Subspace GMM – the math

Dan Povey

Most basic model

$$
\begin{aligned}
p(\mathbf{x}|j) &= \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i) \\
\boldsymbol{\mu}_{ji} &= \mathbf{M}_i\mathbf{v}_j \\
w_{ji} &= \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}
\end{aligned}
$$

j is speech class (phone in context)

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Gaussian Mixture Model

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Same number of Gaussians in each state

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Full covariances shared between states (but different for each Gaussian)

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Means and weights controlled by other parameters

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

State-specific vectors $\mathbf{v}$ determine means and weights

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Globally shared parameters $\mathbf{M}_i$ and $\mathbf{w}_i$ determine the mapping

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{ji}, \boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i \mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T \mathbf{v}_j)}{\sum_{i'=1}^{I} \exp(\mathbf{w}_{i'}^T \mathbf{v}_j)}$$

Mapping of means is linear

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

Mapping of weights is log-linear (with renormalization)

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}{}^T\mathbf{v}_j)}$$

There is a correspondence between Gaussians across states

$$p(\mathbf{x}|j) = \sum_{i=1}^{I} w_{ji}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{ji},\boldsymbol{\Sigma}_i)$$

$$\boldsymbol{\mu}_{ji} = \mathbf{M}_i\mathbf{v}_j$$

$$w_{ji} = \frac{\exp(\mathbf{w}_i^T\mathbf{v}_j)}{\sum_{i'=1}^{I}\exp(\mathbf{w}_{i'}^T\mathbf{v}_j)}$$

Universal Background Model (UBM)

- The UBM is a single Gaussian Mixture Model that has been trained on all speech classes.
- It is used to initialize the SGMM training.
- It is used to prune the Gaussian indexes i during training and decoding.

Likelihood evaluation

- Using the UBM, prune to a subset of the indices i, e.g. the top 10 of them.
- With appropriate pre-computation per frame, each Gaussian i in each state j can have its likelihood evaluated with a single dot product in the dimension of $\mathbf{v}_j$ (typically 40 or 50).
- This can be even faster than a typical mixture-of-Gaussians system.

Prerequisites for model building

- A previously trained system (conventional or SGMM based), needed for initial state alignment.
- A phonetic context tree (use normal methods to get this).
- A trained UBM.

Model initialization

- Typical initialization:
- Set dimension of vectors $\mathbf{v}_j$ to feature-dim + 1 (e.g. 39+1 = 40).
- Set $\mathbf{v}_j$ to a unit vector [1 0 0 0 …]
- Set $\mathbf{M}_i$ to [$\boldsymbol{\mu}_i$ $\mathbf{I}$], where $\boldsymbol{\mu}_i$ is i'th mean in UBM
- Set $\mathbf{w}_i$ to zero vector (so all weights are equal)
- Effect is that the initial GMM in each state j is the same as the UBM (with equal weights).
- This is not the only way to initialize.

Model training

- Training is based on Expectation-Maximization, the same as traditional GMM training
- Each iteration, we pick a parameter type ($\mathbf{M}_i$ or $\mathbf{w}_i$ or $\mathbf{v}_j$), accumulate statistics, update it.
- Each update step is guaranteed to increase likelihood on the training data.
- Can actually combine the updates on a single iteration, within certain constraints.

Model training: $\mathbf{v}_j$

- Auxiliary function is quadratic in $\mathbf{v}_j$:

$$\mathcal{Q}(\mathbf{v}_j) = \mathbf{g}_j \cdot \mathbf{v}_j - \frac{1}{2}\mathbf{v}_j^T \mathbf{H}_j \mathbf{v}_j$$

Linear term

$$\mathcal{Q}(\mathbf{v}_j) = \mathbf{g}_j \cdot \mathbf{v}_j - \frac{1}{2}\mathbf{v}_j^T \mathbf{H}_j \mathbf{v}_j$$

Quadratic term

$$\mathcal{Q}(\mathbf{v}_j) \quad = \quad \mathbf{g}_j \cdot \mathbf{v}_j - \frac{1}{2}\mathbf{v}_j^T \mathbf{H}_j \mathbf{v}_j$$

Obtaining the auxiliary function

- $\mathbf{g}_j$ and $\mathbf{H}_j$ computed from statistics accumulated from the data on each iteration of training (don't accumulate directly: efficiency).
- The part of the auxiliary function that arises from the effect on the means is naturally quadratic
- The part that arises from the effect on the weights is not (we use a quadratic approximation).

Update equation for $\mathbf{v}_j$

$$\mathbf{v}_j = \mathbf{H}_j^{-1}\mathbf{g}_j$$

- Mathematically speaking, $\mathbf{H}_j$ always invertible
- Practically speaking, not always invertible (can have very tiny eigenvalues)
- Will discuss this problem later

Model training: M

- Auxiliary function is also quadratic:

$$\mathcal{Q}(\mathbf{M}_i) = \mathrm{tr}\left(\mathbf{M}_i^T \boldsymbol{\Sigma}_i^{-1} \mathbf{Y}_i\right) - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Sigma}_i^{-1} \mathbf{M}_i \mathbf{Q}_i \mathbf{M}_i^T\right)$$

- $\mathbf{Y}_i$ obtained directly from the data; $\mathbf{Q}_i$ derived from outer products of $\mathbf{v}_j$ weighted by data counts.
- Update is: $\mathbf{M}_i = \mathbf{Y}_i \mathbf{Q}_i^{-1}$
- Again we have a problem when $\mathbf{Q}_i$ is not invertible (but it will normally be invertible).

