LF-MMI: a theoretical tutorial

Peter Bell

November 27, 2018
This tutorial

- Background and motivation
- The fundamentals
- The extra tricks
- Lots of equations!
The fundamental equation of speech recognition

\[ \hat{Y} = \arg \max_Y p(Y|X) \]
\[ = \arg \max_Y p(X, Y) \]
\[ = \arg \max_Y p(X|Y)p(Y) \]

\[ X = (x_1, \ldots x_T) \] and \[ Y = (y_1, \ldots y_N) \] are sequences of acoustic observations and words respectively.
The fundamental equation of speech recognition

\[ \hat{Y} = \arg \max_{Y} p(Y|X) \]
\[ = \arg \max_{Y} p(X, Y) \]
\[ = \arg \max_{Y} p(X|Y)p(Y) \]

\( X = (x_1, \ldots x_T) \) and \( Y = (y_1, \ldots y_N) \) are sequences of acoustic observations and words respectively.

Maximum likelihood training: maximise \( F_{ML}(\theta) = \sum_u p_\theta(X_u|Y_u) \).
The fundamental equation of speech recognition

\[ \hat{Y} = \arg \max_Y p(Y|X) \]
\[ = \arg \max_Y p(X, Y) \]
\[ = \arg \max_Y p(X|Y)p(Y) \]

\( X = (x_1, \ldots, x_T) \) and \( Y = (y_1, \ldots, y_N) \) are sequences of acoustic observations and words respectively.

Maximum likelihood training: maximise \( F_{ML}(\theta) = \sum_u p_\theta(X_u|Y_u) \).

This is sequence training!
Traditional approach

Use an HMM as the sequence model

This is more for convenience than anything else!
The HMM comes with great algorithms for

- finding $p(q_t|X)$, the probability of being in state $q$ at time $t$ (forward-backward)
- finding the most likely state sequence (Viterbi)

ML sequence training = *Baum-Welch re-estimation*
The HMM comes with great algorithms for

- finding $p(q_t|X)$, the probability of being in state $q$ at time $t$ (forward-backward)
- finding the most likely state sequence (Viterbi)

ML sequence training = *Baum-Welch re-estimation*

This is a special case of the EM algorithm
Problem - the HMM assumptions are rubbish!

- Observations are absolutely not conditionally independent, given the hidden state.
- When states are phone-based, observations are not independent of past/future phone states, given the current state.

Two useful hacks:

- Expand the state space to incorporate phonetic context (makes the decoder more complicated, but we can cope...)
- Augment the feature vector to incorporate adjacent acoustic features using deltas, or feature splicing + HLDA, attempting to keep individual elements uncorrelated.

Oops, this massively over-states the framewise probabilities, so throw in an acoustic scaling fudge factor of about 12.0.
Problem - the HMM assumptions are rubbish!

- Observations are absolutely not conditionally independent, given the hidden state
- When states are phone-based, observations are not independent of past/future phone states, given the current state

Two useful hacks:

- Expand the state space to incorporate phonetic context (makes the decoder more complicated, but we can cope...)
- Augment the feature vector to incorporate adjacent acoustic features using deltas, or feature splicing + HLDA, attempting to keep individual elements uncorrelated
Problem - the HMM assumptions are rubbish!

- Observations are absolutely not conditionally independent, given the hidden state
- When states are phone-based, observations are not independent of past/future phone states, given the current state

Two useful hacks:

- Expand the state space to incorporate phonetic context (makes the decoder more complicated, but we can cope...)
- Augment the feature vector to incorporate adjacent acoustic features using deltas, or feature splicing + HLDA, attempting to keep individual elements uncorrelated

Oops, this massively over-states the framewise probabilities, so throw in an acoustic scaling fudge factor of about 12.0
Still not enough? Switch training criterion

- Maximum likelihood is theoretically optimal (even for classification), but only when the model is correct.
Still not enough? Switch training criterion

- Maximum likelihood is theoretically optimal (even for classification), but only when the model is correct.
- If not, an explicitly discriminative training criterion might be better.
Still not enough? Switch training criterion

- Maximum likelihood is theoretically optimal (even for classification), but only when the model is correct
- If not, an explicitly discriminative training criterion might be better
- MCE is a natural choice for classification, but not for sequences
Still not enough? Switch training criterion

- Maximum likelihood is theoretically optimal (even for classification), but only when the model is correct.
- If not, an explicitly discriminative training criterion might be better.
- MCE is a natural choice for classification, but not for sequences.
- Try mutual information (Bahl et al, 1986).
The MMI criterion is given by:

\[ F_{\text{MMI}}(\theta) = \sum_u \log \frac{p(X_u, Y_u)}{p(X_u)p(Y_u)} \]

\[ = \sum_u \left[ \log \frac{p(X_u, Y_u)}{p(X_u)} - \log p(Y_u) \right] \]

\[ = \sum_u \left[ \log \frac{p_\theta(X_u | Y_u)^\kappa P(Y_u)}{\sum_y p_\theta(X_u | y)^\kappa p(y)} - \log p(Y_u) \right] \]

