PGM 2004/05 Tirgul 2

Hidden Markov Models

based on Gal Elidan notes

Introduction

Hidden Markov Models (HMM) are one of the most common form of probabilistic graphical models, although they were developed long before the notion of general models existed (1913). They are used to model time-invariant and limited horizon models that have both an underlying mechanism (hidden states) and an observable consequence. They have been extremely successful in language modeling and speech recognition systems and are still the most widely used technique in these domains.

Markov Models

A Markov process or model assumes that we can predict the future based just on the present (or on a limited horizon into the past):

Let $\{X_1,\ldots,X_T\}$ be a sequence of random variables taking values $\{1,\ldots,N\}$ then the Markov properties are:

Limited Horizon:

$$P(X_{t+1} = i \mid X_1,\ldots,X_t) = P(X_{t+1} = i \mid X_t)$$

Time invariant (stationary):

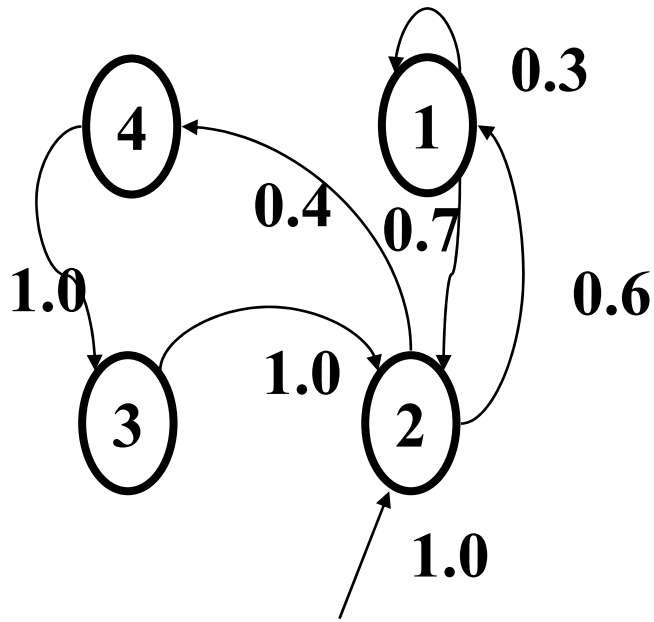
$$= P(X_2 = i \mid X_1)$$

Describing a Markov Chain

A Markov chain can be described by the transition matrix A and initial state probabilities Q:

$$A_{ij} = P(X_{t+1} = j \mid X_t = i) \qquad q_i = P(X_1 = i)$$

or alternatively:



and we calculate:

$$P(X_1,\ldots,X_T) = P(X_1)P(X_2 \mid X_1)\ldots P(X_T \mid X_{T-1}) = q_{X_1}\prod_{t=1}^{T-1} A(X_t, X_{t+1})$$

Hidden Markov Models

In a Hidden Markov Model (HMM) we do not observe the sequence that the model passed through but only some probabilistic function of it. Thus, it is a Markov model with the addition of ***emission probabilities***:

$$B_{ik} = P(Y_t = k \mid X_t = i)$$

Why use HMMs?

- A lot of real-life processes are composed of underlying events generating surface phenomena. Tagging parts of speech is a common example.
- We can usually think of processes as having a limited horizon (we can easily extend to the case of a constant horizon larger than 1)
- We have efficient training algorithm using EM
- Once the model is set, we can easily run it:

t=1, start in state i with probability q_i

forever : move from state i to j with probability a_{ij}

emit y_t=k with probability b_{ik}

t=t+1

The fundamental questions

Likelihood: Given a model $\lambda=(A,B,q)$, how do we efficiently compute the likelihood of an observation $P(Y|\lambda)$?

Decoding: Given the observation sequence Y and a model μ, what state sequence explains it best (MPE)?
This is, for example, the tagging process of an observed sentence.

Learning: Given an observation sequence Y, and a generic model, how do we estimate the parameters that define the best model to describe the data?

