Subspace based/Universal Background Model (UBM) based speech modeling

This paper is available at

`http://dpovey.googlepages.com/jhu_lecture2.pdf`

Daniel Povey

June, 2009

Overview

- Introduce the concept of speech recognition using a shared GMM structure, and mention prior work with MAP adaptation of the GMM

- Introduce the simplest version of the subspace based speech model

- Discuss the optimization of the various parameters in the model

- Describe the extensions to the basic model that were used in previous experiments

- Show recognition results with the previously used model

- Describe extensions to be used in the workshop

Conventional speech system

- A conventional speech system based on mixture of Gaussians models contains several thousand separate Gaussian Mixture Models (GMMs), typically with diagonal covariances.

- Each GMM (i.e. each HMM state) corresponds to one of three positions (begin,middle,end) within one specific phone (phones correspond roughly to letters in the alphabet).

- There are several thousand of these (not 3×40 or so) because of context clustering (phones sound different depending on the phonetic context).

- Each GMM has a potentially different number of Gaussians in it (typically 10-20 or so) and they are separately built*

```
struct GMM_HMM {
  int J; // Number of states.
  int *Mj; // Number of mixtures in each state.
  float **c; // Mixture weights; [j][m]
  float ***means;  //[j][m][d]
  float ***vars;   //[j][m][d]
};
```

*This is not strictly true in BBN-style systems based on "senones".

Maximum A Posteriori (MAP)-based UBM system*.

- First build a generic Gaussian Mixture Model (GMM) on all of speech lumped into one class.
- This can be quite big (e.g. 1000 mixtures).
- Then MAP adapt this to each acoustic state (adapting the means, variances and weights). Recall: MAP update uses τ to control backoff to prior.
- It is possible to use the tree structure of the phonetic context decision tree to improve the smoothing: repeatedly back off to parent nodes.
- Training can proceed for several iterations (MAP backs off to parent nodes, not prior model).
- It was helpful to use separate Semi-tied Covariance transforms for each of 1000 "original" mixtures. (Hard to get much improvement from multiple STC transforms normally).
- Original (unadapted) GMM can be used for pruning.

* "Universal Background Model based Speech Recognition", by D. Povey, S. Chu & B. Varadarajan, ICASSP 2008

Maximum A Posteriori (MAP)-based UBM system: code and results

```
struct UBM_MAP_HMM{ // just the parameters.
   int J; // #states.
   int I; // #mixtures in GMM.  Same for all states!
   float **c; // [j][i].  Mixture weights.  Index [J] for unadapted one.
   float ***means; // [j][i][d]; index [J] for unadapted one.
   float ***vars; // [j][i][d]; index [J] for unadapted one.
   float ***stc; // [j][d][d]; STC transforms.
   float *stc_logdet; // [j]; Determinants of STC transforms.
};
```

- Improvement, on a Broadcast News system with 1300 hours of test data, was 16.2% WER to 14.8% WER (fully adapted: VTLN, constrained MLLR, MLLR).

- This may overstate the improvement because although the baseline was extremely large (1 million Gaussians), experiments on a VTLN-only setup showed that 1% absolute improvement was possible by going to 2 million.

- “Real” improvement may have been as little as 0.5%, exact value doesn’t matter because discriminative training would not have worked on this system (too many parameters).

Subspace UBM system: most basic version

- Different structure: adapt only means and weights, in a subspace.

$$\mu_{ji} = \mathbf{M}_i \mathbf{v}_j^+$$

Weights: $$w_{ji} = \frac{\exp(\mathbf{w}_i \mathbf{v}_j^+)}{\sum_i \exp(\mathbf{w}_i \mathbf{v}_j^+)}.$$

- Notation $\mathbf{v}_j^+$ means appending a 1 to vector $\mathbf{v}_j$.

- Dimension of $\mathbf{v}_j$ is "subspace dimension" S. (This is the dimension of a subspace of the mean parameter space which is of size DI).

