Modelling speech with HMMs – Context-dependent phone models

Steve Renals

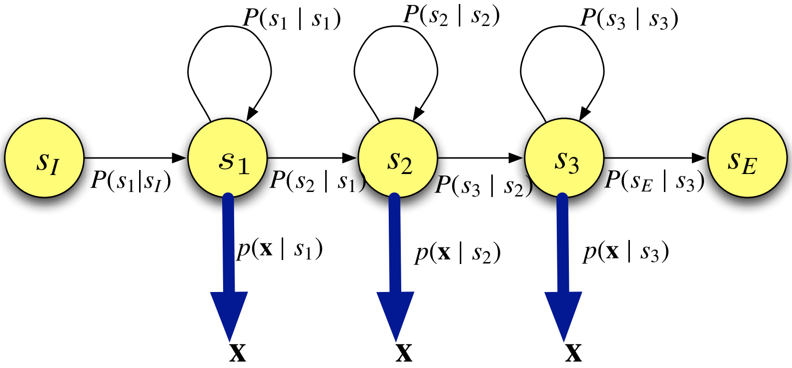
Automatic Speech Recognition
ASR Lectures 6&8
28 January, 4 February 2016

Overview

Phone models

- Modelling phones with HMMs
- The need to model phonetic context
- Triphone models
- Smoothing – interpolation and backing-off
- Parameter sharing – tied mixtures, generalised triphones, state clustering
- Choosing which states to share – phonetic decision trees

Recap: Continuous Density HMM

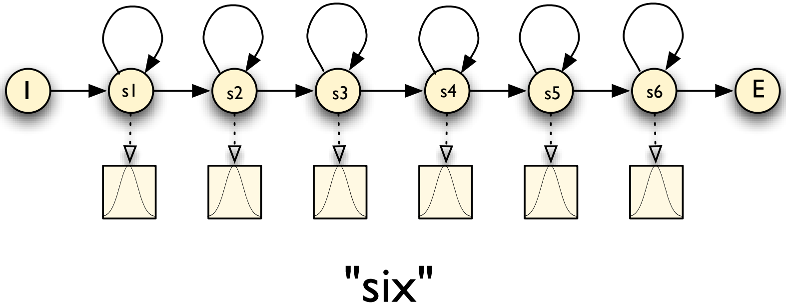


Probabilistic finite state automaton

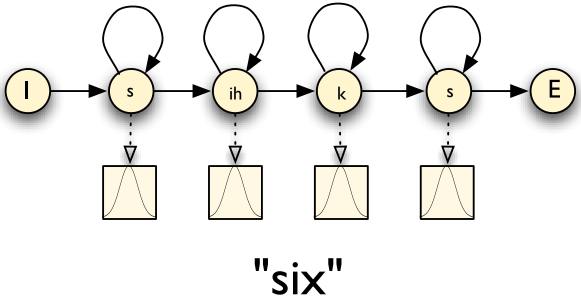
Paramaters $\boldsymbol{\lambda}$:

- Transition probabilities: $a_{kj} = P(s_j \mid s_k)$
- Output probability density function: $b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j)$