Computing the Likelihood

Using P(Y,X)=P(Y|X)P(X) we get:

$$P(Y) = \sum_X P(Y, X) = \sum_X P(Y \mid X) P(X)$$

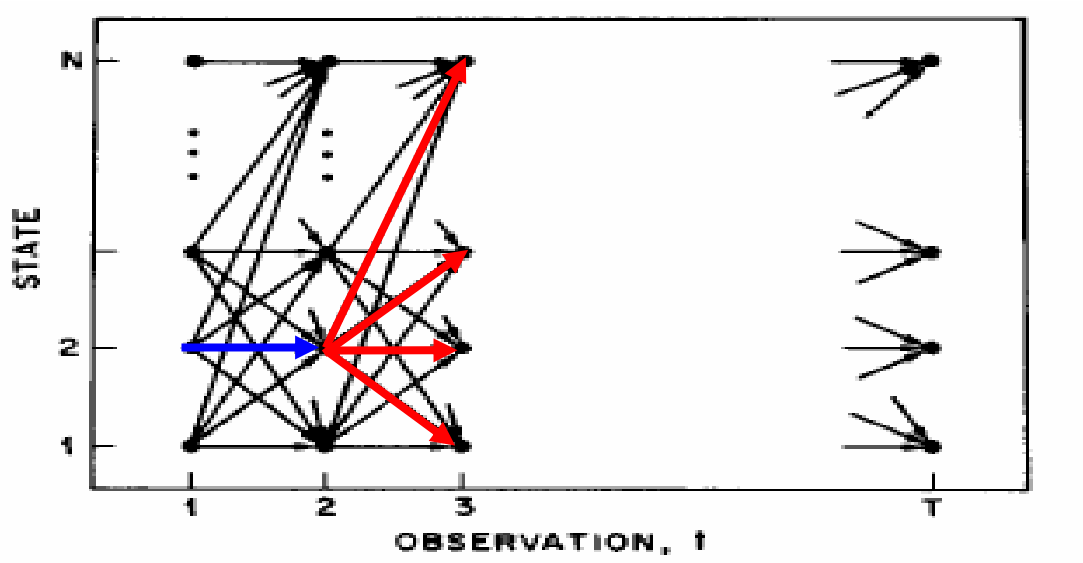
$$= \sum_{x_1 \cdots x_T} q_{x_1} \prod_{t=2}^{T} A_{x_{t-1} x_t} B_{x_t y_t}$$

But, we have $O(TN^T)$ multiplications!

The trellis (lattice) algorithm

To compute likelihood: Need to enumerate over all paths in the lattice (all possible instantiations of $X_1 \ldots X_T$).

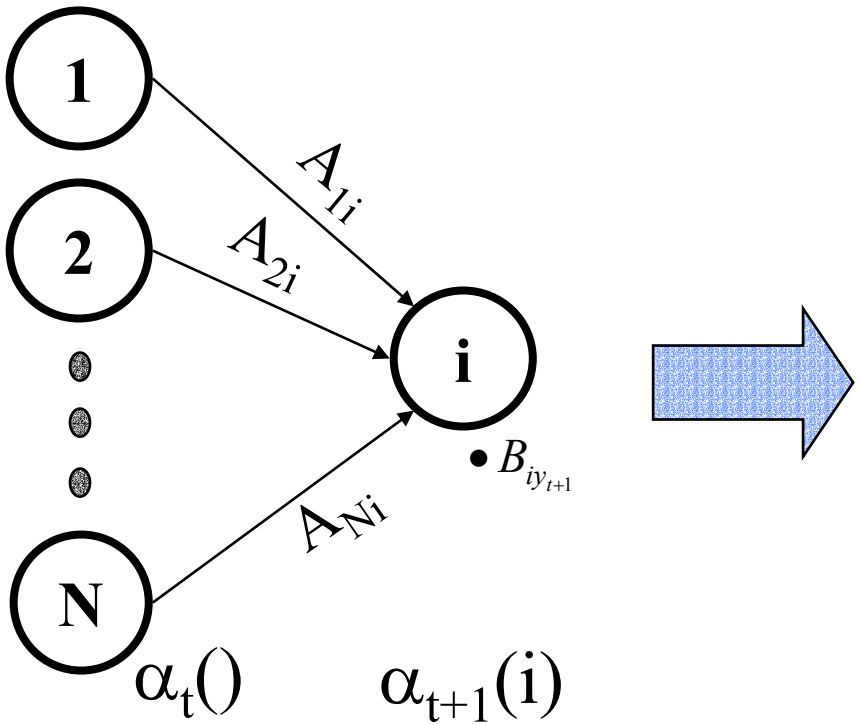
But… some starting subpath (blue) is common to many continuing paths (blue+red)