- In place of STC transforms, use full covariances for each index i in the original GMM, shared across states.

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

- Parameters are $\mathbf{v}_j, \mathbf{M}_i, \mathbf{w}_i, \mathbf{\Sigma}_i$.

Subspace UBM system: most basic version (declarations)

```
struct BACKGROUND_GMM { // Background GMM (UBM) used for pruning only.
  int D; int I; // feature-dim; #mixtures.
  float **diag_vars; // Diagonal covariances.  First stage of pruning.
  float ***vars; // Inverse full covariances.  Second stage of pruning.
  float **means;
  float *weights;
  float *logdets; // of full covariances.
};
struct SUBSPACE_UBM_HMM{ // just the parameters.
   int J; // #states.
   int I; // #mixtures.  Same for all states!
   int S; // Subspace dimension
   float ***vars; // [i][d][d].  Inverse full covariances per i.
   float *dets;   // determinants of variances.
   float **v;  // [j][s].  State-specific vectors v_j.
   float ***M; // [i][d][s].  Projection matrices M_i.
   float **w;  // [i][s].  Weight-projection vectors w_i.
};
```

Basic Subspace UBM system: fast likelihood evaluation

- System has 1000 full-covariance Gaussians per state: too slow? Not if we do it cleverly.
- First use "background model" to pre-prune index i from 1000 to, say, 10.
- Then can make each Gaussian computation as fast as $O(S)$ by appropriate precomputations:

$$
\begin{aligned}
\log p(\mathbf{x}_t|j,i) &= \log w_{ji} - 0.5\left(2\pi D + \log\det \boldsymbol{\Sigma}_i + (\mathbf{x}_t - \mu_{ji})^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x}_t - \mu_{ji})\right) \\
&= n_{ji} + n_i(t) + \mathbf{z}_i(t)^{-} \cdot \mathbf{v}_{jkm} \\
n_{ji} &= \log w_{ji} - 0.5\left(2\pi D + \log\det \boldsymbol{\Sigma}_i + \mu_{ji}^T \boldsymbol{\Sigma}_i^{-1} \mu_{ji}\right) \\
\mathbf{z}_i(t) &= \mathbf{M}_i^T \boldsymbol{\Sigma}_{ki}^{-1} \mathbf{x}_t \\
n_i(t) &= -0.5 \mathbf{x}_t^T \boldsymbol{\Sigma}_i^{-1} \mathbf{x}_t + z_{ki}(t)_{(D+1)}
\end{aligned}
$$

- The per-Gaussian normalizers n_{ji} actually take much more memory than the vectors $\mathbf{v}_i$, but still much less than the expanded means μ_{ji}.

Basic Subspace UBM system: fast likelihood evaluation (code)

```
void compute_likes(int D, float *x,
      vector<int> j_idx, vector<float> &ans, // j_idx=states needed.
      BACKGROUND_GMM *background, SUBSPACE_UBM_HMM *ubm, float **nji){
   vector<int> i_idx;  float *tmp = new float[D], *zi = new float[D+1];
   prune_on_frame(D, x, background, i_idx); // i_idx is pruned indices i.
   for(int n=0;n<i_idx.size();n++){
      int i=i_idx[n];
      m_v_prod(tmp, ubm->vars[i], x, D,D); // tmp=\Sigma_{ki}^{-1} x_t
      m_v_prod_transposed(zi, ubm->M[i], tmp, ubm->S+1, D); // ...
      // zi= transpose(M_i)*tmp
      float nit = -0.5*vmv_prod(x,ubm->vars[i],x,D,D) * zi[D]; // n_i(t)
      for(int m=0;m<j_idx.size();m++){
        int j=j_idx[m];
        float loglike = dot_prod(ubm->v[j], zi, D) + nit + nji[j][i];
        ans[m] = log_add(ans[m], loglike);
      }
    }
  }
}
```

Basic Subspace UBM system: optimization summary

- Optimization for vectors $\mathbf{v}_j$ (ignoring the effect on the weights w_{ji}) is a little like Speaker Adaptive Training with MLLR: need solution of a quadratic auxiliary function.

