DNN-HMM acoustic models for speech recognition

Robert von Zweygbergk
Rasmus Törnqvist
Table of contents

- HMM-GMM networks
  - What is GMM?
- HMM-DNN networks
  - What are DNNs?
  - How are they trained
  - Pretraining & RBMs
  - Deep Belief Networks & Interfacing
- Results
- References
HMM-GMM networks

• Hidden Markov Models and Gaussian Mixture Models are the most commonly used models in speech recognition

• HMMs used to account for variability in speech

• GMMs used to determine how well each state of the HMM corresponds to the coefficients representing the acoustic input
GMM networks

\[ p(y_i) = \Phi^v(y_i) = \sum_{k=1}^{K^v} \alpha_k^v N(y_i; \mu_k^v, \Sigma_k^v) \]

\[ \mu_k^v = \begin{bmatrix} \mu_k^{v,x} \\ \mu_k^{v,f} \end{bmatrix} \quad \text{and} \quad \Sigma_k^v = \begin{bmatrix} \Sigma_k^{v,xx} & \Sigma_k^{v,xf} \\ \Sigma_k^{v,fx} & \Sigma_k^{v,ff} \end{bmatrix} \]

\[ \hat{f}_i = \sum_{k=1}^{K^v} h_k(x_i) \left( \mu_k^{v,f} + \Sigma_k^{v,fx} \left( \Sigma_k^{v,xx} \right)^{-1} (x_i - \mu_k^{v,x}) \right) \]

(1) Gaussian mixture model

(2) Mean vector \( \mu \) comprised of the mean vectors of the voiced MFCC vectors and mean of the fundamental frequency in the cluster \( k \)

(3) Predicted fundamental frequency. \( h(x) \) is the posterior probability of the MFCC coming from that cluster
GMM networks

- GMM requires uncorrelated inputs (e.g. MFCC)
- GMMs inefficient for modeling speech data (Homework: Why?)
HMM-DNN networks

- Instead of using GMMs, use DNNs to produce posterior probabilities over HMM states as output
- Feed-forward networks used
- Several frames of coefficients as inputs
- DNNs with many hidden layers have been shown to outperform GMMs on a variety of speech recognition benchmarks
- Can be trained as DBN --- Correlated inputs OK
Deep Neural Network

- Used to detect patterns in data
- Deep network -> many hidden layers
- More layers -> more complex data
Deep Neural Network

• Each hidden unit, $j$, maps its total input from the layer below, $x_j$, to the scalar state, $y_j$, that it sends to the layer above

$$y_j = \text{logistic}(x_j) = \frac{1}{1 + e^{-x_j}}, \quad x_j = b_j + \sum_i y_i w_{ij}$$

• $b_j$ is the bias of unit $j$, $i$ is an index over units in the layer below, and $w_{ij}$ is the weight for the connection to $j$ from $i$ in the layer below
Deep Neural Network

- For multiclass classification, the "softmax" linearity is used to get a class probability $p_j$

$$ p_j = \frac{\exp(x_j)}{\sum_{k} \exp(x_k)} $$
DNN training

• Discriminative training (DT)
  • By backpropagating derivatives of a cost function that measures the discrepancy between target outputs and actual outputs produced for each training case
  • When using softmax, the natural cost function $C$ is

$$C = - \sum_j d_j \log p_j$$

• For large training sets, a random "minibatch", $t$, of training sets is often used
DNN training

• Large weights can be penalized to reduce overfitting
  • Early stopping is also used

• In DNNs with full connectivity between adjacent layers, the initial weights are given small random values, to prevent all the hidden units from one layer getting the same gradient

• DNN with many layers hard to optimize
  • The backpropagated gradients will have very different magnitudes in different layers, if the initial weights aren't chosen carefully

• DNNs may generalize poorly to held out test data
Generative pretraining

• Start by training feature detector to be good at modeling the structure of the input data, instead of training it to discriminate between classes

• Learn one layer of feature detectors at a time
  • States of the feature detectors in one layer are the data for training the next layer

• After pretraining, use the multiple layers of feature detectors as a starting point for discriminative fine tuning
  • Slightly adjust the weights of the DNN by backpropagation
Generative pretraining

• A single layer can be learned by fitting a generative model with one layer of latent variables to the input data
  • Two classes of generative models normally used

• Directed model
  • One set of parameters to define a prior distribution over the latent variables
  • One set of parameters to define the conditional distributions of the observable variables given the values of the latent variables

• Undirected model
  • Single set of parameters, for e.g. RBM, explained later
Generative pretraining

• Creates some high-level features that won't be of use
  • Others, however, will be far more useful than the raw data

• Allows the fine-tuning to make rapid progress

• Significantly reduces overfitting
Restricted Boltzmann Machines

- RBMs are two-layer neural nets with no intra-layer communication.

- Energy

\[
E(v, h) = - \sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}
\]

- Probability that the network assigns to a visible vector, \( v \)

\[
p(v) = \frac{1}{Z} \sum_h e^{-E(v,h)}
\]
Restricted Boltzmann Machines

• The two conditional distributions

\[ p(h_j = 1 | v) = \text{logistic}(b_j + \sum_i v_i w_{ij}) \]

\[ p(v_i = 1 | h) = \text{logistic}(a_i + \sum_j h_j w_{ij}). \]

P( binary state of each hidden layer = 1 given a randomly selected training case, v )

Unbiased sample of the state of a visible unit, given a hidden vector h

• These can be used for contrastive divergence
Restricted Boltzmann Machines

- CD learning rule: \[ \Delta w_{ij} = \epsilon (\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{recon}}). \]

- This is done by doing a reconstruction by setting \( v_i \) to one according to the conditional probability, then the states of the hidden units are updated again.