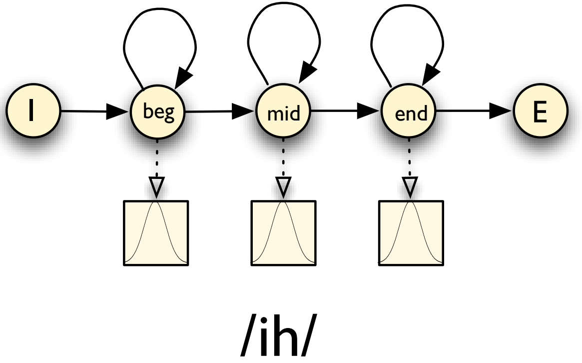
Whole word models



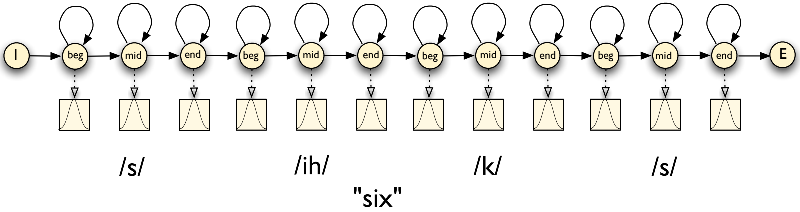
One state per phone models



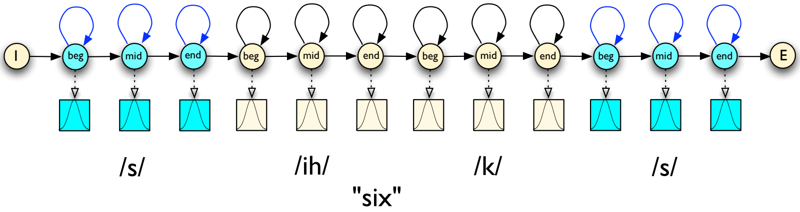
Three-state phone models



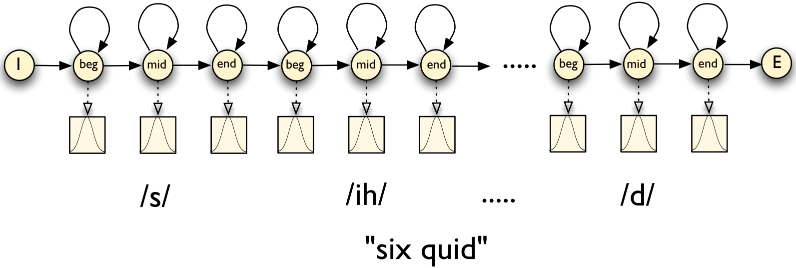
Word model made of phone models



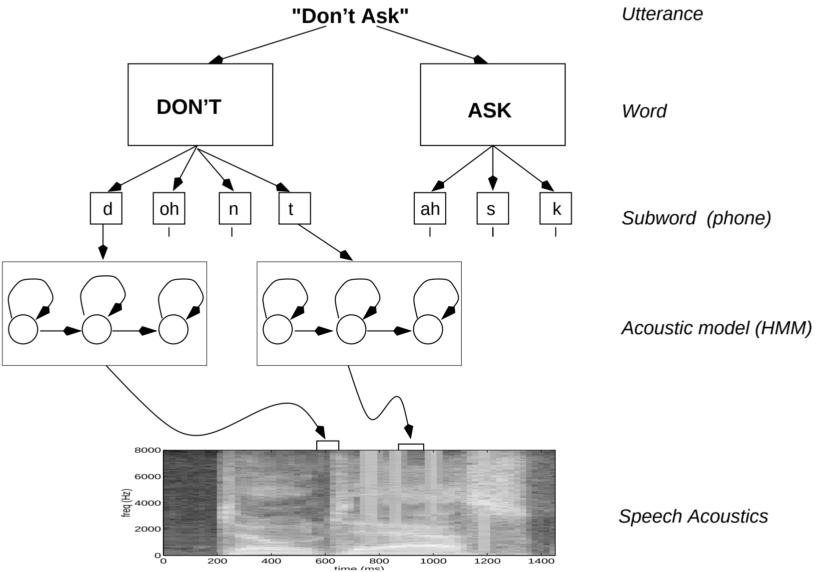
Word model made of phone models



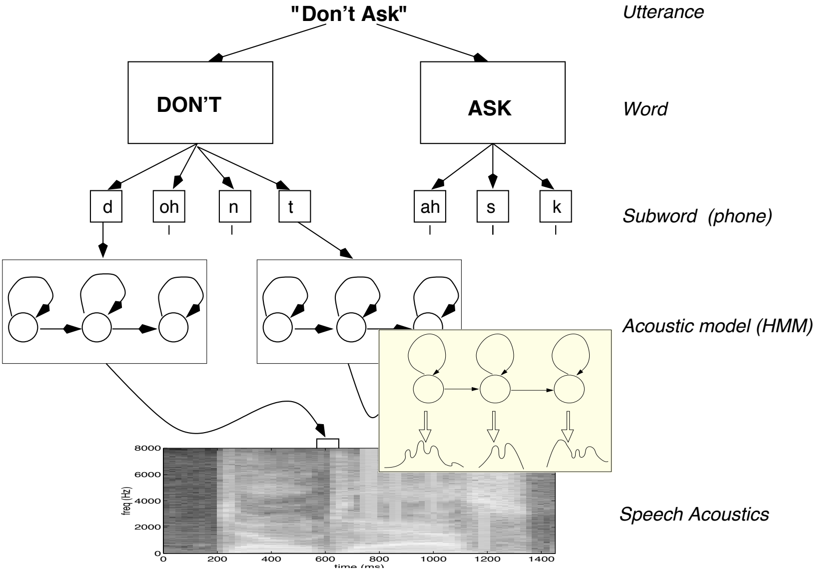
Word sequence models



Hierarchical Modelling in Speech Recognition



Hierarchical Modelling in Speech Recognition



Phonetic Context

- Context The acoustic phonetic context of a speech unit has an effect on its acoustic realization

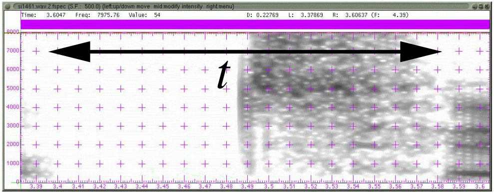
Phonetic Context

- Context The acoustic phonetic context of a speech unit has an effect on its acoustic realization
- Coarticulation the place of articulation for one speech sound depends on a neighbouring speech sound.

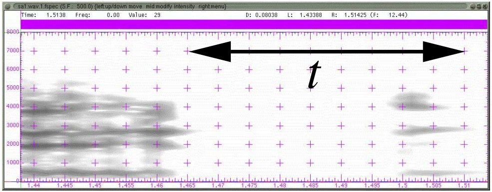
Phonetic Context

- Context The acoustic phonetic context of a speech unit has an effect on its acoustic realization
- Coarticulation the place of articulation for one speech sound depends on a neighbouring speech sound.
- Consider /n/ in ten and tenth
  - dental in ten
  - alveolar in tenth

Phonetic Context Example



"tube"



"suit"

Modelling Context

- Subword units Individual phone units need to deal with a lot of variability
  - Use longer units that incorporate context, eg: diphones, demisyllables, syllables
  - Use multiple models for each: context-dependent phone models
  - Context-dependent phones are termed allophones of the parent phone

Modelling Context

- Subword units Individual phone units need to deal with a lot of variability
  - Use longer units that incorporate context, eg: diphones, demisyllables, syllables
  - Use multiple models for each: context-dependent phone models
  - Context-dependent phones are termed allophones of the parent phone
- Pronunciations
  - "did you" `d ih jh y ah`
  - "around this" `ix r aw n ih s`

Divide and conquer

- Context-dependent models are more specific than context-independent models
- Increase the detail of modelling by extending the state space – but by defining multiple context dependent models, rather than more complex context independent models
- Divide and conquer: as more context-dependent models are defined, each one becomes responsible for a smaller region of the acoustic-phonetic space
- Let the data tell us how many contexts to model

Context-dependent phone models

- Triphones Each phone has a unique model for each left and right context. Represent a phone x with left context l and right context r as l-x+r

Context-dependent phone models

- Triphones Each phone has a unique model for each left and right context. Represent a phone `x` with left context `l` and right context `r` as `l-x+r`
- Word-internal triphones Only take account of context within words, so "don't ask" is represented by:
  `sil d+oh d-oh+n oh-n+t n-t ah+s ah-s+k s-k sil`
  Word internal triphones result in far fewer models than cross-word models, and enable the subword sequence for a word to be known independent of the neighbouring words. But: context is not well-modelled at word boundaries.