- Do not have to store quadratic term in auxiliary function: can work it out from counts.

- Optimization for projections $\mathbf{M}_i$ is like MLLR estimation in a shared-covariance system (because the is only one covariance $\mathbf{\Sigma}_i$ for each index i: more efficent than normal MLLR.

- The parts of the auxiliary function that relate to the weights w_{ji} (controlled by $\mathbf{v}_j$ and $\mathbf{w}_i$) are optimized by making a quadratic approximation to the nonlinearity of: $w_{ji} = \frac{\exp(\mathbf{w}_i \mathbf{v}_j^+)}{\sum_i \exp(\mathbf{w}_i \mathbf{v}_j^+)}$.

Basic Subspace UBM system: auxiliary function

- If we define $\mathbf{\Gamma}$ as all the system parameters, auxiliary function is:

$$\mathcal{Q}(\mathbf{\Gamma};\bar{\mathbf{\Gamma}}) = \sum_{t,i,j} \gamma_{ji}(t;\bar{\mathbf{\Gamma}}) \log(w_{ji}\ \mathcal{N}(\mathbf{x}_t;\mu_{ji},\mathbf{\Sigma}_i))$$

- The "occupancy" $0 \leq \gamma_{ji}(t;\bar{\mathbf{\Gamma}}) \leq 1$ is the posterior of state j, Gaussian i on time t (given current model parameters $\bar{\mathbf{\Gamma}}$).

- We will usually write this as just $\gamma_{ji}(t)$ (dependency on $\bar{\mathbf{\Gamma}}$ is implicit)

- This can be decomposed as

$$\gamma_{ji}(t) = \gamma_j(t) \frac{w_{ji}\mathcal{N}(\mathbf{x}_t;\mu_{ji},\mathbf{\Sigma}_i)}{\sum_i w_{ji}\mathcal{N}(\mathbf{x}_t;\mu_{ji},\mathbf{\Sigma}_i)}$$

  i.e. the posterior of the state times the probabilty of the Gaussian in the state.

- $0 \leq \gamma_j(t) \leq 1$ would be derived from a Forward Backward algorithm, or (more efficiently) from Viterbi (single-best-path) alignment.

- Code on next slide assumes state posteriors γ_j are given by Viterbi alignment of a system (so they would all be zero or one).

Basic Subspace UBM system: computing auxiliary function weights γ_{ji}

```
// i_idx, i_posterior are the output, a subset of indices i with posteriors.
void compute_posteriors(int D, float *x, vector<int> &i_idx,
        vector<float> &i_posterior, BACKGROUND_GMM *background,
        SUBSPACE_UBM_HMM *ubm, int j, float *nji, float min_post=0.01){
   vector<int> i_idx_tmp;  float *tmp = new float[D], *zi = new float[D+1];
   i_idx.clear(); i_posterior.clear(); // clear output arrays.
   prune_on_frame(D, x, background, i_idx); // i_idx_tmp is pruned indices i.
   float state_like=-1.0e+10;  i_posterior.resize(i_idx.size());
   for(int n=0;n<i_idx.size();n++){
     int i=i_idx[n];
     m_v_prod(tmp, ubm->vars[i], x, D,D); // tmp=\Sigma_{ki}^{-1} x_t
     m_v_prod_transposed(zi, ubm->M[i], tmp, ubm->S+1, D); // ...
     // zi= transpose(M_i)*tmp
     float nit = -0.5*vmv_prod(x,ubm->vars[i],x,D,D) * zi[ubm->S]; // n_i(t)
     float loglike = dot_prod(ubm->v[j], zi, ubm->S) + nit + nji[i];
     i_posterior[n]=loglike; // contains log-likes right now.
     state_like = log_add(state_like, like);
   }
   int m=0; for(int n=0;n<i_idx.size();n++){ // compute posteriors and prune.
     float post=exp(i_posterior[n]);
     if(post>=min_post){
       i_idx[m]=i_idx[n]; i_posterior[m]=post; m++;
    }}
   i_idx.resize(m); i_posterior.resize(m);
  }}
```

