Hidden Markove Model for Speech Recognition - Part 2

Discrete Observation Probability Modeling and Segmental k-means Learning Algorithms

ELEC747 Speech Signal Processing

Kyungpook National University

Gil-Jin Jang

`gjang@knu.ac.kr`

May 15, 2018

Recap: HMM for Speech Recognition

INPUT feature matrix (usually MFCC) of size $D \times T$,

$$\mathbf{O} = [\mathbf{o}_1 \ \mathbf{o}_2 \ \ldots \ \mathbf{o}_T] \quad ,$$

where D is feature dimension and T the number of frames.

MODEL Set of models $\{\lambda_1, \lambda_2, \ \ldots, \ \lambda_L\}$, where L is the number of models. Usually word or phoneme models.

OUTPUT According to Bayesian decision rule, find a model with largest *a posteriori* probability.

$$\begin{aligned} \lambda^* &= \arg\max_{\lambda} P(\lambda|\mathbf{O}) = \arg\max_{\lambda} \frac{P(\mathbf{O}|\lambda)P(\lambda)}{P(\mathbf{O})} \\ &= \arg\max_{\lambda} P(\mathbf{O}|\lambda)P(\lambda) \end{aligned}$$

When $P(\lambda_l)$ are all equal for $\forall l$, it becomes a *maximum likelihood* decision rule.

Recap: Elements of HMM

STATES

1. N: a positive integer, *number of states*.
2. $x \in \{1, \ldots, N\}$, *state index*.
3. $\mathbf{x} = (x_1, x_2, \ldots, x_T)$, *state sequence for input length T*.

MODEL $\lambda = \{\pi, \mathbf{A}, \{b_i\}\}$ is a set of model parameters, where

1. $\pi = \{\pi_i\} = \{\Pr(x_1 = i)\}$, for $1 \leq i \leq N$.
   - Length N vector, *initial-state probabilities*.
   - Constraints: $0 \leq \pi_i \leq 1$, $\sum_{i=1}^{N} \pi_i = 1$
2. $\mathbf{A} = \{a_{ji}\} = \{\Pr(x_t = i | x_{t-1} = j)\}$, for $1 \leq i, j \leq N$.
   - $N \times N$ matrix, *state-transition probabilities*.
   - Constraints: $\sum_{i=1}^{N} a_{ji} = 1$
3. $b_j(\mathbf{o}_t) = P(\mathbf{o}_t | x_t = j)$, for $1 \leq j \leq N$.
   - $b_j : \mathcal{R}^D \rightarrow \mathcal{R}$, *observation/emission probabilities*.

Observation Probabilities Modeling

It is straitforward that we can implement π with a real vector, and $\mathbf{A}$ with a real matrix. However, $b_i(\mathbf{o})$ is a function of probabilities, so there is no single method. We can first think of *discrete* and *continuous* cases.

Discrete
: convert the input vector into a finite set of symbols,

$$o_t = Q(\mathbf{o}_t|\theta_d)$$

where θ_d is a set of discrete conversion parameters, and o_t is a positive integer. Use a relative histogram as a pmf (probabilitiy mass function).

$$b_j(\mathbf{o}_t) = \Pr(o_t|x_t = j)$$

Continuous
: use a (parametric) pdf (probability density function) to approximate the obervation probabilities directly

$$b_j(\mathbf{o}_t) = f(\mathbf{o}_t|x_t = j, \theta_c)$$

where θ_c is a set of parameters for the distribution.

k-means Clustering

- Given a data set $\mathbf{O} = [\mathbf{o}_1 \ \mathbf{o}_2 \ \ldots \ \mathbf{o}_T]$ of T observations in a D-dimensional space, our goal is to partition the data set into K clusters or groups (equivalent to vector space partitioning).
- The vectors μ_k, where $k = 1, \ldots, K$, represent k-th cluster, e.g. the centers of the clusters $\Rightarrow$ k-means.
- The objective function that measures distortion is

$$J = \sum_{t=1}^{T} \sum_{k=1}^{K} I(t,k) ||\mathbf{o}_t - \mu_k||^2$$

  where $I(t,k) \in 0,1$ is a binary indicator function, defined for each data point $\mathbf{o}_t$, where $k = 1, \ldots, K$.
- **1-of-K coding scheme**: $\mathbf{o}_t$ is assigned to cluster k then $I(t,k) = 1$, and $I(t,k) = 0$ for $j \neq k$.
- We ought to find $I(t,k)$ and $\{\mu_k\}$ that minimise J.

k-means Solution

Iterative solution: First we choose some initial values for μ_k.

