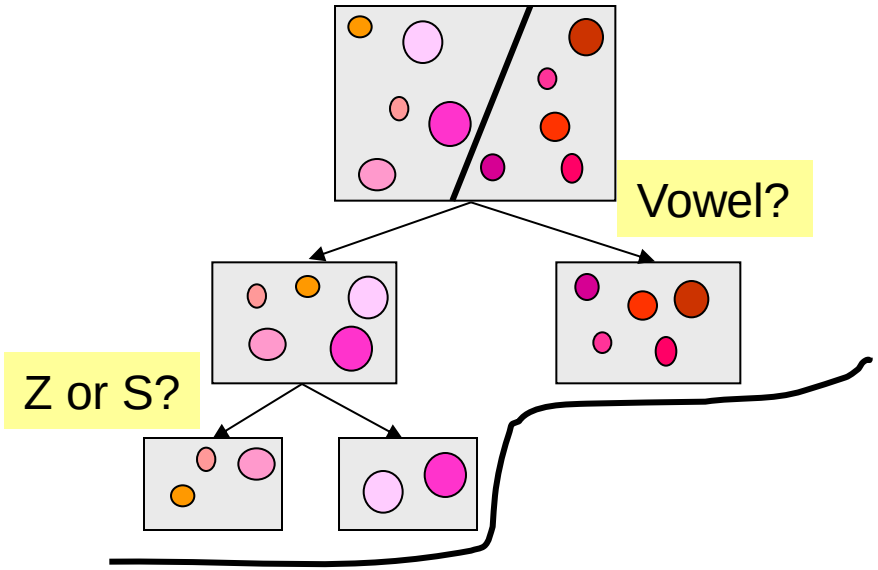
Class1 EY EH IY IH AX AA R W N V Z
Class2 F Z SH JH ZH
Class3 T D K P B
Class4 UW UH OW R W AY
Class5 TH S Z V F
Class6 IH IY UH UW
Class7 W V F
Class8 T K
Class9 R W
…

Partitioning with linguistic questions

- Partitions are derived based only on answers to linguistic questions such as:
  - Does the left context belong to “class1”
  - Does the right context belong to “class1”
  - Does the left context NOT belong to “class1”
  - Does the right context NOT belong to “class1”
- The set of possible partitions based on linguistic questions is restricted and can be exhaustively evaluated
- Linguistic classes group phonemes that share certain spectral characteristics, and may be expected to have similar effect on adjacent phonemes.
  - Partitioning based on linguistic questions imparts “expert knowledge” to an otherwise data-driven procedure.

Composing HMMs for unseen Nphones

- Every non-leaf node in the decision tree has a question associated with it
  - The question that was eventually used to partition the node
- The questions are linguistic questions
  - Since partitioning is exclusively performed with linguistic questions
- Even unseen Nphones can answer the questions
  - They can therefore be propagated to the leaves of the decision trees
- The state output distribution for any state of an unseen Nphone is obtained by propagating the Nphone to a leaf of the appropriate decision tree for its base phone
  - The output distribution for all the states in the leaf is also assigned to the unseen state

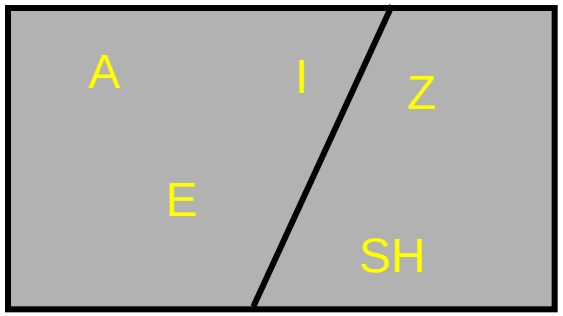


Linguistic questions

Linguistic questions must be meaningful in order to deal effectively with unseen triphones

Linguistic questions effectively substitute expert knowledge for information derived from data

For effective prediction of the distributions of unseen subword units, the linguistic questions must represent natural groupings of acoustic phenomena



Meaningful Linguistic Questions?

Left context: (A,E,I,Z,SH)

ML Partition: (A,E,I) (Z,SH)

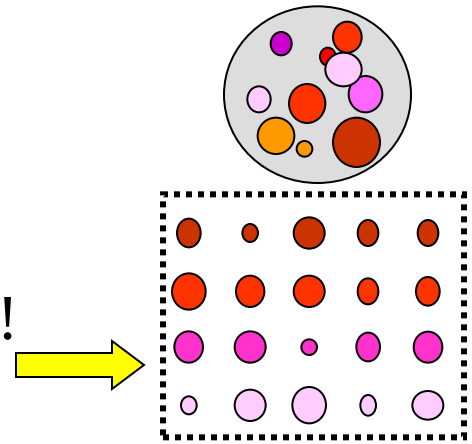
(A,E,I) vs. Not(A,E,I)

(A,E,I,O,U) vs. Not(A,E,I,O,U)

(A,E,I,O,U) vs. Not (A,E,I,O,U) represents a natural grouping of phonemes. The other groupings are not natural.