Basic Subspace UBM system: optimization of vectors $\mathbf{v}_j$: accumulation

- First demonstrating optimization of vectors when weights do not change with $\mathbf{v}_j$ (e.g. with $w_{ji} = \frac{1}{I}$).

- Auxiliary function in $\mathcal{V} = \mathbf{v}_1 \ldots \mathbf{v}_j$ (ignoring effect on weights and constant terms):

$$\mathcal{Q}(\mathcal{V}) = -0.5 \sum_{t,i,j} \gamma_{ji}(t)(\mathbf{x}_t - \mu_{ji})^T \mathbf{\Sigma}_i (\mathbf{x}_t - \mu_{ji})$$

$$\mathcal{Q}(\mathcal{V}) = -0.5 \sum_{t,i,j} \gamma_{ji}(t)(\mathbf{x}_t - \mathbf{M}_i \mathbf{v}_j^+)^T \mathbf{\Sigma}_i (\mathbf{x}_t - \mathbf{M}_i \mathbf{v}_j^+)$$

- Defining the data-count $\gamma_{ji} = \sum_t \gamma_{ji}(t)$,

$$\mathcal{Q}(\mathcal{V}) = \sum_j K' + \mathbf{v}_j^{+T} \mathbf{k}_j - 0.5 \mathbf{v}_j^+ \mathbf{G}_j \mathbf{v}_j$$

  , with $\mathbf{k}_j = \sum_{t,i} \mathbf{M}_i^T \mathbf{\Sigma}_i \mathbf{x}_t$ and $\mathbf{G}_j = \sum_i \gamma_{ji} \mathbf{M}_i^T \mathbf{\Sigma}_i \mathbf{M}_i$.

- Statistics needed are $\mathbf{k}_j$ and the data-counts γ_{ji} ($\mathbf{G}_j$ is worked out from counts).

- Very similar to Speaker Adatptive Training for MLLR (but update is more complicated due to offset $(\cdot^+)$).

Basic Subspace UBM system: optimization of vectors $\mathbf{v}_j$ (no weights): update

- Auxiliary function in one $\mathbf{v}_j$ is: $\mathcal{Q}(\mathbf{v}_j) = \mathbf{v}_j^{+T}\mathbf{k}_j - 0.5\mathbf{v}_j^{+}\mathbf{G}_j\mathbf{v}_j$.

- Equivalent to $\mathcal{Q}(\mathbf{v}_j) = \mathbf{v}_j \cdot (\mathbf{k}_j^{-} - \mathbf{g}_{j(D+1)}^{-}) - 0.5\mathbf{v}_j^{+}\mathbf{G}_j^{--}\mathbf{v}_j$, where $\mathbf{x}^{-}$ means $\mathbf{x}$ with last element removed, $\mathbf{g}_{j(D+1)}$ is last row of $\mathbf{G}_j$, and $\mathbf{M}^{--}$ is $\mathbf{M}$ with last row and column removed.

- The superscripts $\cdot^{+}$, $\cdot^{-}$ and $\cdot^{--}$ are my own personal notation, not standard!

- Solution is $\mathbf{v}_j = (\mathbf{G}_j^{--})^{-1}\left(\mathbf{k}_j^{-} - \mathbf{g}_{j(D+1)}^{-}\right)$

- The actual update we do takes into account the effect on the weights!

Basic Subspace UBM system: optimization of matrices $\mathbf{M}_i$

- The projection matrices $\mathbf{M}_i$ are a little similar to MLLR matrices that transform means.