Context-dependent phone models

- Triphones Each phone has a unique model for each left and right context. Represent a phone `x` with left context `l` and right context `r` as `l-x+r`
- Word-internal triphones Only take account of context within words, so "don't ask" is represented by:
  `sil d+oh d-oh+n oh-n+t n-t ah+s ah-s+k s-k sil`
  Word internal triphones result in far fewer models than cross-word models, and enable the subword sequence for a word to be known independent of the neighbouring words. But: context is not well-modelled at word boundaries.
- Cross-word triphones "don't ask" is represented by:
  `sil sil-d+oh d-oh+n oh-n+t n-t+ah t-ah+s ah-s+k s-k+sil sil`
  Note that triphone context extends across words (eg unit `n-t+ah`)

Triphone models

- How many triphones are there? Consider a 40 phone system. $40^3 = 64\,000$ possible triphones. In a cross-word system maybe 50 000 can occur

Triphone models

- How many triphones are there? Consider a 40 phone system. $40^3 = 64\,000$ possible triphones. In a cross-word system maybe 50 000 can occur
- Number of parameters:
  - 50 000 three-state HMMs, with 10 component Gaussian mixtures per state: 1.5M Gaussians
  - 39-dimension feature vectors (12 MFCCs + energy), deltas and accelerations
  - Assuming diagonal Gaussians: about 790 parameters/state
  - Total about 118 million parameters!

Triphone models

- How many triphones are there? Consider a 40 phone system. $40^3 = 64\,000$ possible triphones. In a cross-word system maybe 50 000 can occur
- Number of parameters:
  - 50 000 three-state HMMs, with 10 component Gaussian mixtures per state: 1.5M Gaussians
  - 39-dimension feature vectors (12 MFCCs + energy), deltas and accelerations
  - Assuming diagonal Gaussians: about 790 parameters/state
  - Total about 118 million parameters!
- We would need a very large amount of training data to train such a system
  - to enable robust estimation of all parameters
  - to ensure that all possible triphones are observed (more than once) in the training data

Modelling infrequent triphones

The number of possible triphone types is much greater than the number of observed triphone tokens.

- Smoothing – combine less-specific and more-specific models

Modelling infrequent triphones

The number of possible triphone types is much greater than the number of observed triphone tokens.

- Smoothing – combine less-specific and more-specific models
- Parameter Sharing – different contexts share models

Modelling infrequent triphones

The number of possible triphone types is much greater than the number of observed triphone tokens.

- Smoothing – combine less-specific and more-specific models
- Parameter Sharing – different contexts share models
  - Bottom-up – start with all possible contexts, then merge

Modelling infrequent triphones

The number of possible triphone types is much greater than the number of observed triphone tokens.

- Smoothing – combine less-specific and more-specific models
- Parameter Sharing – different contexts share models
  - Bottom-up – start with all possible contexts, then merge
  - Top-down – start with a single context, then split

Modelling infrequent triphones

The number of possible triphone types is much greater than the number of observed triphone tokens.

- Smoothing – combine less-specific and more-specific models
- Parameter Sharing – different contexts share models
  - Bottom-up – start with all possible contexts, then merge
  - Top-down – start with a single context, then split
- All approaches are data driven

Smoothing: Backing off

- Basic idea Use less-specific models when there is not enough data to train a more specific one

Smoothing: Backing off

- Basic idea Use less-specific models when there is not enough data to train a more specific one
- For example if a triphone is not observed (or only a few examples are observed) use a biphone model:
  `sh-iy+l` → `iy+l`

Smoothing: Backing off

- Basic idea Use less-specific models when there is not enough data to train a more specific one
- For example if a triphone is not observed (or only a few examples are observed) use a biphone model:
  `sh-iy+l` → `iy+l`
- If only a few biphone occurrences use a monophone:
  `sh-iy+l` → `iy+l` → `iy`

Smoothing: Backing off

- Basic idea Use less-specific models when there is not enough data to train a more specific one
- For example if a triphone is not observed (or only a few examples are observed) use a biphone model:
  `sh-iy+l` → `iy+l`
- If only a few biphone occurrences use a monophone:
  `sh-iy+l` → `iy+l` → `iy`
- Use a minimum training example count to determine whether a triphone should be modelled or backed-off to a biphone (likewise for biphones)

Smoothing: Backing off

- Basic idea Use less-specific models when there is not enough data to train a more specific one
- For example if a triphone is not observed (or only a few examples are observed) use a biphone model:
  `sh-iy+l` → `iy+l`
- If only a few biphone occurrences use a monophone:
  `sh-iy+l` → `iy+l` → `iy`
- Use a minimum training example count to determine whether a triphone should be modelled or backed-off to a biphone (likewise for biphones)
- Ensures that each model is well trained

Smoothing: Backing off

- Basic idea Use less-specific models when there is not enough data to train a more specific one
- For example if a triphone is not observed (or only a few examples are observed) use a biphone model:
  `sh-iy+l` → `iy+l`
- If only a few biphone occurrences use a monophone:
  `sh-iy+l` → `iy+l` → `iy`
- Use a minimum training example count to determine whether a triphone should be modelled or backed-off to a biphone (likewise for biphones)
- Ensures that each model is well trained
- But training data is sparse (especially when cross-word triphones are used) so relatively few specific triphone models

Smoothing: Interpolation

- Basic idea Combine less-specific models with more specific models

Smoothing: Interpolation

- Basic idea Combine less-specific models with more specific models
- Interpolate the parameters of a triphone λ^{tri} with those of a biphone λ^{bi} and a monophone λ^{mono}:

$$\hat{\lambda}^{tri} = \alpha_3 \lambda^{tri} + \alpha_2 \lambda^{bi} + \alpha_1 \lambda^{mono}$$

Smoothing: Interpolation

- Basic idea Combine less-specific models with more specific models
- Interpolate the parameters of a triphone λ^{tri} with those of a biphone λ^{bi} and a monophone λ^{mono}:

$$\hat{\lambda}^{tri} = \alpha_3 \lambda^{tri} + \alpha_2 \lambda^{bi} + \alpha_1 \lambda^{mono}$$

- Estimate the interpolation parameters α using deleted interpolation

Smoothing: Interpolation

- Basic idea Combine less-specific models with more specific models
- Interpolate the parameters of a triphone λ^{tri} with those of a biphone λ^{bi} and a monophone λ^{mono}:

$$\hat{\lambda}^{tri} = \alpha_3 \lambda^{tri} + \alpha_2 \lambda^{bi} + \alpha_1 \lambda^{mono}$$