Step 1: We minimise J with respect to $I(t,k)$, keeping μ_k fixed. J is a linear function of $I(t,k)$, we have a closed form solution

$$I(t,k) = \begin{cases} 1, & \text{if } k = \arg\min_j ||\mathbf{o}_t - \mu_j||^2 \\ 0, & \text{otherwise} \end{cases}$$

Step 2: We minimise J with respect to μ_k, keeping the set $I(t,k)$ fixed. J is the quadratic function of μ_k, so we set its derivative with respect to μ_k to zero.

$$2\sum_{t=1}^{T} I(t,k)(\mathbf{o}_t - \mu_k) = 0 \Rightarrow \mu_k = \frac{\sum_t I(t,k)\mathbf{o}_t}{\sum_t I(t,k)}$$

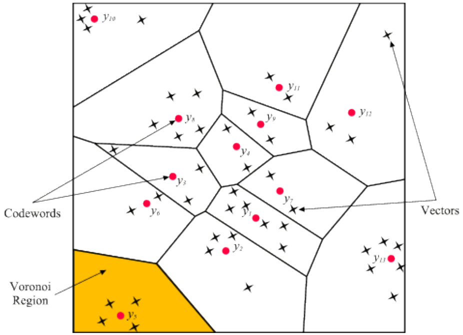
Step 3: Repeat steps 1 and 2 until convergence.

Result of k-means: Voronoi Diagram

The conversion of real vector to a finit set of symbols is, finding the index of the nearest mean vector given the conversion parameter set, $\theta_d = \{\mu_k\}$.

$$o_t = Q(\mathbf{o}_t|\theta_d) = \arg\min_j ||\mathbf{o}_t - \mu_j||^2$$

The resultant partitioning is creating Voronoi diagram in the vector space.



Maximum Likelihood Approximation

The problem of speech recognition is that given the observation matrix $\mathbf{O} = [\mathbf{o}_1 \cdots \mathbf{o}_T]$, find the best model that has the maximum *a posteriori* probability, which is equivalent to a maximum likelihood problem according to Bayes' rule:

$$\lambda^* = \arg\max_{\lambda} P(\lambda|\mathbf{O}) = \arg\max_{\lambda} P(\mathbf{O}|\lambda)\frac{P(\lambda)}{P(\mathbf{O})} = \arg\max_{\lambda} P(\mathbf{O}|\lambda)P(\lambda)\,. \tag{1}$$

If we assume that all the models are equi-probable, $P(\lambda)$ is all equal for any λ, then Equation 1 becomes finding a model with maximum likelihood, $P(\mathbf{O}|\lambda)$.

Viterbi Decoding Algorithm

However, exact computation of $P(\mathbf{O}|\lambda)$ requires summation of the probabilities of all possible state sequences, which is intractable in practical situations. Instead, we approximate the model likelihood by a single, maximum probability:

$$P(\mathbf{O}|\lambda) = \sum_{\text{all}\,\mathbf{X}} P(\mathbf{O}, \mathbf{X}|\lambda) \approx c \max_{\mathbf{X}} P(\mathbf{O}, \mathbf{X}|\lambda)\,, \tag{2}$$

where $\mathbf{X} = \{x_1, \ldots, x_T\}$ is the HMM state sequence, c is a certain constant. It then requires finding a state sequence that has the maximum joint probability of observation sequence and corresponding state sequence as follows:

$$\mathbf{X}^*|_{\mathbf{O},\lambda} = \arg\max_{\mathbf{X}} P(\mathbf{O}, \mathbf{X}|\lambda). \tag{3}$$

The algorithm to solve Equation 3 is called Viterbi decoding.

