02457 Non-Linear Signal Processing, Mini project in speech processing, part I of III

This exercise is based on L.R. Rabiner *A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of the IEEE **77**, 257-286, (1989).

Print and comment on the figures produced by the software as outlined below at the **Checkpoints**, this exercise is split in three parts, hence, performed over three Thursday sessions.

Speech recognition

Consider a system where the task is to recognize a single spoken word, generally referred to as isolated word recognition. Thus, assume we have a vocabulary of R words to be recognized and that each word is to be modeled by a distinct sequence model. This involves the following steps:

- Feature Extraction: A frame based spectral and/or temporal analysis of the speech signals is performed to give observation vectors, $\mathbf{y}_t$, which can be used to train the sequence models.
- Symbol identification: Use a training set of L occurrences of each spoken word, i.e., $L \times R$ sequences, to derive a codebook containing K possible observation (feature) vectors using vector quantization methods. Subsequently, any observation vector used for either training or recognition is quantized using this codebook.
- Training of the sequence models based on a training set.
- Recognition using the sequence models.

Feature Extraction

The feature vector sequence $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$ is obtained from front-end spectral analysis of the speech samples. Many features have been used as observations, but most prevalent are the LP parameters, cepstral parameters, and related quantities.

First, the sampled speech signal is blocked into frames of N samples, for example $N = 320$ corresponding to 32 ms at 10 kHz sampling frequency. Consecutive frames are spaced ΔN samples apart, e.g., $\Delta N = 80$ samples, such that the analysis frame end-times are $m = \{320, 400, \ldots, N_s\}$ which become observation times $t = \{1, 2, \ldots, T\}$. For a typical word utterance lasting 1 sec, the signal length $N_s = 10000$ samples and the number of frames are $T = 122$.

Each frame is multiplied by an N-sample window (Hamming), and the autocorrelation function is found to lag M, where M is the order of the desired LPC analysis (we use $M = 12$). The so-called LPC coefficients are computed by using the Levinson-Durbin recursion, and then converted to $Q = 12$ cepstral coefficients. To add dynamic information, Q temporal difference cepstral coefficients are computed and concatenated to form a $2Q$ dimensional vector representing each frame.

Checkpoint 10.1

Use the matlab script `main10a.m` to perform feature extraction on a speech signal using the above described procedure. The script produces two figures. Figure 1 depicts the features derived from windows of three letters s,o,f, each letter has two instances that we here refer to as the training and test cases. Locate the part of the images that "belong" to the three letters. Comment on the similarity and differences between the letter features and the relation between training and test sets.

The second figure shows a scatterplot of data projected on the two most variant directions in feature space. Comment on the separation of the feature vectors for the three letters s,o,f.

Symbol identification

Since we want to use a sequence model with a discrete observation symbol density, rather than the continuous feature vectors above, a clustering algorithm is used to derive a "codebook" containing K possible observation values. Thus, for each occurrence of a word, the feature extraction creates a sequence $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T\}$ of observation vectors (one for each frame), which are quantized into one of the permissible set resulting in the scalar, discrete observation sequence $\{y_1, y_2, \ldots, y_T\}$, where each of the variables y_t may take only integer values k (codebook index) in the range $1 \leq k \leq K$. When we finally assemble a speech recognizer (part III of the mini project) we will use the K-means algorithm to form the codebook.

Training simple sequence model

Given discrete observation sequences from the words in the vocabulary, we next would like to estimate sequence models for each word, i.e., compute the probabilities $P(\{y_n\}|\text{word})$. Using the Bayes rule

$$P(\text{word}|\{y_n\}) = \frac{P(\{y_n\}|\text{word})P(\text{word})}{P(\{y_n\})} \tag{1}$$

we can compute the posterior probability of each word given a sequence.

Here we will use a simple Markov chain model, in the second part of the exercise (next Thursday) we will generalize this to hidden Markov models. Let y_n be a sequence of N symbols with K states. Let $a_{j,j'}$ be the probability of jumping from j to j'. To ensure that we always jump to some state, the matrix $a_{j,j'}$ must satisfy $\sum_{j'} a_{j,j'} = 1$. a can be estimated by maximum likelihood:

$$\begin{aligned} P(\{y_n\}|a) &= P(y_1)\prod_{n=2}^{N} P(y_n|y_{n-1}, a) \\ &= P(y_1)\prod_{j,j'} (a_{j,j'})^{\mathrm{n_{j,j'}}} \end{aligned}$$

where $n_{j,j'}$ is the occurrence of the transition, and $P(y_1)$ is the probability of starting in state y_1. Since we only have one sequence for training we will let this be estimated as $P(y_1) = 1/K$. Using softmax to ensure the normalization condition $\sum_{j'} a_{j,j'} = 1$ we obtain the solution,

$$\widehat{a}_{j,j'} = \frac{n_{j,j'}}{\sum_{j'} n_{j,j'}}$$

Checkpoint 10.2

Use the matlab script `main10b.m` to create a random transition matrix. This matrix is used as a "teacher" that can create training and test sequences. First, we find the "stationary distribution" of symbols, this is the probability distribution that satisfies,

$$P^*(j') = \sum_{j=1} P^*(j) a_{j,j'}.$$

Explain how the program estimates the stationary distribution and explain it's significance.

Use the transition matrix to create increasing length sequences, and observe how the histogram of the observed sequences converge to the stationary distribution. Explain the function `getint.m`.

Verify the maximum likelihood estimate of the transition matrix. Use the matlab script (`main10b.m`) to generate increasing length sequences. Train a "student" transition matrix on these sequences, and show that the error of the student matrix converge to zero, hence, the student matrix converge to the teacher matrix for large training sets.

Peter S. K. Hansen and Lars Kai Hansen, November, 2001, 2006.