Where \( p_Y \) is constant, this is equivalent to conditional maximum likelihood:

\[ F_{\text{CML}}(\theta) = \sum_u \log \frac{P(Y_u | X_u)}{p(X_u)p(Y_u)} \]

\[ = \sum_u \log \frac{p_\theta(X_u | Y_u)^\kappa P(Y_u)}{\sum_y p_\theta(X_u | y)^\kappa p(y)} - \log p(Y_u) \]
\[ F_{\text{MMI}}(\theta) = \sum_u \log \frac{p(X_u, Y_u)}{p(X_u)p(Y_u)} \]

\[ = \sum_u \left[ \log \frac{p(X_u, Y_u)}{p(X_u)} - \log p(Y_u) \right] \]

\[ = \sum_u \left[ \log \frac{p_{\theta}(X_u|Y_u)\kappa P(Y_u)}{\sum_y p_{\theta}(X_u|y)\kappa p(y)} - \log p(Y_u) \right] \]

Where \( p(y) \) is constant, this is equivalent to conditional maximum likelihood:

\[ F_{\text{CML}}(\theta) = \sum_u \log P(Y_u|X_u) = \sum_u \log \frac{p_{\theta}(X_u|Y_u)\kappa P(Y_u)}{\sum_y p_{\theta}(X_u|y)\kappa p(y)} \]
Example

The diagram above represents a scatter plot with three distinct clusters of data points. The x-axis ranges from -6 to 8, while the y-axis ranges from -4 to 3. The clusters are color-coded: red, blue, and green, each representing a different group of data points. The pattern and distribution of these points suggest a complex relationship between the variables being measured.
Full covariance model with ML training

![Graph showing data distribution with full covariance model and ML training.](image)
Diagonal covariance model with ML training
Diagonal covariance model with MMI training
Frame-level models are pretty good, sequence-level models are poor ⇒ need to operate at the sequence level.

Hard part is estimating the denominator probabilities over the sequence

$$\sum_{Y} p_{\theta}(X_{u}|y)p(Y)$$

for anything beyond small tasks
Vari...1990s-2000s.

- Extended Baum-Welch (Gopalakrishnan et al, 1989)
- Version for GMM-HMMs (Normandin & Morega, 1991)
- Small vocabulary tasks (Normandin et al, 1994)
- Initial work with lattices using gradient descent (Valtchev et al, 1996, 1997)
- Lattice forward-backward with appropriate scaling, smoothing, LM (Povey & Woodland, 2000-2003)
Various attempts to get MMI training to work 1990s-2000s.

- Extended Baum-Welch (Gopalakrishnan et al, 1989)
- Version for GMM-HMMs (Normandin & Morega, 1991)
- Small vocabulary tasks (Normandin et al, 1994)
- Initial work with lattices using gradient descent (Valtchev et al, 1996, 1997)
- Lattice forward-backward with appropriate scaling, smoothing, LM (Povey & Woodland, 2000-2003)

Dan is very good at getting things to work!
DNNs are better at incorporating wider context because they can more easily model correlated input features.

We’re still breaking the conditional independence assumption – scaling factor still needed

Fortunately discriminative training (rebranded “sequence training”) can be used as before (Vesely et al, 2013).
Regardless of the method used, we need to compute two sets of occupancy probabilities:

**Numerator:** \( \gamma_{j}^{\text{num}}(t) = p(q_t = j | X_u, M_u^{\text{num}}) \)

**Denominator:** \( \gamma_{j}^{\text{den}}(t) = p(q_t = j | X_u, M_u^{\text{den}}) \)

\( M_u^{\text{num}} \) represents the HMM state sequence corresponding to the transcription of utterance \( u \)

\( M_u^{\text{num}} \) represents all possible HMM state sequences for \( u \)
The fundamentals of MMI training

Regardless of the method used, we need to compute two sets of occupancy probabilities:

**numerator:** \( \gamma_{num}^j(t) = p(q_t = j | X_u, \mathcal{M}_u^{num}) \)

**denominator:** \( \gamma_{den}^j(t) = p(q_t = j | X_u, \mathcal{M}_u^{den}) \)

\( \mathcal{M}_u^{num} \) represents the HMM state sequence corresponding to the transcription of utterance \( u \)

\( \mathcal{M}_u^{num} \) represents all possible HMM state sequences for \( u \)

Computing \( \gamma_{den}^j(t) \) is hard
The fundamentals of MMI training

Regardless of the method used, we need to compute two sets of occupancy probabilities:

**numerator:**  \( \gamma_{j}^{\text{num}}(t) = p(q_t = j | X_u, M_{u}^{\text{num}}) \)

**denominator:**  \( \gamma_{j}^{\text{den}}(t) = p(q_t = j | X_u, M_{u}^{\text{den}}) \)

\( M_{u}^{\text{num}} \) represents the HMM state sequence corresponding to the transcription of utterance \( u \)

\( M_{u}^{\text{num}} \) represents all possible HMM state sequences for \( u \)

Computing \( \gamma_{j}^{\text{den}}(t) \) is hard

Computing \( \gamma_{j}^{\text{num}}(t) \) is the standard forward-backward algorithm. (But we need to make sure that the statistics are consistent with \( \gamma_{j}^{\text{den}}(t) \) )
Using the occupancy probabilities

**GMM:** accumulate 0th, 1st and 2nd order statistics for numerator and denominator

\[
\theta^{(0)}_j(X) = \sum_t \gamma_j(t), \quad \theta^{(1)}_j(X) = \sum_t \gamma_j(t)x_t, \quad \theta^{(2)}_j(X) = \sum_t \gamma_j(t)x_tx_T
\]

**DNN:**

\[
\frac{\partial F_{\text{MMI}}}{\partial \log p(x_t|j)} = \gamma_j^{\text{num}}(t) - \gamma_j^{\text{den}}(t)
\]

\[
\frac{\partial F_{\text{MMI}}}{\partial \log a_t(s)} = \sum_j \frac{\partial F_{\text{MMI}}}{\partial \log p(x_t|j)} \frac{\partial \log p(x_t|j)}{\partial a_t(s)}
\]

\[
= (\gamma_j^{\text{num}}(t) - \gamma_j^{\text{den}}(t)) \frac{\partial \log p(x_t|j)}{\partial a_t(s)}
\]
Standard forward backward algorithm

Forward probability $\alpha_j(t) = p(x_1, \ldots, x_t | q_t = j)$

$$\alpha_j(t) = \sum_i \alpha_i(t-1) a_{ij} p(x_t | q_t = j)$$

Backward probability $\beta_j(t) = p(x_{t+1}, \ldots, x_T | q_t = j)$

$$\beta_i(t) = \sum_j a_{ij} p(x_{t+1} | q_{t+1} = j) \beta_j(t+1)$$

Then compute state occupancy as

$$\gamma_j(t) = \frac{\alpha_j(t) \beta_j(t)}{\alpha_E(T)}$$
Lattice-based MMI

- Approximate $\sum_Y$ with a sum over a lattice
- Generate lattice for each utterance using an initial model
- Use a weak language model
- But attempt to minimise the size of the lattice
- Derive phone arcs from the lattice
Define forward and backward probabilities over phone arcs $r$ with known start and end times

\[ \alpha_r = \sum_{r' \rightarrow r} \alpha_{r'} a_{r',r} p(r) \]
\[ \beta_r = \sum_{r \rightarrow r'} a_{rr'} p(r') \alpha_{r'} \beta(r') \]
\[ \gamma_r = \alpha_r \beta_r \]

Where $p(r)$ denotes the log likelihood over the arc

Use standard FB algorithm within arcs to compute state occupancies for time $t$
“Purely sequence-trained models for ASR based on lattice-free MMI”
“Purely sequence-trained models for ASR based on lattice-free MMI”

A seminal paper, or just “getting it to work”? 
A seminal paper

- Solves a fundamental problem – a practical method for computing HMM “true” state posteriors
- Uses this to train a properly normalised sequence model, trained with MMI from the beginning
- Removes the need for an acoustic scaling fudge factor
Getting it to work...

The core idea

- Parallelise denominator forward-backward computation on a GPU
- Replace word-level LM with a 4-gram phone LM for efficiency
- Reduce the frame rate
  - might be a good idea for other reasons...
- Changes to HMM topology motivated by CTC
Getting it to work...

The core idea

- Parallelise denominator forward-backward computation on a GPU
- Replace word-level LM with a 4-gram phone LM for efficiency
- Reduce the frame rate
  - might be a good idea for other reasons...
- Changes to HMM topology motivated by CTC

Extra tricks

- Train on small fixed-size chunks (1.5s)
  - probably enough to counter the flaws in the conditional independence assumption
- Careful optimisation of denominator FST to minimise the size
- Various types of regularisation
Connections with CTC

- Replace standard 3-state HMM with topology that can be traversed in a single frame
- View HMM-DNN system as directly generating the labels \((s_1, \ldots s_T)\)
  - neural net outputs considered as log pseudo-likelihoods
  - probabilities normalised over the sequence, rather than per frame as in CTC (or CE training)
- Different PDFs associated with forward loops and self-loops
- HMMs are represented as FSTs (as you would expect in Kaldi)
HMM topologies