Viterbi decoding for discrete input

To find $\mathbf{X}^*$, derive Viterbi decoding algorithm in the case of discrete input, $Q(\mathbf{o}_t|\theta_d) = o_t = 1, 2, \ldots,$ clarify notations as follows:

Index Variables and Constants

- Number of observed symbols: $T \in \mathcal{N}$ (natural number);
- Time variable: $1 \leq t \leq T$;
- Number of discrete, different events: $K \in \mathcal{N}$;
- Observation symbol index variable: $1 \leq k \leq K$;
- Number of HMM states: $N \in \mathcal{N}$;
- State index variables: $1 \leq x_t \leq N$, $1 \leq i \leq N$ and $1 \leq j \leq N$;

Observed symbols

- $o_t \in [1\ K]$

Viterbi decoding for discrete input

HMM parameters

- Initial state probabilities: $\pi_i = P(x_1 = i)$ where x_1 is the initial state;
- State transition probabilities denoted by a square matrix $\mathbf{A}$ of size $N \times N$, whose elements are: $a_{ji} = P(x_t = i | x_{t-1} = j)$, where x_t and x_{t-1} are the state at time t and $t-1$, respectively.
- State-wise observation probabilities: computed by the following pmf (probability mass function),

$$b_i(k) = P(o_t = k | x_t = i)$$

Defining the objective function

Define the probability for the given path:

$$\begin{aligned} P(\mathbf{O}, \mathbf{X}|\lambda) &= P(\mathbf{X}|\lambda) \cdot P(\mathbf{O}|\mathbf{X}, \lambda) \\ &= \left\{ \pi_{x_1} \prod_{t=2}^{T} a_{x_{t-1}x_t} \right\} \cdot \left\{ \prod_{t=1}^{T} b_{x_t}(o_t) \right\} \\ &= \pi_{x_1} b_{x_1}(o_1) \prod_{t=2}^{T} a_{x_{t-1}x_t} b_{x_t}(o_t). \end{aligned} \tag{4}$$

The *Viterbi decoding* is an efficient, inductive method to find the optimal state sequence $\mathbf{X}^*$ that maximizes $P(\mathbf{O}, \mathbf{X}|\lambda)$ — joint probability of the (given) observations $\mathbf{O}$ and the path $\mathbf{X}^*$ given an HMM λ.

Inductive derivation of the objective function

Given that for all states i, and for a fixed frame index t, define the maximum probability that the give observation $\{o_1, \ldots, o_t\}$ ends at state i at time t as:

$$\delta_t(i) = \max_{x_1 \ldots x_{t-1}} P[o_1, \ldots, o_t, X_1 = x_1, \ldots, X_{t-1} = x_{t-1}, X_t = i]. \tag{5}$$

Inductive derivation under the first-order Markov assumption:

$$\begin{aligned} & P[o_1, \ldots, o_t, X_1 = x_1, \ldots, X_{t-1} = x_{t-1}, X_t = i] \\ = \quad & P[o_t, X_t = i | \underbrace{o_1, \ldots, o_{t-1}}_{O_t \perp O_{1:t-1}}, \underbrace{x_1, \ldots, x_{t-1}}_{X_t \perp X_{1:t-2}}] \cdot P[o_1, \ldots, o_{t-1}, x_1, \ldots, x_{t-1}] \\ = \quad & P[o_t, X_t = i | \underline{X_{t-1} = x_{t-1}}] \cdot P[o_1, \ldots, o_{t-1}, x_1, \ldots, x_{t-1}] \end{aligned} \tag{6}$$