- Estimate the interpolation parameters α using deleted interpolation
- This enables more triphone models to be estimated, but adds robustness by sharing training data from other contexts (through the biphone and monophone models)

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts
  - enables training data to be shared between the models

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts
  - enables training data to be shared between the models
  - enables models to share parameters

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts
  - enables training data to be shared between the models
  - enables models to share parameters
- Sharing can take place at different levels

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts
  - enables training data to be shared between the models
  - enables models to share parameters
- Sharing can take place at different levels

1. Sharing Gaussians: all distributions share the same set of Gaussians but have different mixture weights (tied mixtures)

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts
  - enables training data to be shared between the models
  - enables models to share parameters
- Sharing can take place at different levels

1. Sharing Gaussians: all distributions share the same set of Gaussians but have different mixture weights (tied mixtures)
2. Sharing states: allow different models to share the same states (state clustering)

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts
  - enables training data to be shared between the models
  - enables models to share parameters
- Sharing can take place at different levels

1. Sharing Gaussians: all distributions share the same set of Gaussians but have different mixture weights (tied mixtures)
2. Sharing states: allow different models to share the same states (state clustering)
3. Sharing models: merge those context-dependent models that are the most similar (generalised triphones)

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts
  - enables training data to be shared between the models
  - enables models to share parameters
- Sharing can take place at different levels

1. Sharing Gaussians: all distributions share the same set of Gaussians but have different mixture weights (tied mixtures)
2. Sharing states: allow different models to share the same states (state clustering)
3. Sharing models: merge those context-dependent models that are the most similar (generalised triphones)

Sharing Gaussians: Tied mixture models

- Basic idea all states share the same Gaussians

Sharing Gaussians: Tied mixture models

- Basic idea all states share the same Gaussians
- Have a pool of G Gaussians shared between all HMM states – each state has a G-component GMM output distribution

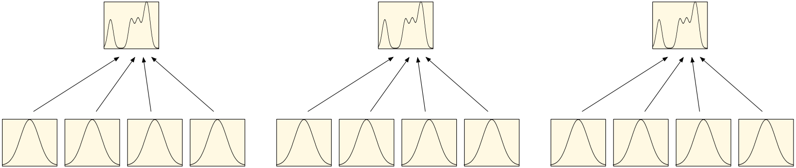
Sharing Gaussians: Tied mixture models

- Basic idea all states share the same Gaussians
- Have a pool of G Gaussians shared between all HMM states – each state has a G-component GMM output distribution
- Therefore the mean vectors and covariance matrices are shared between states

Sharing Gaussians: Tied mixture models

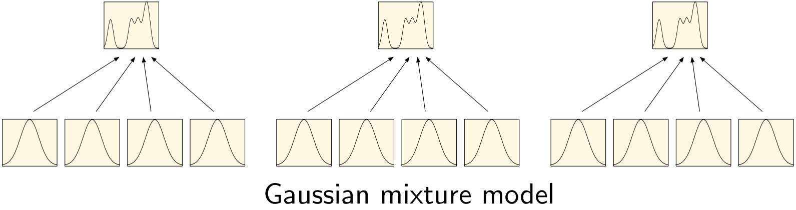
- Basic idea all states share the same Gaussians
- Have a pool of G Gaussians shared between all HMM states – each state has a G-component GMM output distribution
- Therefore the mean vectors and covariance matrices are shared between states
- The mixture component weights are specific to each state

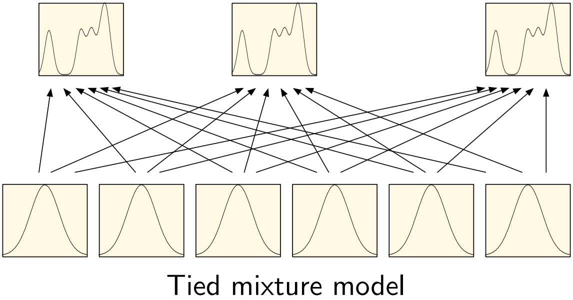
Tied Mixture Model



Gaussian mixture model

Tied Mixture Model





Sharing Gaussians: Tied mixture models

- Basic idea all states share the same Gaussians
- Have a pool of G Gaussians shared between all HMM states – each state has a G-component GMM output distribution
- Therefore the mean vectors and covariance matrices are shared between states
- The mixture component weights are specific to each state
- In context-dependent models, the mixture component weights may be smoothed using interpolation

Sharing Gaussians: Tied mixture models

- Basic idea all states share the same Gaussians
- Have a pool of G Gaussians shared between all HMM states – each state has a G-component GMM output distribution
- Therefore the mean vectors and covariance matrices are shared between states
- The mixture component weights are specific to each state
- In context-dependent models, the mixture component weights may be smoothed using interpolation
- Tied mixture systems work well due to the large amount of parameter sharing and smoothing of the weights

Sharing Gaussians: Tied mixture models

- Basic idea all states share the same Gaussians
- Have a pool of G Gaussians shared between all HMM states – each state has a G-component GMM output distribution
- Therefore the mean vectors and covariance matrices are shared between states
- The mixture component weights are specific to each state
- In context-dependent models, the mixture component weights may be smoothed using interpolation
- Tied mixture systems work well due to the large amount of parameter sharing and smoothing of the weights
- But we can do better (state clustering)!

