HMM Overview

Jeff Berry

October 20, 2008
Deterministic Processes

- Each state is entirely dependent on the previous state.
- Green will always follow red, red will always follow yellow, yellow will always follow green.
Markov Chains

- A Markov Chain models a process that depends on the previous $n$ states
- We will assume the weather can be predicted based on the weather of the previous day
- Each transition has a probabilistic weight – the prediction is probabilistic rather than deterministic
Markov Chains

- A state transition matrix shows all the transition probabilities for the weather

<table>
<thead>
<tr>
<th></th>
<th>sun</th>
<th>cloud</th>
<th>rain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>sun</strong></td>
<td>0.50</td>
<td>0.375</td>
<td>0.125</td>
</tr>
<tr>
<td><strong>Yesterday</strong> cloud</td>
<td>0.25</td>
<td>0.125</td>
<td>0.625</td>
</tr>
<tr>
<td><strong>rain</strong></td>
<td>0.25</td>
<td>0.375</td>
<td>0.375</td>
</tr>
</tbody>
</table>
To initialize the system, we need to know the probability of the weather at the beginning. This is represented by the $\pi$ vector:

<table>
<thead>
<tr>
<th></th>
<th>sun</th>
<th>cloud</th>
<th>rain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Now we have a definition of a Markov chain:

1. **States** – sun, cloud, rain
2. **\(\pi\) vector** – initial probabilities
3. **Transition matrix** \(P\) – the probability of the weather given the previous day’s weather
Hidden Processes

What if we don’t have access to the sequence of states, but only to a sequence of events that are probabilistically related to the weather?

Suppose we want to know what the weather was like last year during October, but we don’t have access to the weather records. Instead we have daily sales records of ice cream from a local shop.

The amount of ice cream sold is probabilistically related to the weather as shown in the emission matrix.
Ice Cream

<table>
<thead>
<tr>
<th></th>
<th>$1k</th>
<th>$2k</th>
<th>$3k</th>
</tr>
</thead>
<tbody>
<tr>
<td>sun</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>cloud</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>rain</td>
<td>0.6</td>
<td>0.3</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Weather

HMM Overview
Definition of HMM

- Hidden States
- Sequence of Observations
- Transition Matrix
- $\pi$ Vector
- Emission Matrix
HMMs for Speech Recognition

- **Underlying States** – phonemic, sub phonemic, or multiphone units
- **Observed** – Acoustic signal
  1. Gaussian acoustic models, or
  2. Code indices from vector quantization

![Diagram of HMMs for Speech Recognition]

- Cepstral Feature Extraction
DSP

- A common DSP technique for ASR is Mel-Frequency Cepstral Coefficient (MFCC)
- MFCC returns a feature vector $y_t$
Since MFCC returns vectors that are continuously valued, we have to somehow categorize them.

One approach is called **Vector Quantization**, where we try to match the vector to a predetermined standard, i.e. a codebook.

With VQ, the observed sequence emitted by the HMM is a sequence of codebook indices.
Instead of VQ, we can assume that the values of the MFCC vector are normally distributed.

Using multivariate Gaussians, we can compute the likelihood of any given observation.

The variance of each acoustic dimension is considered separately.

\[ \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} ; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} ; \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix} \]
In real speech a single Gaussian is not sufficient.

So we use a mixture – **Gaussian Mixture Model** (GMM).
Summary... so far

- An ASR system is made up of a bunch of HMMs – one for each phoneme (or tri-phone, syllable, etc.)
- Each of these HMMs has a set of underlying states
- Each state in the HMM emits a likelihood function (GMM) for what the values of the MFCC feature vector will be for that state
- The ASR system has to choose which HMM is the best fit based on the sequence of observed MFCC feature vectors

Questions

- Where do the numbers in the HMMs come from?
- How does the system decide which HMM is the best fit?
Where do the numbers in the HMMs come from? – Training

- To train an HMM, we specify how many states and how many mixtures will be in the GMMs.
- Then we take a bunch of pre-labeled sound files of the word, tri-phone, etc. that we want to model and run a training algorithm on them.
- Training algorithms, such as Baum-Welch calculate the probabilities of the transition and emission matrices of the HMM, and the means and variances of the GMMs based on the training data.
How does the system decide which HMM is the best fit?

Viterbi Algorithm

- Calculates the probability of the observed sequence for the HMM
- The algorithm is run on every HMM to see which one has the highest probability of emitting the observed sequence
- Use of an N-gram grammar or Kohonen neural network can improve performance by weighting more likely HMMs higher
Overview of ASR

Now for a demo...