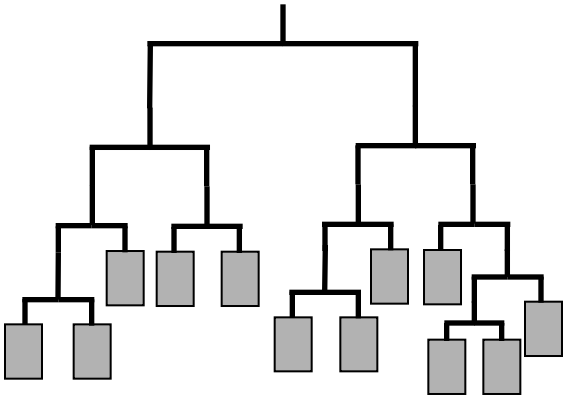
Context Dependent Untied (CD untied) training for building decision trees

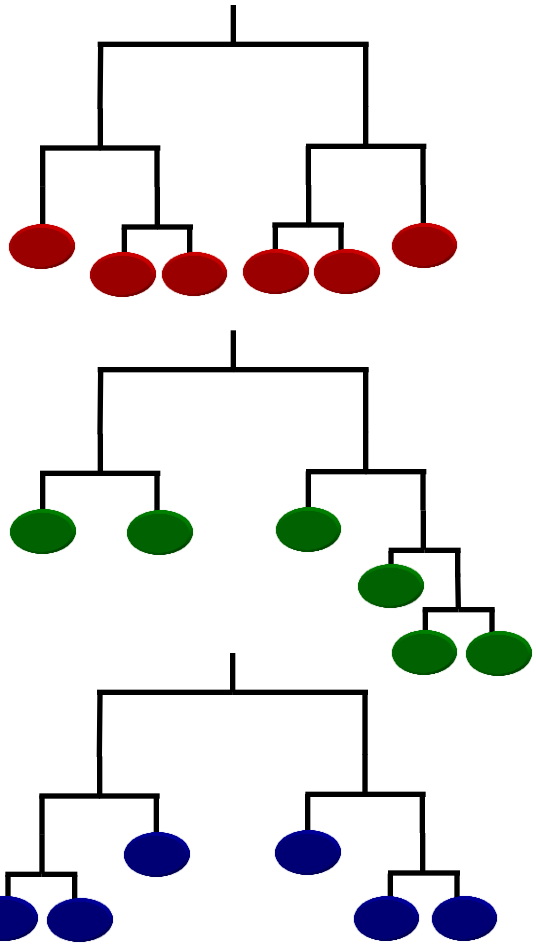
Before decision trees are built, all these HMMs must be trained !



- Train HMMs for all triphones (Nphones) in the training data with no sharing of parameters
- Use these "untied" HMM parameters to build decision trees
  - For every state of every base phone
- This stage of training is the "context-dependent untied training"

Prining decision trees before state tying





There are several ways of pruning a tree to obtain a given number of leaves (6 in this example)

Each leaf represents a cluster of triphone states that will share parameters. The leaves are called *tied states* or *senones*

Senones are building blocks for triphone HMMs. A 5-state HMM for a triphone will have 5 senones

Context Dependent tied state (CD tied) models

- Final CD models are trained for the triphones using the tied states
- Training finer models: Train mixtures of Gaussians for each senone
  - All states sharing the senone inherit the entire mixture
  - Mixtures of many Gaussians are trained by iterative splitting of Gaussians
- Gaussian splitting is performed for continuous models
  - Initially train single Gaussian state distributions
  - Split Gaussian with largest mixture weight by perturbing mean vector and retrain
  - Repeat splitting until desired number of Gaussians obtained in the Gaussian mixture state distributions

Other forms of parameter sharing

- Ad-hoc sharing: sharing based on human decision
  - Semi-continuous HMMs – all state densities share the same Gaussians
  - This sort of parameter sharing can coexist with the more refined sharing described earlier.

Baum-Welch with shared state parameters

– Update formulae are the same as before, except that the numerator and denominator for any parameter are also aggregated over all the states that share the parameter

Mean of k^{th} Gaussian of any state in the set of states Θ that share the k^{th} Gaussian

$$\mu_k^{\Theta} = \frac{\sum_{s\in\Theta}\sum_{utterance}\sum_{t}\gamma_{utt}(s,t)P(k\mid x_t,s)x_t}{\sum_{s\in\Theta}\sum_{utterance}\sum_{t}\gamma_{utt}(s,t)P(k\mid x_t,s)}$$

Covariance of k^{th} Gaussian of any state in the set of states Θ that share the k^{th} Gaussian

$$C_k^{\Theta} = \frac{\sum_{s\in\Theta}\sum_{utterance}\sum_{t}\gamma_{utt}(s,t)P(k\mid x_t,s)(x_t-\mu_k)(x_t-\mu_k)^T}{\sum_{s\in\Theta}\sum_{utterance}\sum_{t}\gamma_{utt}(s,t)P(k\mid x_t,s)}$$

Mixture weight of k^{th} Gaussian of any state in the set of states Θ that share a Gaussian mixture

$$P_{\Theta}(k) = \frac{\sum_{s\in\Theta}\sum_{utterance}\sum_{t}\gamma_{utt}(s,t)P(k\mid x_t,s)}{\sum_{s\in\Theta}\sum_{utterance}\sum_{t}\sum_{j}\gamma_{utt}(s,t)P(j\mid x_t,s)}$$

Overall Training Process

- Train CI models
- Train CD (triphone) models with no parameter sharing
  - Initialize with CI models
- Compute decision trees
  - Determine parameter sharing
- Retrain CD models
  - With parameter sharing
  - Initialize all tied state distributions with state output distribution of single-Gaussian CI models
  - After each round of training has converged, split Gaussians to increase no. of Gaussians/tied state