Sharing Gaussians: Tied mixture models

- Basic idea all states share the same Gaussians
- Have a pool of G Gaussians shared between all HMM states – each state has a G-component GMM output distribution
- Therefore the mean vectors and covariance matrices are shared between states
- The mixture component weights are specific to each state
- In context-dependent models, the mixture component weights may be smoothed using interpolation
- Tied mixture systems work well due to the large amount of parameter sharing and smoothing of the weights
- But we can do better (state clustering)!
- Tied mixtures are still used when time and memory efficiency is important (eg embedded systems)

Sharing Models: Generalized triphones

- Basic idea Merge similar context-dependent models

Sharing Models: Generalized triphones

- Basic idea Merge similar context-dependent models
- Instead of using phones as left and right contexts, define context classes that cover multiple phone types

Sharing Models: Generalized triphones

- Basic idea Merge similar context-dependent models
- Instead of using phones as left and right contexts, define context classes that cover multiple phone types
- Top down merging: Use broad phonetic classes (eg stop, fricative) as context classes

Sharing Models: Generalized triphones

- Basic idea Merge similar context-dependent models
- Instead of using phones as left and right contexts, define context classes that cover multiple phone types
- Top down merging: Use broad phonetic classes (eg stop, fricative) as context classes
- Bottom-up merging: Compare allophone models with different triphone contexts and merge those that are similar

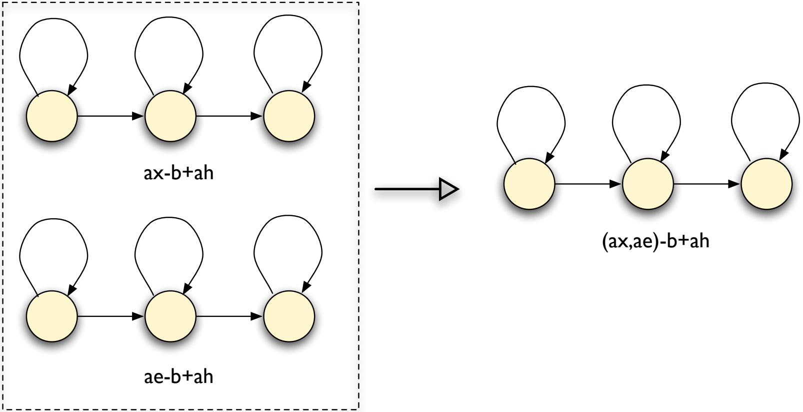
Sharing Models: Generalized triphones

- Basic idea Merge similar context-dependent models
- Instead of using phones as left and right contexts, define context classes that cover multiple phone types
- Top down merging: Use broad phonetic classes (eg stop, fricative) as context classes
- Bottom-up merging: Compare allophone models with different triphone contexts and merge those that are similar
- Merged models will be estimated from more data than individual models: more accurate models, fewer models in total

Sharing Models: Generalized triphones

- Basic idea Merge similar context-dependent models
- Instead of using phones as left and right contexts, define context classes that cover multiple phone types
- Top down merging: Use broad phonetic classes (eg stop, fricative) as context classes
- Bottom-up merging: Compare allophone models with different triphone contexts and merge those that are similar
- Merged models will be estimated from more data than individual models: more accurate models, fewer models in total
- The resultant merged models are referred to as generalized triphones

Example: Generalized Triphones



Modelling speech with HMMs – Context-dependent phone models

Steve Renals

Automatic Speech Recognition
ASR Lectures 6&8
28 January, 4 February 2016

Overview

Phone models

- Modelling phones with HMMs
- The need to model phonetic context
- Triphone models
- Smoothing – interpolation and backing-off
- Parameter sharing – tied mixtures, generalised triphones, state clustering
- Choosing which states to share – phonetic decision trees

Parameter Sharing

- Basic idea Explicitly share models or parameters between different contexts
  - enables training data to be shared between the models
  - enables models to share parameters
- Sharing can take place at different levels

1. Sharing Gaussians: all distributions share the same set of Gaussians but have different mixture weights (tied mixtures)
2. Sharing states: allow different models to share the same states (state clustering)
3. Sharing models: merge those context-dependent models that are the most similar (generalised triphones)

Sharing States: State clustering

- Basic idea States which are responsible for acoustically similar data are shared

Sharing States: State clustering

- Basic idea States which are responsible for acoustically similar data are shared
- By clustering similar states, the training data associated with individual states may be pooled together – results in better parameter estimates for the state

Sharing States: State clustering

- Basic idea States which are responsible for acoustically similar data are shared
- By clustering similar states, the training data associated with individual states may be pooled together – results in better parameter estimates for the state
  1. Create a set of context dependent models for a parent phone

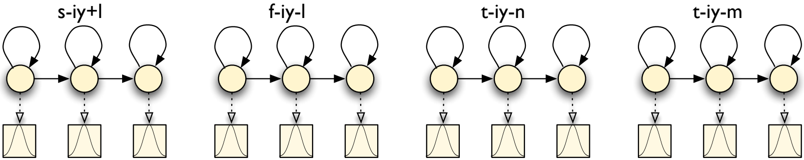
Sharing States: State clustering

- Basic idea States which are responsible for acoustically similar data are shared
- By clustering similar states, the training data associated with individual states may be pooled together – results in better parameter estimates for the state
    1. Create a set of context dependent models for a parent phone
    2. Cluster and tie similar states, ensuring that each resultant clustered state is responsible for "enough" training data (ie setting a minimum state occupation count)

Sharing States: State clustering

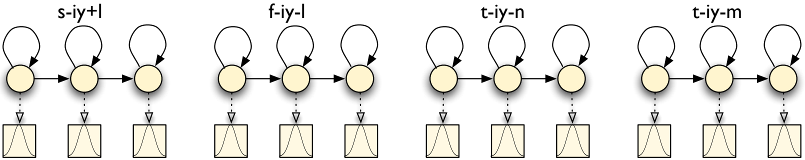
- Basic idea States which are responsible for acoustically similar data are shared
- By clustering similar states, the training data associated with individual states may be pooled together – results in better parameter estimates for the state
  1. Create a set of context dependent models for a parent phone
  2. Cluster and tie similar states, ensuring that each resultant clustered state is responsible for "enough" training data (ie setting a minimum state occupation count)
- More flexible than clustering whole models: left and right contexts may be clustered separately

Generalized triphones

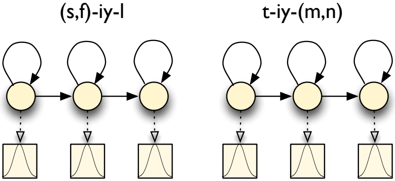


Simple triphones (no sharing)

