CMPT 825: Natural Language Processing **Spring 2008**

Lecture 1 — Jan 23, 2008

Lecturer: Anoop Sarkar *Scribe: Ajeet Grewal*

1.1 Markov Processes

Consider a set of states $S_1, S_2, \ldots S_N$. A discrete Markov process is one in which the system is in a particular state at any given time. The state can be changed only at discrete intervals of time. We denote the time instants associated with state changes with as $t = 1, 2, \ldots$ and we denote the actual state at time t as q_t. The current state in an n-order Markov process is assumed to depend on the previous n states. In particular, in a first order Markov process

$$P(q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \ldots) = P(q_t = S_j | q_{t-1} = S_i)$$

This is called the *Markov Assumption*. In addition, we also make the *Stationary Distribution Assumption*. This means that the transition probabilities don't change over time.

$$P(X_t = S_j | X_{t-1} = S_i) = P(X_{t+l} = S_j | X_{t+l-1} = S_i)$$

1.2 Hidden Markov Models

Usually, we have a sequence of observations $O = (O_1, O_2, \ldots O_T)$ and we want to build a model that best describes this sequence. In this case, we don't know what the states are at each time instant. These are called *Hidden Markov Models* (HMMs) because the states are hidden.

An HMM is defined by the following,

1. The number of states N
2. The number of observation symbols per state M.

3. The state transition probability distribution $A = a_{ij}$ where

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i), 1 \leq i, j \leq N$$

4. The observation probability distribution at state j $B = b_j(k)$ where

$$b_j(k) = P(v_k \text{ at t } | q_t = S_j), 1 \leq j \leq N, 1 \leq k \leq M$$

5. The initial state probability distribution $\pi = \pi_i$ where

$$\pi_i = P(q_1 = S_i), 1 \leq i \leq N$$

Thus an HMM can be defined by the triplet $\lambda = (A, B, \pi)$.

An example that was discussed in class was the urn and ball model shown in figure 1.1. Each urn corresponds to a state in the HMM. In addition an urn can contain any of M different coloured balls (the observation alphabet). Based on some random initial distribution, we pick the initial urn. After selecting a ball and replacing it, we choose the next urn based on the distribution associated with the current urn. Thus we get a finite observation sequence of colours, which can be modelled as that of an HMM.

We can understand more about HMMs by assuming that the above process generated the observation sequence, and we want to build an HMM to model it. Note that we don't know the number of urns, or the transition and emission probabilities. These will be learned from the observation sequence. Suppose that each urn contains exactly the same number of balls of different colours, i.e. $b_i(1) = b_i(2) = \ldots = b_i(M)$. It is clear that at any point in the observation sequence, we are unable to determine which state we are in. In fact, we can model this sequence by an HMM that contains only one state with equal emission probabilities for all observation symbols. Thus we note that *there can be more than one correct model for a given observation sequence*. In fact, the above case is very similar to parameter tying which will be discussed in sections 1.4 and 1.5.3.

1.3 Basic Problems in HMMs

1.3.1 HMM as Language Model

Problem Statement: Given an observation sequence $O = O_1, O_2, \ldots, O_T$ and a model $\lambda = (A, B, \pi)$ compute $P(O|\lambda)$.

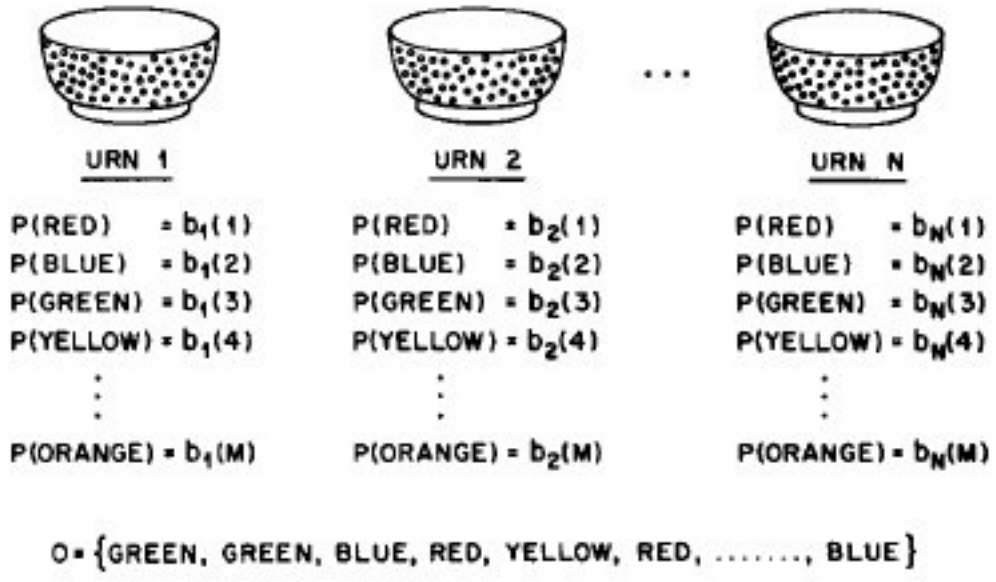

Figure 1.1. An N-state urn and ball model which illustrates the working of an HMM