Inductive derivation of the objective function

$$
\begin{aligned}
\delta_t(i) &= \max_{x_1 \ldots x_{t-1}} P[o_t, X_t = i | \underline{X_{t-1} = x_{t-1}}] \cdot P[o_1, \ldots, o_{t-1}, \underline{x_1, \ldots, x_{t-1}}] \\
&= \max_j P[o_t, X_t = i | X_{t-1} = j] \underbrace{\max_{x_1 \ldots x_{t-2}} P[o_1, \ldots, o_{t-1}, x_1, \ldots, X_{t-1} = j]}_{\triangleq \delta_{t-1}(j)} \\
&= \max_j P[o_t, \underline{X_t = i} | \underline{X_{t-1} = j}] \cdot \delta_{t-1}(j) \\
&= \max_j P[\underbrace{o_t | X_t = i, X_{t-1} = j]}_{O_t \perp X_{t-1}} \cdot \underbrace{P[X_t = i | X_{t-1} = j]}_{\triangleq a_{ji}} \cdot \delta_{t-1}(j) \\
&= \max_j P[o_t | X_t = i] \cdot a_{ji} \cdot \delta_{t-1}(j) \quad = \quad \underbrace{P[o_t | X_t = i]}_{\triangleq b_i(o_t)} \cdot \max_j [a_{ji} \cdot \delta_{t-1}(j)] \\
&= b_i(o_t) \cdot \max_j [a_{ji} \cdot \delta_{t-1}(j)] \qquad (7)
\end{aligned}
$$

Also define the immediate predecessor state index to state i with maximal probability as

$$\psi_t(i) \quad = \quad \arg\max_j [a_{ji} \cdot \delta_{t-1}(j)] \qquad (8)$$

Find the best path

Algorithm: Viterbi

1. Initialization:

$$\delta_1(i) = \pi_i b_i(o_1), \quad \psi_1(i) = \text{null or } 0$$

2. Recurrence for $t = 2, \ldots, T$,

$$\delta_t(i) = b_i(o_t) \max_j \left[a_{ji}\delta_{t-1}(j)\right], \quad \psi_t(i) = \arg\max_j \left[a_{ji}\delta_{t-1}(j)\right]$$

3. Termination at $t = T$,

$$P(\mathbf{O}, \mathbf{X}^* | \lambda) = \max_i \delta_T(i)$$

4. Finding the optimal path by backtracking: decrementing t,

$$\begin{array}{rclrcl} x_T^* & = & \arg\max_i \delta_T(i), & x_{T-1}^* & = & \psi_T(x_T^*) \\ x_{T-2}^* & = & \psi_{T-1}(x_{T-1}^*), & \cdots & & \\ x_1^* & = & \psi_2(x_2^*), & \mathbf{X}^* & = & [x_1^* \ x_2^* \ \ldots \ x_T^*] \end{array}$$

Practical implementation of Viterbi algorithm

- The probability in Equation 4 is composed of multiplications as many as $2T$. Multiplying probabilities, less than 1, huge number of times may cause underflows — approaches 0. So the direct implementation of multiplicative Viterbi decoding may easily fail in searching the best path in case of long observation sequences because most of $\delta_i(o_t)$ may become 0 and indistinguishable.
- In practical situations, logarithm operation which is monotonically increasing is quite useful, because there are only multiplications in Equation 4 which are converted to additions, and prevents underflows.

Logarithmic Viterbi Algorithm Derivation

$$\log P(\mathbf{O}, \mathbf{X}|\lambda) = \log \pi_{x_1} + \log b_{x_1}(o_1) + \sum_{t=2}^{T} \left\{ \log a_{x_{t-1}x_t} + \log b_{x_t}(o_t) \right\}. \quad (9)$$

In the same way,

$$\log \delta_t(i) \quad = \quad \log b_i(o_t) + \max_j [\log a_{ji} + \log \delta_{t-1}(j)] \qquad (10)$$

One thing we have to be careful of is preventing $\log 0 = -\inf$ that is intractable with computer implementation. One method that can avoid it is replacing $\log x$ with $\log \max\{\varepsilon, x\}$ ($\varepsilon = 10^{-10} \sim 10^{-6}$ is a typical choice).
In addition, the Viterbi algorithm can be viewed as a matrix-filling problem with index $t = 1, \ldots, T$. So it can be rewritten in a very efficient manner as follows:

Algorithm: Logarithmic Viterbi by Matrix-Filling

1. Initialization:

   $\Delta := T \times N, \quad \Psi := T \times N, \quad \Pi := N \times 1$

   $B(o,i) := \log \max\{\varepsilon, b_i(o)\}, \quad A(j,i) := \log \max\{\varepsilon, a_{ji}\}$