Generalized triphones

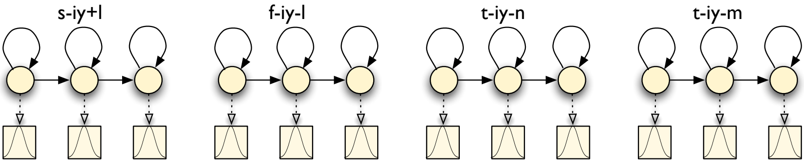


Simple triphones (no sharing)



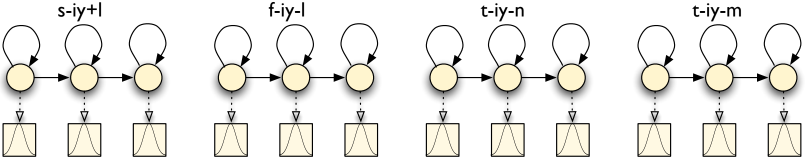
Generalized triphones (model sharing)

State Clustering

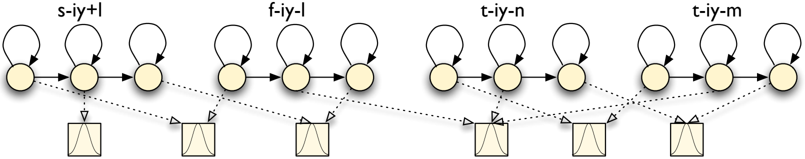


Simple triphones (no sharing)

State Clustering

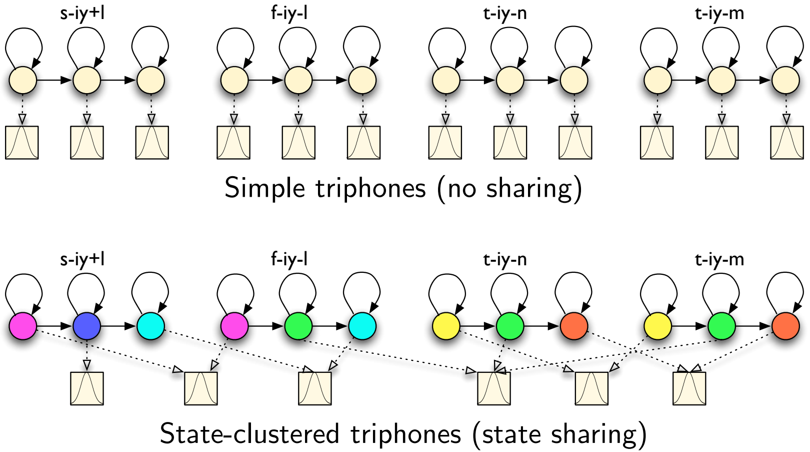


Simple triphones (no sharing)



State-clustered triphones (state sharing)

State Clustering



Good contexts to share

- Which states should be clustered together?

Good contexts to share

- Which states should be clustered together?
- Bottom-up clustering, for triphones of the same parent phone

Good contexts to share

- Which states should be clustered together?
- Bottom-up clustering, for triphones of the same parent phone
  1. Create raw triphone models for each observed triphone context

Good contexts to share

- Which states should be clustered together?
- Bottom-up clustering, for triphones of the same parent phone
  1. Create raw triphone models for each observed triphone context
  2. Cluster states as before

Good contexts to share

- Which states should be clustered together?
- Bottom-up clustering, for triphones of the same parent phone
  1. Create raw triphone models for each observed triphone context
  2. Cluster states as before
- Top-down clustering: start with a parent context independent model then successively split models to create context dependent models

Good contexts to share

- Which states should be clustered together?
- Bottom-up clustering, for triphones of the same parent phone
  1. Create raw triphone models for each observed triphone context
  2. Cluster states as before
- Top-down clustering: start with a parent context independent model then successively split models to create context dependent models
- Phonetic decision trees

Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node

Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node
- At the root of the tree, all states are shared

Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node
- At the root of the tree, all states are shared
- Questions split the pool of states, the resultant state clusters are given by the leaves of the tree

Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node
- At the root of the tree, all states are shared
- Questions split the pool of states, the resultant state clusters are given by the leaves of the tree
- Example questions:

Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node
- At the root of the tree, all states are shared
- Questions split the pool of states, the resultant state clusters are given by the leaves of the tree
- Example questions:
  - Is the left context a nasal?

Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node
- At the root of the tree, all states are shared
- Questions split the pool of states, the resultant state clusters are given by the leaves of the tree
- Example questions:
  - Is the left context a nasal?
  - Is the right context a central stop?

Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node
- At the root of the tree, all states are shared
- Questions split the pool of states, the resultant state clusters are given by the leaves of the tree
- Example questions:
  - Is the left context a nasal?
  - Is the right context a central stop?
- The questions at each node are chosen from a large set of predefined questions

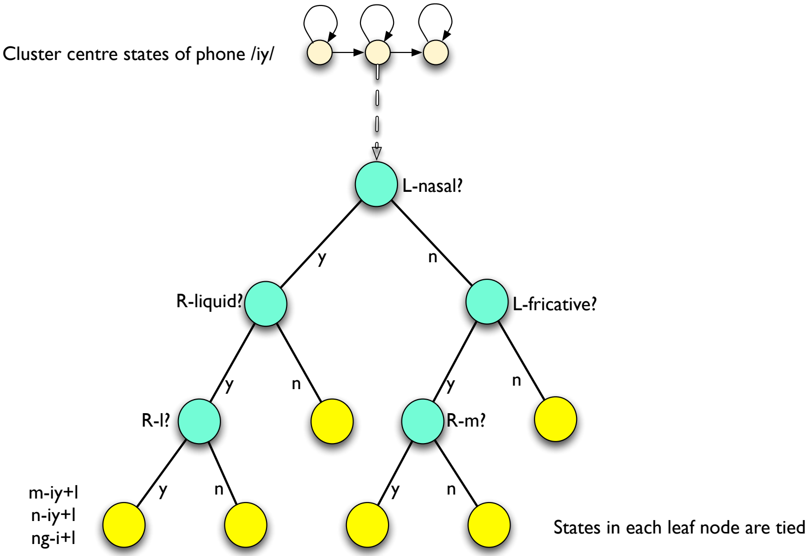
Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node
- At the root of the tree, all states are shared
- Questions split the pool of states, the resultant state clusters are given by the leaves of the tree
- Example questions:
  - Is the left context a nasal?
  - Is the right context a central stop?
- The questions at each node are chosen from a large set of predefined questions
- Choose the question which maximizes the likelihood of the data given the state clusters