The solution to this problem is a special case of the solution to the third problem (using the Forward-Backward algorithm) which is covered in Subsection 1.3.3. Here we don't need the backward step and just compute the forward α terms and we get

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

1.3.2 HMM as a parser

Problem Statement: Given an observation sequence $O = O_1, O_2, \ldots, O_T$ and a model $\lambda = (A, B, \pi)$, compute the "best" sequence of states that explains O.

The solution to this problem is the *Viterbi* algorithm. It is useful to think of a trellis structure as shown in the figure 1.2. Each column of states represents a particular time step. In the figure, an HMM is shown which has two states q and r. The idea of the Viterbi algorithm is to store the best path upto each state. Call this best path $\delta_t(i)$ (the best path cost at time

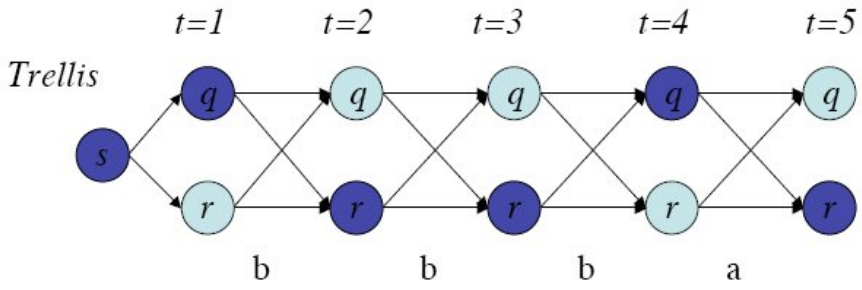

Figure 1.2. Trellis

step t and ending in state S_i). We can define this recursively as follows

$$\delta_{t+1}(i) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1})$$

Thus we can efficiently compute the best path upto each state. To get the actual sequence of states we backtrack along the best path to get the sequence of states that were chosen.

1.3.3 HMM as a learner

Problem Statement: Given an observation sequence $O = O_1, O_2, \ldots, O_T$, optimize the model $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$.

This is the hardest problem among the three. There is no known way to analytically solve for the model such that $P(O|\lambda)$ is maximized. We do not have a simple way of estimating the model parameters $\lambda = (A, B, \pi)$. We can however, arrive at a *local maximum* of $P(O|\lambda)$ given the model parameters. To do this, we use the *Forward-Backward* algorithm named because of the use of the forward and backward variables (These can be viewed as using a different semi-ring than that in the Viterbi algorithm).

- The forward variable $\alpha_t(i)$ stores the probability of the partial observation sequence upto time t and the state S_i at time t, given the model i.e. $P(O_1, O_2, \ldots, O_t, q_t = S_i|\lambda)$. The initial values can be obtained from the initial distribution probabilities π. α can be defined recursively as,

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(O_{t+1})$$

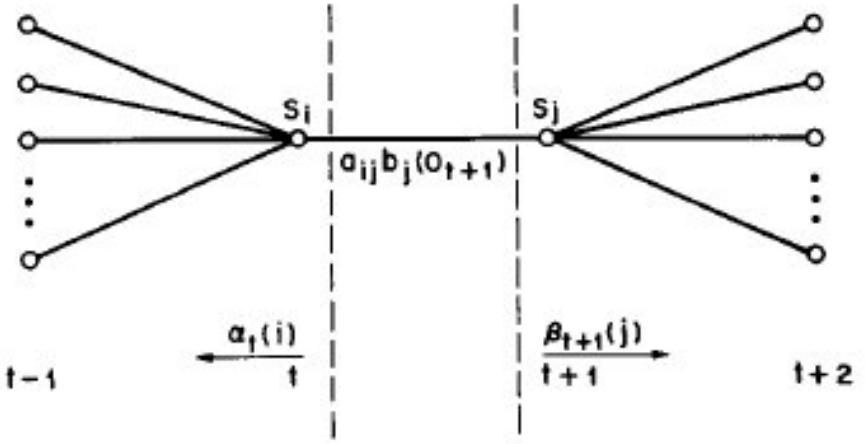

Figure 1.3. Forward-Backward algorithm

- The backward variable $\beta_t(i)$ is defined analogously as the probability of the partial observation sequence *after* time t, given that the state at time t was S_i and given the model. $\beta_T(i)$ is assumed to be 1. It can again be defined recursively as,

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

The reestimation procedure considers the probability of the *link* shown in the figure 1.3, i.e $\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_{i+1}|O, \lambda)$. This can be rewritten in terms of the forward and backward variables.