- Real-valued data is better modeled by GRBMs.
Gaussian Restricted Boltzmann Machine

• Same idea, different equations

\[ E(v, h) = \sum_{i \in \text{vis}} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j \in \text{hid}} b_j h_j - \sum_{i,j} \frac{v_j}{\sigma_i} h_i w_{ij} \]

• \( h_j \) still binary, \( v_i \) Gaussian noise with a standard deviation of \( \sigma \)
Gaussian Restricted Boltzmann Machine

- Conditional distributions for GRBMs:

\[
p(h_j | v) = \text{logistic} \left( b_j + \sum_i \frac{v_i}{\sigma_i} w_{ij} \right)
\]

\[
p(v_i | h) = \mathcal{N} \left( a_i + \sigma_i \sum_j h_j w_{ij}, \sigma_i^2 \right)
\]
Deep Belief Networks

- DBNs are created by stacking RBNs

- How?
  1. Train an RBM on the data
  2. Train another RBM to model the significant dependencies between the hidden units of the RBM in step 1
  3. Repeat step 2 for additional layers
  4. Replace undirected connections of lower level RBMs by top-down, directed connections
  5. Add softmax output layer to create pretrained DBN-DNN
Deep Belief Networks
Interfacing a DNN with a HMM

• After the fine-tuning, the DNN will have outputs of the form $p(\text{HMMstate} | \text{AcousticInput})$

• For Viterbi/Forward-Backward algorithm, the probability $p(\text{AcousticInput} | \text{HMMstate})$ is needed

• Through Bayes rule, $p(\text{AcousticInput} | \text{HMMstate})$ is obtained as $p(\text{HMMstate} | \text{AcousticInput}) \times p(\text{AcousticInput}) / p(\text{HMMstate})$

• $p(\text{AcousticInput})$ is unknown
  • All likelihoods scaled by same factor -> no effect on alignment

• Conversion important when unbalanced training data
Small Vocabulary Results (TIMIT)

TIMIT is a small acoustic-phonetic continuous speech corpus. It is often used in benchmarking speech recognition methods. However, good performance with TIMIT does not guarantee good performance with larger corpora.

Observations:

- Triphone GMM-HMM outperformed monophone DBN-DNNs (with MFCC)
- Monophone DBN-DNNs on filterbanks outperformed triphone GMM-HMM
Large Vocabulary Results

- DNN-HMM systems tested on five Large Vocabulary tasks
  - Outperformed GMM-HMMs in every task

- Bing-Voice-Search Speech Recognition Task
  - Trained on 24 hrs of speech data with different variations
  - 5 hidden layers of size 2048
  - Window of 11 frames used to classify middle frame into corresponding HMM state
  - Tri-phone states
  - Sentence accuracy of 69%, compared to 63.8% for the GMM-HMM system
Large Vocabulary Results

- **Switchboard Speech Recognition Task**
  - Dataset over 300h of speech data, test set 6.3h
  - Same system as in Bing, but with 7 hidden layers

- **Google Voice Input Speech Recognition Task**
  - 5870h of data
  - 4 hidden layers, 2560 hidden units per layer
  - Window of 11 frames, 40 log-filterbank features for each frame
  - DNN sparsified -> weights below certain threshold set to zero
Large Vocabulary Results

• **Youtube Speech Recognition Task**
  • Goal is to transcribe Youtube data
  • No strong language model --> Strong acoustic model essential
  • 1400hrs of data
  • 17552 context-dependent tri-phone states in HMM
  • Only 4 hidden layers, to save computation resources for softmax layer

• **English Broadcast News Speech Recognition Task**
  • 50 hrs of data
  • 6 hidden layers, 1024 units each
  • Window of 9 frames
  • 2220 tri-phone HMM states
## Large Vocabulary Results

A comparison of the percentage WERs using DNN-HMMs and GMM-HMMs on five different large vocabulary tasks.

<table>
<thead>
<tr>
<th>Task</th>
<th>Hours of training data</th>
<th>DNN-HMM</th>
<th>GMM-HMM</th>
<th>GMM-HMM using larger training data</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWITCHBOARD (TEST SET 1)</td>
<td>309</td>
<td>18.5</td>
<td>27.4</td>
<td>18.6 (2000h)</td>
</tr>
<tr>
<td>SWITCHBOARD (TEST SET 2)</td>
<td>309</td>
<td>16.1</td>
<td>23.6</td>
<td>17.1 (2000h)</td>
</tr>
<tr>
<td>ENGLISH BROADCAST NEWS</td>
<td>50</td>
<td>17.5</td>
<td>18.8</td>
<td></td>
</tr>
<tr>
<td>BING VOICE SEARCH (sentence error rates)</td>
<td>24</td>
<td>30.4</td>
<td>36.2</td>
<td></td>
</tr>
<tr>
<td>GOOGLE VOICE INPUT</td>
<td>5870</td>
<td>12.3</td>
<td>36.2</td>
<td>16.0 (&gt;&gt; 5870h)</td>
</tr>
<tr>
<td>YouTube</td>
<td>1400</td>
<td>47.6</td>
<td>52.3</td>
<td></td>
</tr>
</tbody>
</table>

Aalto-yliopisto
Homework

1. Why is the GMM-model inefficient for modeling speech data?

2. Why do log Mel-scale filter-bank outputs work better than MFCC vectors as inputs for DNN?
References


Questions?