Phonetic Decision Trees

- Basic idea Build a decision tree for each state of each parent phone, with yes/no questions at each node
- At the root of the tree, all states are shared
- Questions split the pool of states, the resultant state clusters are given by the leaves of the tree
- Example questions:
  - Is the left context a nasal?
  - Is the right context a central stop?
- The questions at each node are chosen from a large set of predefined questions
- Choose the question which maximizes the likelihood of the data given the state clusters
- Stop splitting if either: (a) the likelihood does not increase by more than a predefined threshold; or (b) the amount of data associated with a split node would below a threshold

Phonetic Decision Tree



Phonetic questions

- Ask questions of the form: does phone at offset s have feature f?
- Offsets are $+/-1$ for triphone context
- Example general questions:
  - Stop: `b d g p t k`
  - Nasal: `m n ng`
  - Fricative: `ch dh f jh s sh th v z zh`
  - Liquid: `l r w y`
  - Vowel: `aa ae ah ao aw ax axr ay eh er ...`
- Example consonant questions: Un/voiced, front/central/back, fortis (`ch f k p s sh t th`), lenis (`b d dh g jh v z zh`), voiced stop, ...
- Example vowel questions: front, central, back, long, short, diphthong, rounded, ...

Most useful phonetic questions

- All states of all models:
  `+Vowel -Vowel +Unrounded -UnFortisLenis +UnFortisLenis +r`
- Entry state of all models:
  `-UnFortisLenis -Vowel -Nasal -CentralFront -Unrounded -Fortis`
- Exit state of all consonants:
  `+Vowel +Unrounded +High +ee +Rounded +Syllabic`

(for Wall St Journal read speech – Young, Odell and Woodland 1994)

Likelihood of a state cluster (1)

- Basic idea Compute the log likelihood of the data associated with a pool of states

Likelihood of a state cluster (1)

- Basic idea Compute the log likelihood of the data associated with a pool of states
- All states pooled in a single cluster at the root

Likelihood of a state cluster (1)

- Basic idea Compute the log likelihood of the data associated with a pool of states
- All states pooled in a single cluster at the root
- All states have Gaussian output pdf

Likelihood of a state cluster (1)

- Basic idea Compute the log likelihood of the data associated with a pool of states
- All states pooled in a single cluster at the root
- All states have Gaussian output pdf
- Let $\mathbf{S} = \{s_1, s_2, \ldots, s_K\}$ be a pool of K states forming a cluster, sharing a common mean $\boldsymbol{\mu}_S$ and covariance $\boldsymbol{\Sigma}_S$

Likelihood of a state cluster (1)

- Basic idea Compute the log likelihood of the data associated with a pool of states
- All states pooled in a single cluster at the root
- All states have Gaussian output pdf
- Let $\mathbf{S} = \{s_1, s_2, \ldots, s_K\}$ be a pool of K states forming a cluster, sharing a common mean $\boldsymbol{\mu}_S$ and covariance $\boldsymbol{\Sigma}_S$
- Let $\mathbf{X}$ be the set of training data

Likelihood of a state cluster (1)

- Basic idea Compute the log likelihood of the data associated with a pool of states
- All states pooled in a single cluster at the root
- All states have Gaussian output pdf
- Let $\mathbf{S} = \{s_1, s_2, \ldots, s_K\}$ be a pool of K states forming a cluster, sharing a common mean $\boldsymbol{\mu}_S$ and covariance $\boldsymbol{\Sigma}_S$
- Let $\mathbf{X}$ be the set of training data
- Let $\gamma_s(\mathbf{x})$ be the probability that $\mathbf{x} \in \mathbf{X}$ was generated by state s (i.e. state occupation probability)

Likelihood of a state cluster (1)

- Basic idea Compute the log likelihood of the data associated with a pool of states
- All states pooled in a single cluster at the root
- All states have Gaussian output pdf
- Let $\mathbf{S} = \{s_1, s_2, \ldots, s_K\}$ be a pool of K states forming a cluster, sharing a common mean $\boldsymbol{\mu}_S$ and covariance $\boldsymbol{\Sigma}_S$
- Let $\mathbf{X}$ be the set of training data
- Let $\gamma_s(\mathbf{x})$ be the probability that $\mathbf{x} \in \mathbf{X}$ was generated by state s (i.e. state occupation probability)
- The log likelihood of the data associated with cluster $\mathbf{S}$ is:

$$L(\mathbf{S}) = \sum_{s \in \mathbf{S}} \sum_{\mathbf{x} \in \mathbf{X}} \log P(\mathbf{x} | \boldsymbol{\mu}_S, \boldsymbol{\Sigma}_S) \gamma_s(\mathbf{x})$$

Likelihood of a state cluster (2)

- Don't need to iterate through all data for each state
- If the output pdfs are Gaussian it can be shown that

$$L(\mathbf{S}) = -\frac{1}{2}\left(\log(2\pi)^d|\boldsymbol{\Sigma}_S)|) + d\right)\sum_{s\in\mathbf{S}}\sum_{\mathbf{x}\in\mathbf{X}}\gamma_s(\mathbf{x})$$

  where d is the dimension of the data
- Thus $L(\mathbf{S})$ depends on only
  - the pooled state variance $\boldsymbol{\Sigma}_S$ - can be computed from the means and variances of the individual states in the pool
  - and the state occupation probabilities already computed when forward-backward was carried out

State splitting (1)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

State splitting (1)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question
- Split $\mathbf{S}$ into two partitions $\mathbf{S}_y$ and $\mathbf{S}_n$ using a question about the phonetic context

State splitting (1)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question
- Split $\mathbf{S}$ into two partitions $\mathbf{S}_y$ and $\mathbf{S}_n$ using a question about the phonetic context
- Each partition is now clustered together to form a single Gaussian output distribution with mean $\boldsymbol{\mu}_{S_y}$ and covariance $\boldsymbol{\Sigma}_{S_y}$) (for partition S_y)

