02457 Non-Linear Signal Processing, Exercise 11: Speech and sequence processing

This exercise is based on Christopher Bishop: *Machine Learning and Pattern Recognition*, section 13.1.

Print and comment on the figures produced by the software as outlined below at the **Checkpoints**.

Speech recognition

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

- Feature Extraction: A frame based spectral 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., a total of $L \times R$ sequences, to derive a 'codebook' containing K possible types of observations ('cluster centers') using vector quantization methods, for example Kmeans. Subsequently, any observation vector used for either training or recognition is represented by the cluster index, i.e., quantized, using this codebook.
- Training of R sequence models based on the training set.
- Recognition using these 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 so-called *cepstral* parameters.

First, the sampled speech signal is blocked into frames of N samples, for example $N = 320$ corresponding to 32 ms at a 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 $N_s = 10000$ is samples and the number of frames thus about $T = 125$.

Each frame is multiplied by an N-sample window (Hamming), and we estimate Q the *cepstral coefficients* by low-pass filtering of the signal's log-powerspectrum. Let the windowed signal be denoted by $s_w(n)$ and let the Fourier representation of the signal be $S_w(k) = \sum_{n=1}^{N} s_w(n) \exp(2\pi i \frac{nk}{N})$. The cepstral features are then computed as

$$C_w(m) = \sum_{k=1}^{N} \log(|S_w(k)|) \exp(-2\pi i \frac{km}{N})/N. \quad (1)$$

To capture the spectral envelope we use features $\mathbf{y}_m = C_w(m), m = 2 : (Q + 1)$. To further add dynamic information, Q temporal difference cepstral coefficients (neighbor window differences) are computed and concatenated with $\mathbf{y}$ to form a $2Q$ dimensional vector representing each frame.

Checkpoint 11.1

Use the matlab script main11a.m to perform feature extraction on a speech signal using the above described procedure. The script produces four figures. A figure shows the time functions for letters s,o,f. A figure shows the spectrograms (short term time frequency analysis using the Fourier transform).

A figure shows the temporal development of the ceptral coefficients. Each letter has two instances that we here refer to as the training and test cases. Locate the parts of the images that 'belong' to each of the three letters. Comment on the similarity and differences between the letter features and the relation between training and test sets.

A final figure shows a scatter plot of data projected on the two most variant directions in feature space (principal component analysis). Comment on the clustering of the feature vectors for the three letters s,o,f.

Symbol identification

In speech recognition sequence models are based on discrete observation symbols, rather than the continuous feature vectors above. Therefore we apply a clustering algorithm to derive a *codebook* containing K values to quantize the short time features. 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 (exercise 12) we will use the K-means algorithm to form the codebook. Here we will simply assume that each window now is represented by a single number.

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 Bayes' rule

$$P(\text{word}|\{y_n\}) = \frac{P(\{y_n\}|\text{word})P(\text{word})}{P(\{y_n\})} \tag{2}$$

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

Here we will investigate a simple Markov chain model, in exercise 12 (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 11.2

Use the matlab script main11b.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 which is the probability distribution that satisfies,

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

Explain the role of the stationary distribution and explain how the program estimates the stationary distribution.

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 (main11b.m) to generate increasing length sequences. Train a *student* transition matrix on these sequences and show that the relative 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, 2007, 2012.