- Because each variance $\mathbf{\Sigma}_i$ is full but each $\mathbf{M}_i$ is only associated with one $\mathbf{\Sigma}_i$, the optimization of $\mathbf{M}_i$ is the same as optimizing MLLR on a system with a single, shared full covariance matrix.

- The accumulation and update are very simple. The only statistics we need are, for each $\mathbf{M}_i$, a matrix $\mathbf{K}_i$ which is the same dimension as $\mathbf{M}_i$ and dictates the linear term in the (quadratic) objective function. This relates to the matrix $\mathbf{K} = \begin{bmatrix} \mathbf{k}_i \\ \vdots \\ \mathbf{k}_D \end{bmatrix}$ which we normally accumulate in MLLR.

Basic Subspace UBM system: optimization of within-class variances $\boldsymbol{\Sigma}_i$

- Estimation of the within-class variances $\boldsymbol{\Sigma}_i$ is trivial

- Just use the weighted scatter of the data points around the means μ_{ji}.

Basic Subspace UBM system: optimization of weights w_{ji}, introduction.

- The sufficient statistics to optimize the weights w_{ji} are just the summed data counts γ_{ji}.

- But the weights w_{ji} are not parameters of the model. They are controlled by the state-specific parameters $\mathbf{v}_j$ and the globally shared paramters $\mathbf{w}_i$:
$w_{ji} = \frac{\exp(\mathbf{w}_i \mathbf{v}_j^+)}{\sum_i \exp(\mathbf{w}_i \mathbf{v}_j^+)}$.

- The reason we do this is to reduce the number of parameters in the model: why waste 1000 parameters per state on the weights when we are using only e.g. 50 for the means and the weights are typically considered "less important" than the means. (E.g. in speaker ID tasks, weights are not estimated at all).

- The terms in the auxiliary function that relates to the weights are as follows

$$\mathcal{Q}(\Gamma; \hat{\Gamma}) = \ldots + \sum_{i,j} \gamma_{ji} \left(\mathbf{w}_i \mathbf{v}_j^+ - \log \sum_i \exp(\mathbf{w}_i \mathbf{v}_j^+) \right).$$

- There are two further steps that we use to get this into a quadratic form that is easily optimized (see next two slides).

Basic Subspace UBM system: optimization of weights w_{ji}, step one.

- We are trying to manipulate the auxiliary function terms in the weights into a quadratic form.

- The first step uses the inequality $1-(x/\bar{x}) \leq -\log(x/\bar{x})$ (which is an equality at $x=\bar{x}$).

- The aim here is to get rid of the log in the expresssion $\log \sum_i \exp(\mathbf{w}_i \mathbf{v}_j^+)$

- The letter x in the inequality corresponds to $\sum_i \exp(\mathbf{w}_i \mathbf{v}_j^+)$.

- We can use this to derive the new auxiliary function

$$\mathcal{Q}'(\Gamma;\hat{\Gamma}) = \ldots + \sum_{i,j} \gamma_{ji} \left(\mathbf{w}_i \mathbf{v}_j^+ - \frac{\sum_i \exp(\mathbf{w}_i \mathbf{v}_j^+)}{\sum_i \exp(\bar{\mathbf{w}}_i \bar{\mathbf{v}}_j^+)} \right).$$

.

- This is related to the old auxiliary function $\mathcal{Q}$ by the same kinds of inequalities that our normal $\mathcal{Q}$ is related to $\mathcal{P}$, and we know if we increase $calQ'$ we increase $\mathcal{Q}$.

- This step involves discarding some terms in the old parameters $\hat{\Gamma}$ that will not affect the optimization.

Basic Subspace UBM system: optimization of weights w_{ji}, step two.