2. First sample case:

   Define X_0 as a set of indexes of states that the first observation can occur, then

```
for $i = 1 : N$,
    $\Pi(i) = \log(1/|X_0|)$ if $i \in X_0$, $\log\varepsilon$ otherwise;
    $\Delta(1,i) = \Pi(i) + B(o_1, i)$
    $\Psi(1,i) = 0$
end
```

Algorithm: Logarithmic Viterbi by Matrix-Filling

3. Filling object matrices in a sequential order:

```
for $t = 2 : T$,
    for $i = 1 : N$,
        $\Delta(t,i) = B(o_t, i) + \max_j \left[A(j,i) + \Delta(t-1, j)\right]$
        $\Psi(t,i) = \arg\max_j \left[A(j,i) + \Delta(t-1, j)\right]$
    end
end
```

4. Termination:

$$\log P(\mathbf{O}, \mathbf{X}^* | \lambda) = \max_i \Delta(T, i) \;, \qquad x_T^* = \arg\max_i \Delta(T, i)$$

5. Backtracking

```
for $t = T-1 : -1 : 1$,
    $x_t^* = \Psi(t+1, x_{t+1}^*)$
end
```

HMM Learning Derivation Using Viterbi Decoding

So far, we have assumed that all the HMM parameters are available and fixed, and derived Viterbi decoding. However, the problem we may encounter more often is how to obtain the optimal HMM parameters. Segmental k-means is one of the popular method to train an HMM.

Learning problem -

Input: Training data, $\mathbf{O}_m = o_{t,m}$, for $m = 1, \ldots, M$, where M is the number of training data.

Output: HMM $\lambda = \{\mathbf{A}, \mathbf{B}, \Pi\}$, where Π is a vector of initial transition probabilities, $\mathbf{A}$ is a transition probability matrix, and $\mathbf{B}$ is observation probability parameters. $\mathbf{B}$ is a numerical matrix of probability mass function (pmf) in the case of discrete symbol inputs, and function matrix of probability density function (pdf) for real inputs (elements are pdfs).

Segmental k-means training for HMM

1. *Viterbi decoding:* For training sample $\mathbf{O}_m$, perform Viterbi decoding and obtain the optimal state sequence $\mathbf{X}_m^* = (x_{1,m}^*, \ldots, x_{T,m}^*)$.
2. *State membership:* Define a 3-input arguments indicator function $I(t,i,m) \in \{0,1\}$, where $t \in \{1,\ldots,T\}$ is a time index, $i \in \{1,\ldots,N\}$ is a state index, and $m \in \{1,\ldots,M\}$ is a training data sample index.
$$I(t,i,m) = \begin{cases} 1 & \text{if } x_{t,m}^* = i \\ 0 & \text{otherwise} \end{cases}$$
3. *Update initial state probabilities:*
$$\hat{\Pi}(i) = \frac{\sum_{m=0}^{M} I(1,i,m)}{\sum_{j=0}^{N}\sum_{m=0}^{M} I(1,j,m)}$$

Segmental k-means (continued)

4. *Update state transition probabilities:* (Number of transitions from state i to j) / (Number of transitions from state i to all states)

$$\hat{A}(j,i) = \frac{\sum_{m=0}^{M}\sum_{t=2}^{T} I(t-1,j,m)I(t,i,m)}{\sum_{m=0}^{M}\sum_{t=2}^{T} I(t-1,j,m)}, \quad 1 \le i,j \le N$$

5. *Update discrete observation probabilities:* use relative frequencies (number of k occurrences in state i) / (number of all occurrences in state i)

$$\hat{B}(k,i) = \frac{\sum_{m=0}^{M}\sum_{t=1}^{T} I(o_t = k)I(t,i,m)}{\sum_{m=0}^{M}\sum_{t=1}^{T} I(t,i,m)}$$

6. Repeat the above steps until convergence.

(Q: how to obtain the initial parameters for λ?)