The reestimation formulas for π, A and B can be computed using the above variables, by counting the expected frequency of occurrence of that event. For details please refer to the Rabiner paper [1].

1.4 Types of HMMs

HMMs can vary depending on their structure.

- Ergodic HMMs: These are HMMs in which each state can be reached from any other in a finite number of steps.

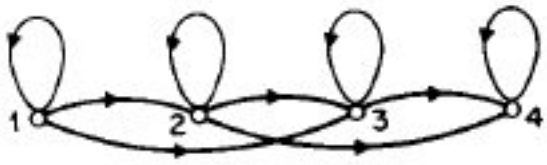

Figure 1.4. Left to right HMM

- Left-to-right HMMs: These are ones like those shown in figure 1.4, where once a state is changed, the previous state cannot be reached.
- HMMs in which observations are associated with the transitions, rather than the states. Of particular importance in these is the null-transition which is a state transition without consuming an observation symbol.

Many modifications could be made to different parts of an HMM. Some of them are listed below

- Continuous Observation Densities: Here instead of having a discrete set of observations symbols, we have a continuous distribution.
- Tied Parameters: Sometimes the number of parameters in the HMM becomes prohibitively large. In such cases, we can reduce the number of parameters of the system by "tying" some parameters together. The choice of parameters to tie depends on the domain. An example that was discussed in class is the case of part of speech tags. Sometimes the number of tags is large and there is insufficient data to train the model. There would be certain tags or tag sequences that would not be seen in the data. In this case, we can tie the observation probabilities of all the unseen tag sequences as one parameter.
- Explicit State Duration Density: The inherent duration density in the above models is exponential. The example discussed in class (and in the Rabiner paper [1]) is the one where we want to determine the expected number of days the weather doesn't change given some probability distribution. Sometimes this is undesirable and we would like to have an explicit duration density. This causes modifications in the forward and backward variables and is discussed in the paper [1]. Usually having explicit duration densities causes a large increase in processing time.

- Different Optimization Criterion: Instead of the maximum likelihood reestimation procedure described above, one can train discriminative HMM models which are a group of models such that the "Mutual Information" is maximized or the "Discrimination Information" is minimized.

1.5 Implementation Issues

1.5.1 Scaling

The forward and backward variables are a product of a number of transition and emission probabilities. Given a large enough HMM, it is possible that some of these values underflow and impact the algorithms described above. To overcome this, at each time step we scale the forward and backward variables by the scaling coefficient

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$$

It is to be noted that even the backward variables are scaled by this factor. This is done so that in the reestimation equations, the scaling terms cancel out and we get the exact answer. Also, it should be mentioned that the above procedure is not required for the Viterbi algorithm, as we can take the log scale probabilities and just sum them up to get the best path. This will avoid any underflow problems.

1.5.2 Multiple Observation Sequences

Sometimes we train the model based on multiple observation sequences instead of a single one as described above. Again, an example for this would be the part of speech tagging task, where each sentence is considered as a new observation sequence. The algorithms described above change based on the fact that we now need to optimize $P(\mathbf{O}|\lambda)$, where $\mathbf{O}$ is the vector of observation sequences.

1.5.3 Insufficient Data

As discussed in Section 1.4, sometimes the model is too large to be modelled accurately by the given data. In this case, we tie some of the parameters of

the initial model λ to get a new reduced model λ'. Our final model can be a weighted combination of the two,

$$\lambda_f = \epsilon\lambda + (1 - \epsilon)\lambda'$$

Here ϵ is the weighting coefficient which is closer to 1 if there is a large amount of data and closer to 0 if the data is insufficient. To find out the value of epsilon, we can again use a variant HMM model (where the transitions correspond to observations and null-transitions are possible) and train this on a held-out subset of the data. This technique is called *deleted interpolation*.

Bibliography

[1] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 267–296. Kaufmann, San Mateo, CA, 1990.