02457 Non-Linear Signal Processing, Exercise 12

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). Additional background can be found in C. Bishop: 'Pattern Recognition and Machine Learning', chapter 13.

We will investigate the Hidden Markov Model (HMM) for sequence modeling and based on the HMM we will construct a simple speech recognition pipeline.

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

Training a sequence model

First, let us define the elements of a discrete observation based HMM, $\mathcal{M} = \{S, \pi(1), \mathbf{A}, \mathbf{B}, \{\mathbf{y}_k, 1 \leq k \leq K\}\}$, characterized by the state equation and observation equation, i.e.,

$$\begin{aligned} \pi(t) &= \mathbf{A}\pi(t-1), & \pi(t) &= [P(x_t = 1), \cdots, P(x_t = S)]^T \\ \mathbf{p}(t) &= \mathbf{B}\pi(t), & \mathbf{p}(t) &= [P(y_t = 1), \cdots, P(y_t = K)]^T \end{aligned}$$

Here, S is the number of states in the model, x_t is the state random process, $\pi(t)$ is the state probability vector at time t, and $\mathbf{A}$ is the S-by-S state transition matrix with elements $a_{ij} = a(i|j) = P(x_t = i|x_{t-1} = j)$ (note: This code uses a transposed notation for $\mathbf{A}$ compared exercise 11, Bishop, and Rabiner). In the observation equation, K is the number of distinct observation symbols per state (the number of vectors $\mathbf{y}_k$ in the KMeans based vector quantization (VQ) codebook), y_t is the observation random process, $\mathbf{p}(t)$ is the observation probability vector at time t, and $\mathbf{B}$ is the K-by-S observation probability matrix with elements $b_{ki} = b(k|i) = P(y_t = k|x_t = i)$.

Hence an HMM requires specification of the two model parameters, S and K, and specification of the three probability tables $\mathbf{A}$, $\mathbf{B}$, and $\pi(1)$. Thus, the training problem is to adjust the model parameters to maximize the likelihood $P(y|\mathcal{M})$, i.e., the probability of the observation sequence $y = \{y_1, y_2, \ldots, y_T\}$, given the model.

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_T\}$ denote a specific state sequence, then the likelihood $P(y|\mathcal{M})$ can be found as

$$\begin{aligned} P(y|\mathcal{M}) &= \sum_{\mathcal{I}} P(y, \mathcal{I}|\mathcal{M}) = \sum_{\mathcal{I}} P(y|\mathcal{I}, \mathcal{M}) P(\mathcal{I}|\mathcal{M}) \\ &= \sum_{\mathcal{I}} b(y_1|i_1) b(y_2|i_2) \cdots b(y_T|i_T) \times P(x_1 = i_1) a(i_2|i_1) a(i_3|i_2) \cdots a(i_T|i_{T-1}) \end{aligned}$$

With huge savings compares to a direct method, computation of $P(y|\mathcal{M})$ is based on the

following forward/backward recursions (dynamic programming),

$$\begin{aligned}
\alpha(y_1^t, i) &= \sum_{j=1}^{S} \alpha(y_1^{t-1}, j) a(i|j) b(y_t|i), \qquad \alpha(y_1^1, i) = P(x_1 = i) b(y_1|i) \\
\beta(y_{t+1}^T|i) &= \sum_{j=1}^{S} \beta(y_{t+2}^T|j) a(j|i) b(y_{t+1}|j), \qquad \beta(y_{T+1}^T|i) = 1
\end{aligned}$$

where $\alpha(y_1^t, i)$ is the joint probability of having generated the partial forward sequence y_1^t and having arrived at state i at the t'th step. Similarly, $\beta(y_{t+1}^T|i)$ denote the probability of generating the backward partial sequence y_{t+1}^T, given that the state sequence emerges from state i at time t. To avoid numerical problems (underflow) in the computations of the forward/backward recursions, we must scale $\alpha(\cdot,\cdot)$ and $\beta(\cdot|\cdot)$ in each step with c_t, to obtain $\hat{\alpha}(\cdot,\cdot)$ and $\hat{\beta}(\cdot|\cdot)$.

Based on the scaled forward/backward recursions, an iterative estimation procedure for estimating a Hidden Markov Model, $\mathcal{M}$, corresponding to a local maximum of the likelihood $P(y|\mathcal{M})$, can be devised. This algorithm, known as F-B reestimation, takes a model $\mathcal{M} = \{S, \pi(1), \mathbf{A}, \mathbf{B}, \{\mathbf{y}_k, 1 \le k \le K\}\}$ and the training observation, $y = y_1^T$, to compute a new model $\bar{\mathcal{M}} = \{S, \bar{\pi}(1), \bar{\mathbf{A}}, \bar{\mathbf{B}}, \{\mathbf{y}_k, 1 \le k \le K\}\}$ as

$$\begin{aligned}
\bar{a}(j|i) &= \frac{\sum_{t=1}^{T-1} \hat{\alpha}(y_1^t, i) a(j|i) b(y_{t+1}|j) \hat{\beta}(y_{t+2}^T|j)}{\sum_{t=1}^{T-1} \hat{\alpha}(y_1^t, i) \hat{\beta}(y_{t+1}^T|i)} \\
\bar{b}(k|j) &= \frac{\sum_{y_t=k, t=1}^{T} \hat{\alpha}(y_1^t, j) \hat{\beta}(y_{t+1}^T|j)}{\sum_{t=1}^{T} \hat{\alpha}(y_1^t, j) \hat{\beta}(y_{t+1}^T|j)} \\
\bar{\pi}_i(1) &= P(x_1 = i) = \hat{\alpha}(y_1^1, i) \hat{\beta}(y_2^T|i).
\end{aligned}$$