Standard topology

CTC-like topology
Denominator FST

- LM is essentially a 3-gram phone LM
- No pruning and no backoff to minimise the size
  - Use of unpruned 3-grams means that there is always a 2-word history.
  - Minimises the size of the recognition graph when phonetic context is incorporated
- Addition of a fixed number of 4-grams
- Conversion to $HCLG$ FST in the normal way
- $HCLG$ size reduced by a series of FST reversal, weight pushing and minimisation operations, followed by epsilon removal
The normalisation FST

- The phone-LM assumes that we are starting at the beginning of an utterance → not suitable for use with 1.5s chunks
- Need to adjust the initial probabilities for each HMM state
- Iterate 100 times through the denominator FST to get better initial occupancy probabilities

\[ \alpha_j^{(n)}(0) = \sum_i a_{ij} \alpha_i^{(n-1)}(0) \]

- Add a new initial state to the denominator FST that connects to each state with the new probabilities → the “normalisation FST”
Numerator FST

Originally

- Used GMM system to generate lattices for training utterances, representing alternate pronunciations
- Lattice determines which phones are allowed to appear in which frames, with an additional tolerance factor
- Constraints encoded as an FST
- Compose with the normalisation FST to ensure that the logprob objective function is always < 0
More recently, unconstrained numerator found to work better (Hadian, Povey et al, IEEE SLT, 2018)
Specialised forward-backward algorithm

- Work with probabilities rather than log-probs to avoid expensive log/exp operations
- Numeric overflow and underflow is a big problem
- Two specialisations:
  - re-normalise probabilities at every time step
  - the “leaky HMM” - gradual forgetting of context
Probability re-normalisation

Define

\[ A(t) = \sum_i \alpha_i(t) \]

Forward/backward passes become

\[ \alpha_j(t) = \sum_i \alpha_i(t-1) a_{ij} p(x_t|q_t = j)/A(t) \]

\[ \beta_i(t) = \sum_j a_{ij} p(x_{t+1}|q_{t+1} = j) \beta_j(t+1)/A(t+1) \]

Add a correction factor to the total log probability:

\[ \log p(X) = \log \alpha_E(T) + \sum_t \log A(t) \]
Leaky-HMM

The above is still susceptible to overflow in backward computation. Introduce a leak-probability, $\eta$, of transition to any other state.
Leaky-HMM

The above is still susceptible to overflow in backward computation. Introduce a leak-probability, $\eta$, of transition to any other state. Probability of transition to state $i$ given by

$$\eta \alpha_i(0)$$

(where these are the iterated occupancy probabilities)
The above is still susceptible to overflow in backward computation.

Introduce a leak-probability, $\eta$, of transition to any other state.

Probability of transition to state $i$ given by

$$\eta \alpha_i(0)$$

(where these are the iterated occupancy probabilities)

Define

$$\hat{\alpha}_i(t) = \alpha_i(t) + \eta A(t) \alpha_i(0)$$
The above is still susceptible to overflow in backward computation.

Introduce a leak-probability, $\eta$, of transition to any other state

Probability of transition to state $i$ given by

$$\eta \alpha_i(0)$$

(where these are the iterated occupancy probabilities)

Define  
\[ \hat{\alpha}_i(t) = \alpha_i(t) + \eta A(t) \alpha_i(0) \]

Forwards pass:

\[ \alpha_j(t) = \sum_i \hat{\alpha}_i(t-1) a_{ij} p(x_t|q_t = j) / A(t) \]

\[ p(X) = \sum_i \hat{\alpha}_i(T) \]
Define backwards variables:

\[ \hat{\beta}_i(T) = 1/p(X) \]

\[ B(t) = \eta \sum_i \alpha_i(0) \hat{\beta}_i(t) \]

\[ \beta_i(t) = \hat{\beta}_i(t) + B(t) \]

Backwards recursion:

\[ \hat{\beta}_i(t) = \sum_j a_{ij} p(x_{t+1}|q_{t+1} = j) \beta_j(t + 1)/A(t + 1) \]

\[ \gamma_j(t) = \sum_j a_{ij} \hat{\alpha}_i(t) p(x_t|q_t = j) \beta_j(t + 1)/A(t) \]
Regularisation

- use standard CE objective as a secondary task
  - all but the final hidden layer shared between tasks
  - use numerator posteriors for convenience
- $l_2$ norm penalty on the main output
- Leaky HMM (mentioned earlier)
Other uses

Ability to properly compute state posterior probabilities over arbitrary state sequences also opens possibilities for

- Semi-supervised training
- Cross-model student-teacher training

... where sequence information is critical
Other uses

Ability to properly compute state posterior probabilities over arbitrary state sequences also opens possibilities for

- Semi-supervised training
- Cross-model student-teacher training

... where sequence information is critical

Anything else..?