Idea: using dynamic programming, calculate a path in terms of shorter sub-paths

The trellis (lattice) algorithm

We build a matrix of the probability of being at time t at state i - $\alpha_t(i)=P(x_t=i,y_1y_2\ldots y_t)$ is a function of the previous column (forward procedure):



$$\alpha_1(i) = q_i B_{jy_1}$$

$$\alpha_{t+1}(i) = B_{iy_{t+1}} \sum_{j=1}^{N} \alpha_t(j) A_{ji}$$

$$P(Y) = \sum_{i=1}^{N} \alpha_T(i)$$

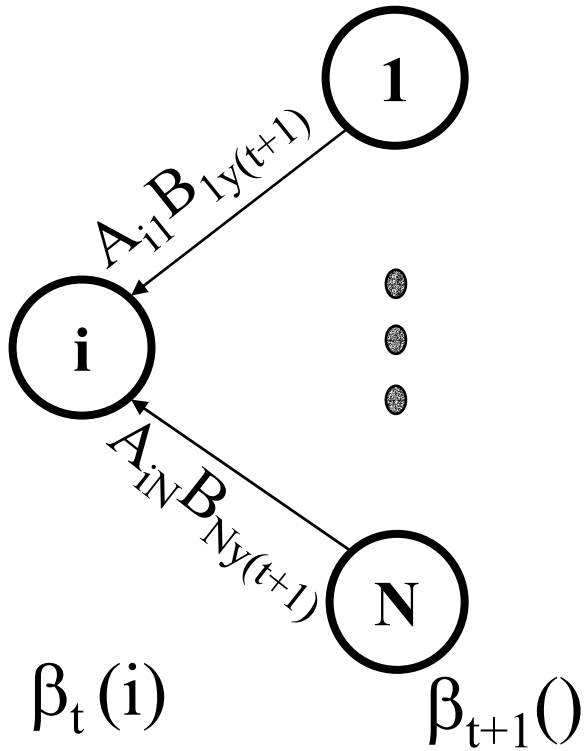
The trellis (lattice) algorithm (cont.)

We can similarly define a backwards procedure for filling the matrix $\beta_t(i) = P(y_{t+1} \cdots y_T \mid X_t = i)$

$$\beta_T(i) = 1$$

$$\beta_t(i) = \sum_{j=1}^{n} A_{ij} B_{jy_{t+1}} \beta_{t+1}(j)$$

$$P(Y) = \sum_{i=1}^{N} q_i B_{iy_1} \beta_1(i)$$



The trellis (lattice) algorithm (cont.)

And we can easily combine:

$$
\begin{aligned}
P(y, X_t = i) &= P(y_1 \cdots y_T, X_t = i) \\
&= P(y_1 \cdots y_t, X_t = i, y_{t+1} \cdots y_T) \\
&= P(y_1 \cdots y_t, X_t = i) P(y_{t+1} \cdots y_T \mid y_1 \cdots y_t, X_t = i) \\
&= P(y_1 \cdots y_t, X_t = i) P(y_{t+1} \cdots y_T \mid X_t = i) \\
&= \alpha_t(i) \beta_t(i)
\end{aligned}
$$

And then

$$
P(y) = \sum_i P(y, X_t = i) = \sum_i \alpha_t(i) \beta_t(i)
$$

Finding the best state sequence

We would like to the most likely path (and not just the most likely state at each time slice)

$$\arg\max_x P(X \mid Y) = \arg\max_x P(X, Y)$$

The Viterbi algorithm is an efficient trellis method for finding the MPE:

$$\delta_1(i) = q_i B_{iy_1}$$

$$\delta_{t+1}(i) = B_{iy_{t+1}} \max_j \delta_t(j) A_{ji} \qquad \gamma_{t+1}(i) = \arg\max_j \delta_t(j) A_{ji}$$

and we to reconstruct the path:

$$P(\hat{X}) = \max_i \delta_T(i) \quad \hat{X}_T = \arg\max_i \delta_T(i)$$

$$\hat{X}_{t-1} = \gamma_t(\hat{X}_t)$$

The Casino HMM

A casino switches from a fair die (state F) to a loaded one (state U) with probability 0.05 and the other way around with probability 0.1. The loaded die has 0.5 probability of showing a 6. The casino, honestly, reads off the number that was rolled.

$$A = \begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.9 \end{pmatrix} \qquad B = \begin{pmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/10 & 1/10 & 1/10 & 1/10 & 1/10 & 1/2 \end{pmatrix}$$

$$P(3151166661 \mid FFFFFUUUUU) =$$

$$1/6 \bullet 1/6 \bullet 1/6 \bullet 1/6 \bullet 1/6 \bullet 1/2 \bullet 1/2 \bullet 1/2 \bullet 1/2 \bullet 1/10$$

The Casino HMM (cont.)

What is the likelyhood of 3151166661?