If the likelihood has increased such that $P(y|\bar{\mathcal{M}}) - P(y|\mathcal{M}) \ge \epsilon$, where ϵ is a given tolerance, then re-estimate the model with $\mathcal{M} = \bar{\mathcal{M}}$. The required likelihood can be obtained from any time slot in the lattice, i.e.,

$$P(y|\mathcal{M}) = \sum_{i=1}^{S} \alpha(y_1^t, i) \beta(y_{t+1}^T|i) = \sum_{i=1}^{S} \alpha(y_1^T, i) = \left(\prod_{\tau=1}^{T} c_\tau \right)^{-1}$$

To avoid numerical problems ($P(y|\mathcal{M})$ is small), the logarithm of likelihood measure is reported

$$\log P(y|\mathcal{M}) = -\sum_{\tau=1}^{T} \log c_\tau.$$

The final thing to consider is that we typical have a training set of utterances L for a given word HMM. Let $y^{(l)}$ denote the lth observation of length T_l, and let superscript l indicate results for this sequence, then the F-B re-estimation algorithm is modified as

$$\begin{aligned}
\bar{a}(j|i) &= \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\hat{\alpha}^{(l)}(y_1^t,i)a(j|i)b(y_{t+1}|j)\hat{\beta}^{(l)}(y_{t+2}^{T_l}|j)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l-1}\hat{\alpha}^{(l)}(y_1^t,i)\hat{\beta}^{(l)}(y_{t+1}^{T_l}|i)} \\
\bar{b}(k|j) &= \frac{\sum_{l=1}^{L}\sum_{y_t=k,t=1}^{T_l}\hat{\alpha}^{(l)}(y_1^t,j)\hat{\beta}^{(l)}(y_{t+1}^{T_l}|j)/c_t^{(l)}}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\hat{\alpha}^{(l)}(y_1^t,j)\hat{\beta}^{(l)}(y_{t+1}^{T_l}|j)} \\
\bar{\pi}_i(1) &= P(x_1=i) = \frac{1}{L}\sum_{l=1}^{L}\hat{\alpha}^{(l)}(y_1^1,i)\hat{\beta}^{(l)}(y_2^T|i).
\end{aligned}$$

Checkpoint 12.1

Inspect the update rules for HMM training. Show that they are multiplicative in $\mathbf{A}$, i.e., that the updated value $\bar{\mathbf{A}}$ is proportional to the 'old' value $\mathbf{A}$. This means that if an entry in transition matrix is initialized to value zero it will remain zero after the update.

We consider a simulation based on a given HMM 'teacher', a model with a specific set of parameters as set up in the matlab script main12a.m. Inspect the matrices and deduce which type of sequences you expect to be generated by the teacher HMM. How many states and emissions are present in the teacher HMM? Is the EM algorithm trained HMM reproducible, i.e., consistent in determining the number of states? How many parameters are estimated in the HMM for a given number of hidden states?

We want to generate bias-variance trade off curves, i.e. training and test log-likelihoods as a function of the number of hidden states in the student HMM. This experiment will reveal if we can 'over-train' the HMM when too many hidden states are used. Explain the bias-variance trade off for these models. Can the transition matrices and emission probabilities be directly compared? (Hint: permutation symmetry of the hidden state index). What is the overall typical number of states to recommend?

Try to halt the matlab script and investigate the estimated transition matrix, e.g., when the no. of states is 4 (insert a break point).

Speech recognition

In the second part of the exercise we will now combine the speech representation from Exercise 11 and the HMM to construct a simple speech recognizer for isolated words. Assume we have a vocabulary of R words to be recognized and that each word is to be modeled by a distinct sequence model. Let us remind ourselves that this involves the following steps:

- Feature Extraction: A frame based cepstral 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.
- Word 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 cepstral analysis of the speech samples. As in Exercise 11, 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 $t = \{1, 2, \ldots, T\}$. The resulting $2Q$-dimensional feature vector represents the spectral envelope of the signal in the window, and the rate of temporal change of this envelope. We use an approach like described in Rabiner Section VI to estimate cepstral vectors using filters.

Symbol identification

Since we want to use a sequence model with a discrete observation symbol density, a clustering algorithm is used to reduce the real valued cepstral vectors to a single number with reference to a 'codebook' containing K cluster centers. 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 then quantized to a discrete observation sequence $\{y_1, y_2, \ldots, y_T\}$, where each of the variables y_t may take only integer values k in the range $1 \leq k \leq K$. Here we use the K-means algorithm to form the codebook.

Checkpoint 12.2

The speech recognition experiment concerns recognition of spoken digits. The data file is `ti46.mat` which contains a list of further data files. The total amount of data is quite large (3 MB). We use part of the data set for training, the rest can be used for testing.

Use the matlab script main12b.m to train a codebook and a set of ten HMM's - one HMM for each word (i.e., the digits 0,...,9). Figure 2 shows the likelihood for a couple of the training sessions are they increasing as expected? Figure 3 shows the transition matrix (A) and the emission probability matrix (B) for two of the HMM-models. What is the number of hidden states and what is the size of the codebook in this case? Finally, we examine the performance on two test signals (Figure 4-5). What is the probability of the 'runner up' in the two cases? Can you find a test case where the system fails?

Peter S. K. Hansen and Lars Kai Hansen,
Rasmus Elsborg Madsen and Ole Winther, November 2001-2012.