- We want to get rid of the exponential function in $\mathcal{Q}'(\Gamma;\hat{\Gamma})$.
- We do this with a quadratic approximation: a quadratic approximation to $\exp(x)$ around $x = x_0$ is: $\exp(x) \simeq \exp(x_0)(1 + (x - x_0) + 0.5(x - x_0)^2)$. This is just the second order Taylor series around x_0.
- We won't write down the altered auxiliary function $\mathcal{Q}''(\Gamma;\hat{\Gamma})$ at this point because the expression is not very pretty.
- However, given $\mathcal{Q}''(\Gamma;\hat{\Gamma})$ it easy to optimize $\mathbf{v}_j$ or $\mathbf{w}_i$.
- It is important to note that going to the maximum of the auxiliary function $\mathcal{Q}''(\Gamma;\hat{\Gamma})$ no longer guarantees increasing the original auxiliary function $\mathcal{Q}(\Gamma;\hat{\Gamma})$.
- However, we can measure the value of $\mathcal{Q}(\Gamma;\hat{\Gamma})$ because we have the sufficient statistics γ_{ji}, so if it decreases we can take, say half the proposed change and check again.
- The optimization for $\mathbf{w}_i$ involves just the steps mentioned here.
- The optimization for $\mathbf{v}_j$ also involves terms relating to the means, but those terms have the same (quadratic) form so we can simply add the two types of terms together.

Basic Subspace UBM system: optimization, overall process.

- There are various different kind of parameters to optimize: the state-specific vectors $\mathbf{v}_j$, the projection matrices $\mathbf{M}_i$, the within-class variances $\mathbf{\Sigma}_i$, the weight-projection vectors $\mathbf{w}_i$.

- The derivation of the auxiliary function would suggest to optimize these separately, e.g. all the $\mathbf{v}_j$ on one iteration, all the $\mathbf{M}_i$ on the second, all the $\mathbf{\Sigma}_i$ on the third, etc.: in general, it is hard to prove that the process will converge if we do them more than one at a time.

- In practice it is possible to optimize them all simultaneously.

- Sometimes instabilities occur (especially in later versions where we introduce even more parameters) but they can be controlled by introducing a constant $0 < \nu \leq 1$ that interpolates between the new and updated parameters

Basic Subspace UBM system: initialization.

- In my previous experiments with this type of system, the initialization was based on storing mean statistics μ_{ij} for each i and j, based on Gaussian posterior probabilities derived from the "background model".

- I.e. using $\gamma_{ij} = \gamma_i \gamma_j$ where γ_i is the posterior of Gaussian i in the "background model" and γ_j is the (zero or one) state posterior based on the alignment of a baseline system.

- This allowed many iterations of the types of update described above, in memory, without re-accessing the data. It is possible to start this from a random initialization of the vectors.

- The reason why keeping these large statistics in memory is not possible in general is twofold:

  1. We need to discard means with small counts because of memory constraints, which leads to an approximate answer
  2. Later we will introduce "substates" which increases the amount of memory we would need.

- For simplicity, probably the way we will do it during this workshop is skip the special initialization phase but initialize the parameters $\mathbf{M}_i$ in a nonrandom way so that the $\mathbf{v}_j$ just correspond initially to an offset on the means.

Basic Subspace UBM system: initialization and training of of "background GMM"

- In the previous slide, we mentioned that the initial posteriors are based on the posteriors of the "background GMM".
- This is a generic GMM that we have trained on all of speech.
- It is also used for pruning.
- In previous experiments, it worked best to initialize this by clustering all the Gaussians in a trained system; this was then trained for a few iterations on speech data.
- This can also optionally be further trained during training of the rest of the model parameters, based on posteriors of Gaussians in the model with the same index i.

Subspace UBM system: (previously used) extensions to the basic model

- Introduce "sub-states" $1 \leq m \leq M_j$ within each state j. Each sub-state has its own vector $\mathbf{v}_{jm}$ and its own weight c_{jm}. Note M_j has no connection to $\mathbf{M}_i$.