Model training: w

- Training the "weight projection vectors" $\mathbf{w}_i$ is a little less easy: there is no natural auxiliary function
- It is possible to obtain an approximate quadratic auxiliary function that leads to an update that converges quite fast.
- The sufficient statistics for updating $\mathbf{w}_i$ are the data counts for each state index j and Gaussian index i

Non-invertible matrices

- Sometimes the matrices that represent the quadratic terms in the auxiliary functions are singular.
- This situation corresponds to “don’t-care” directions in the parameter space (the linear term will also be zero in these directions).
- If we attempt to invert singular matrices we will tend to get unpredictable results.
- This is more of a problem with small datasets.

Solutions to non-invertibility

- Can introduce priors over the parameters.
  - Can make prior based on an ad-hoc formula with a τ value (like HTK-based MAP estimation)
  - OR estimate them from (estimated) parameters
- Can use a solution based on a "least squares" approach: get parameter with smallest squared value that gives maximum of auxiliary function
  - This is not possible to do exactly given the form of the statistics we accumulate (because impossible to distinguish very small from zero eigenvalues) but can approximate it in an acceptable way.
- Can just refuse to update those parameters.
- Not clear what the best solution is.

“Extra” stuff

- On top of the basic model, we can do various things. Will summarize the more important of these in the next slides:
  - Sub-states
  - Speaker subspace
  - Constrained MLLR

Sub-states

- Introduce within each state, a weighted mixture of what we previously had:

$$
\begin{aligned}
p(\mathbf{x}|j) &= \sum_{m=1}^{M_j} c_{jm} \sum_{i=1}^{I} w_{jmi} \mathcal{N}(\mathbf{x}; \mu_{jmi}, \mathbf{\Sigma}_i) \\
\mu_{jmi} &= \mathbf{M}_i \mathbf{v}_{jm} \\
w_{jmi} &= \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jm}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jm}}.
\end{aligned}
$$

- New parameter type: mixture weights c_{jm}
- Sub-states help more than increasing dim of $\mathbf{v}$

Speaker Subspace

- Use a similar approach to cover speaker variation:

$$
\begin{aligned}
p(\mathbf{x}|j,s) &= \sum_{m=1}^{M_j} c_{jm} \sum_{i=1}^{I} w_{jmi} \mathcal{N}(\mathbf{x}; \mu_{jmi}^{(s)}, \mathbf{\Sigma}_i) \\
\mu_{jmi}^{(s)} &= \mathbf{M}_i \mathbf{v}_{jm} + \mathbf{N}_i \mathbf{v}^{(s)} \\
w_{jmi} &= \frac{\exp \mathbf{w}_i^T \mathbf{v}_{jm}}{\sum_{i'=1}^{I} \exp \mathbf{w}_{i'}^T \mathbf{v}_{jm}},
\end{aligned}
$$

- Do not make the mixture weights depend on speaker (makes decoding too slow)
- New parameters $\mathbf{N}_i$ (globally shared), $\mathbf{v}^{(s)}$ (speaker specific)

Constrained MLLR

- A linear feature transformation

$$\mathbf{x} \rightarrow \mathbf{A}^{(s)}\mathbf{x} + \mathbf{b}^{(s)}$$

- Estimated for each speaker s
- During decoding this is independent of the model but the estimation formulas need to be specially formulated (full covariance)

Typical setup

- About 400-1000 Gaussians in the UBM
- Phonetic subspace and speaker subspace both have dimension 40-60
- Mix up to about half the number of sub-states that the baseline system had Gaussians
- Speaker-specific adaptation parameters $\mathbf{v}^{(s)}$, $\mathbf{A}^{(s)}$, $\mathbf{b}^{(s)}$ all to be estimated on speech only (not silence)
- Language model weight smaller than normal system, e.g. 8-10 vs. 13-14 with normal system
- Obtain UBM by clustering (diagonal) Gaussians from a normal system, doing full-covariance re-estimation

Issues with adaptation

- Speaker-vector adaptation not data-hungry
- Better done per segment rather than given reasonable segment lengths (not just 1 or 2 seconds)
- We developed the parameter-subspace CMLLR to enable both on per-segment basis
- Hard to get adaptation working in our setup:
  - Segments were very short (most utterances 1 or 2 seconds)
  - Adapting on silence frames bad! (needed to exclude them)
  - Not clear whether the corpus, model, or feature extraction
  - Full covariances for silence were getting floored eigenvalues (floored to 1/1000 of the largest). Strange silence features?

Software design

- This technique is more complicated than a normal mixture of Gaussians system – needs better testing.
- We separately unit-tested all easily testable code (e.g. linear algebra code)
- Printed out copious diagnostics; measured all auxiliary function changes and compared with likelihoods
- Used as much as possible blocks that can be swapped in and out with other blocks, for easier testing
- E.g. we wrote two separate versions of the update code and used each to help debug the other.
- Debugged the calling code by writing a simple GMM based acoustic model with the same C++ interface as the SGMM.