State splitting (1)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question
- Split $\mathbf{S}$ into two partitions $\mathbf{S}_y$ and $\mathbf{S}_n$ using a question about the phonetic context
- Each partition is now clustered together to form a single Gaussian output distribution with mean $\boldsymbol{\mu}_{S_y}$ and covariance $\boldsymbol{\Sigma}_{S_y}$) (for partition S_y)
- The likelihood of the data after partition is given by $L(\mathbf{S}_y) + L(\mathbf{S}_n)$

State splitting (1)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question
- Split $\mathbf{S}$ into two partitions $\mathbf{S}_y$ and $\mathbf{S}_n$ using a question about the phonetic context
- Each partition is now clustered together to form a single Gaussian output distribution with mean $\boldsymbol{\mu}_{S_y}$ and covariance $\boldsymbol{\Sigma}_{S_y}$) (for partition S_y)
- The likelihood of the data after partition is given by $L(\mathbf{S}_y) + L(\mathbf{S}_n)$
- The total likelihood of the partitioned data will increase by

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

State splitting (2)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

State splitting (2)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

- Cycle through all possible questions, compute Δ for each and choose the question for which Δ is biggest

State splitting (2)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

- Cycle through all possible questions, compute Δ for each and choose the question for which Δ is biggest
- Continue by splitting each of the new clusters $\mathbf{S}_y$ and $\mathbf{S}_n$

State splitting (2)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

- Cycle through all possible questions, compute Δ for each and choose the question for which Δ is biggest
- Continue by splitting each of the new clusters $\mathbf{S}_y$ and $\mathbf{S}_n$
- Terminate when

State splitting (2)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

- Cycle through all possible questions, compute Δ for each and choose the question for which Δ is biggest
- Continue by splitting each of the new clusters $\mathbf{S}_y$ and $\mathbf{S}_n$
- Terminate when
  1. Maximum Δ falls below a threshold

State splitting (2)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

- Cycle through all possible questions, compute Δ for each and choose the question for which Δ is biggest
- Continue by splitting each of the new clusters $\mathbf{S}_y$ and $\mathbf{S}_n$
- Terminate when
  1. Maximum Δ falls below a threshold
  2. The amount of data associated with a split node falls below a threshold

State splitting (2)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

- Cycle through all possible questions, compute Δ for each and choose the question for which Δ is biggest
- Continue by splitting each of the new clusters $\mathbf{S}_y$ and $\mathbf{S}_n$
- Terminate when
  1. Maximum Δ falls below a threshold
  2. The amount of data associated with a split node falls below a threshold
- For a Gaussian output distribution: State likelihood estimates can be estimated using just the *state occupation counts* (obtained at alignment) and the parameters of the Gaussian – no need to use the acoustic data

State splitting (2)

- Basic idea Use the likelihood of the parent state and of the split states to choose the splitting question

$$\Delta = L(\mathbf{S}_y) + L(\mathbf{S}_n) - L(\mathbf{S})$$

- Cycle through all possible questions, compute Δ for each and choose the question for which Δ is biggest
- Continue by splitting each of the new clusters $\mathbf{S}_y$ and $\mathbf{S}_n$
- Terminate when
  1. Maximum Δ falls below a threshold
  2. The amount of data associated with a split node falls below a threshold
- For a Gaussian output distribution: State likelihood estimates can be estimated using just the *state occupation counts* (obtained at alignment) and the parameters of the Gaussian – no need to use the acoustic data
- State occupation count: sum of state occupation probabilities for a state over time

"Mixing up"

- Basic idea Transforming an HMM-based system based on Gaussian distributions to one based on mixtures of Gaussians

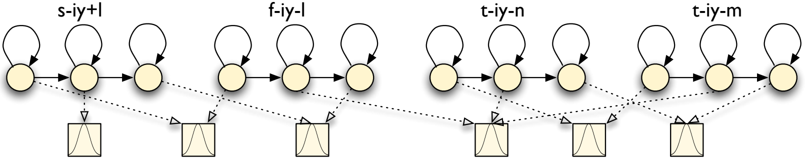
“Mixing up”

- Basic idea Transforming an HMM-based system based on Gaussian distributions to one based on mixtures of Gaussians
- The above methods for state clustering assume that the state outputs are Gaussians – this makes the computations much simpler
- BUT: Gaussian mixtures offer much better acoustic models than Gaussians

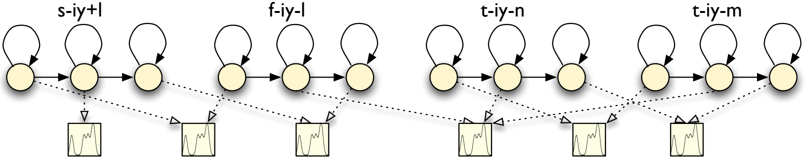
"Mixing up"

- Basic idea Transforming an HMM-based system based on Gaussian distributions to one based on mixtures of Gaussians
- The above methods for state clustering assume that the state outputs are Gaussians – this makes the computations much simpler
- BUT: Gaussian mixtures offer much better acoustic models than Gaussians
- Solution:
  - Perform state clustering using Gaussian distributions
  - Split the Gaussian distributions in the clustered states, by cloning and perturbing the means by a small fraction of the standard deviation, and retrain.
  - Repeat by splitting the dominant (highest state occupation count) mixture components in each state

"Mixing up"



State-clustered triphones (Gaussians)



State-clustered triphones (GMMs)

Summary: Context-dependent acoustic modelling

- Share parameters through state clustering
- Cluster states using phonetic decision trees for each state of parent phone
- Use Gaussian distributions when state clustering
- Then split Gaussians and retrain to obtain a GMM state clustered system

References: context-dependent acoustic modelling

- *c*1980: First proposed by Bahl et al (IBM)
- Schwartz et al (1985): first paper using triphone models
- Lee (1990): generalized triphones
- Bellegarda (1990), Huang (1992): tied mixture modelling
- Bahl et al (1991): phonetic decision trees first proposed
- Young and Woodland (1994): state clustering
- Young et al (1994): decision tree-based state clustering