- Analogous to a mixture of Gaussians– except each vector $\mathbf{v}_{jm}$ expands to a mixture of Gaussians so it is a mixture of mixtures of Gaussians.

- Introduce "speaker factors":

$$\mu_{jmi}^{(s)} = \mathbf{M}_i \mathbf{v}_{jm}^{+} + \mathbf{N}_i \mathbf{v}^{(s)+}$$

- The matrices $\mathbf{N}_i$ are a separate set of projection matrices that define a "speaker subspace".

- The vectors $\mathbf{v}^{(s)}$ are speaker-specific subspace vectors that can be estimated with very little data (only about 50 or so parameters).

Subspace UBM system: prior results

- On the next page we will show three tables of results.
- The data-sets were:
  - Small English system (50 hours of Broadcast News)
  - Large Arabic system (about 1000 hours)
  - Large Mandarin system (about 700 hours)
- For the English system we show results with and without discriminative training.
- For the large systems we show results only with discriminative training (we did model-space discriminative training for the "Subspace based" system, and model and feature-space discriminative training for the baseline).
- Trends were: much more improvement with small dataset, more improvement with Maximum Likelihood than discriminative training.

Subspace UBM system: prior results (tables)

Conditions:	Baseline	Subspace
VTLN+fMLLR+MLLR	24.3%	19.6%
+fMMI+MMI	18.2%	17.3%

Table 2: Subspace-adapt vs normal system on 50 hours English BN (test: RT'04).

System:	Dev07	Dev08	Eval07 (unsequestered)
VxU.vfr	10.0%	11.5%	14.4%
SUBxU.vfr	9.5%	11.1%	14.2%

Table 3: Arabic evaluation system, January 2009.

System:	Dev07	Dev08	Eval07 (unsequestered)
TxN.vfr	9.7%	8.6%	9.2%
SUBxN.vfr	9.7%	8.3%	9.4%

Table 4: Mandarin evaluation system, January 2009.

Subspace UBM system: further extensions

- There are some extensions to the approach described above that we intend to try during the workshop (we will probably think of more too).
- One is to have a mixture of the entire model described above, so we introduce an extra index k, representing coarse regions of acoustic space.
  - Each state j has at least one mixture for each k.
  - This modeling approach is more memory efficient for systems with multiple mixtures (fewer normalizers to store).
  - We can have a separate constrained MLLR transform for each k.
- Another extension regards contrained MLLR (speaker adaptive feature transforms):
  - We have worked out a method for doing a subpace version of constrained MLLR, that should be efficient. It requires fewer per-speaker parameters to estimate.
  - This makes it easy to work out multiple constrained MLLR matrices (i.e., for each k) with relatively little data.
- It is possible to train the generic (non-state-specific) parameters using multiple domains or languages and the state-specific ones specific to the domain or language, which may make better use of out-of-domain data.

References and further reading

- Previous work with MAP-based adaptation: "Universal Background Model Based Speech Recognition", Povey D., Chu S. M. and Varadarajan, B., ICASSP 2008. `http://dpovey.googlepages.com/icassp08_fmllr.pdf`

- Book chapter containing the tables included here (results on prior work on subspace-based approach): "Approaches to Speech Recognition based on Speaker Recognition Techniques", Povey D., Chu S. M., Pelecanos J., book chapter in forthcoming book on GALE project. `http://dpovey.googlepages.com/ubmdoc.pdf`

- Technical report containing the mathematical details omitted from the above chapter: "Subspace Gaussian Mixture Models for Speech Recognition", Povey D., Microsoft Research technical report MSR-TR-2009-64, `http://research.microsoft.com/apps/pubs/default.aspx?id=80931`

- Technical report with a lot of detail on the approaches proposed for the workshop: "A Tutorial-style introduction to Factor Analyzed GMMs for Speech Recognition", D. Povey, will be published as a technical report. `http://dpovey.googlepages.com/ubmtutorial.pdf`