Y= 3 1 5 1 1 6 6 6 6 1

$\alpha_1(1)=0.5*1/6=1/12,$

$\alpha_1(2)=0.5*0.1=0.05$

$\alpha_2(1)=1/6*(0.95*1/12+0.1*0.05) \cong 0.014$

$\alpha_2(1)=0.1*(0.05*1/12+0.9*0.05) \cong 0.0049$

$\alpha_3(1) \cong 0.0023,\ \alpha_3(2) \cong 0.0005$

$\alpha_4(1) \cong 0.0004,\ \alpha_4(1) \cong 0.0001$

$\alpha_5(1) \cong 0.0001,\ \alpha_5(1) < 0.0001$

… all smaller then 0.0001!

The Casino HMM (cont.)

What explains 3151166661 best?

Y= 3 1 5 1 1 6 6 6 6 1

$\delta_1(1)=0.5*1/6=1/12$, $\delta_1(2)=0.5*0.1=0.05$

$\delta_2(1)=1/6*\max(0.95*1/12,0.1*0.05) \cong 0.0132$

$\delta_2(1)=1/10*\max(0.05*1/12, 0.9*0.05) \cong 0.0045$

$\delta_3(1) \cong 0.0021$, $\delta_3(2) \cong 0.0004$

$\delta_4(1) \cong 0.0003$, $\delta_4(1) < 0.0001$

$\delta_5(1) \cong 0.0001$, $\delta_5(1) < 0.001$ …

…

The Casino HMM (cont.)

An example of reconstruction using Viterbi (Durbin):

Rolls	3151162464466442453113216311641521336
Die	0000000000000000000000000000000000
Viterbi	0000000000000000000000000000000000
Rolls	2514454363165662656666665116645313265
Die	0000000011111111111111111111100000000
Viterbi	0000000000011111111111111111100000000
Rolls	1245636664631636663162326455236266666
Die	0000111111111111111110001111111111111
Viterbi	0000111111111111111111111111111111111

Learning

If we were given both X and Y, we could choose

$$\arg\max_{A,B} P(Y_{training}, X_{training} \mid A, B, Q)$$

Using the Maximum Likelihood principal, we simply assign the parameter for each relative frequency

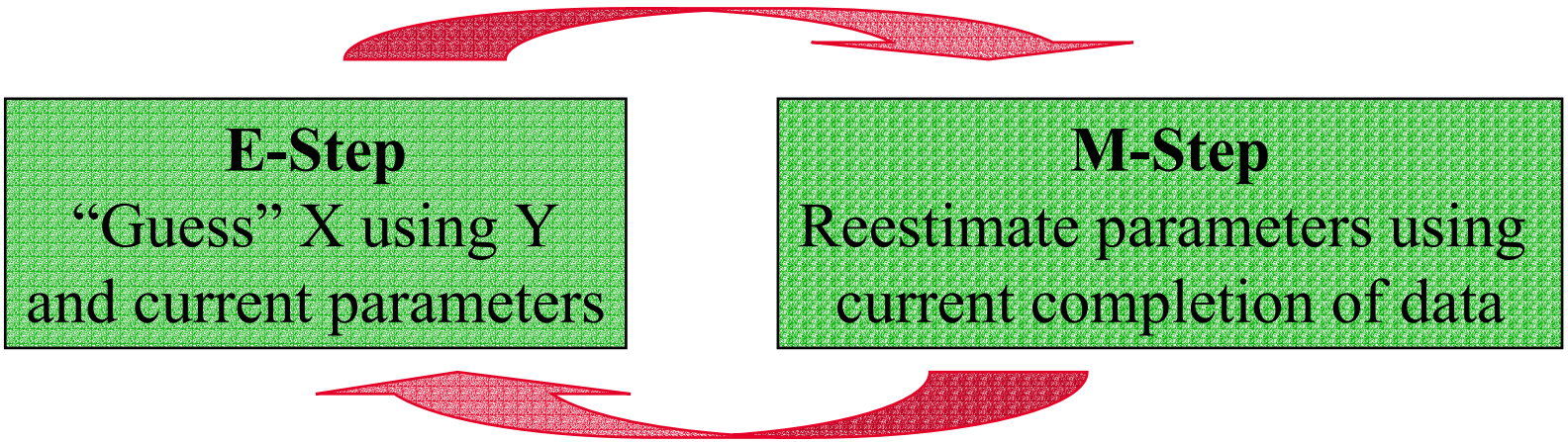
What do we do when we have only Y?

$$\arg\max_{A,B} P(Y_{training} \mid A, B, Q)$$

ML here does not have a closed form formula!

EM (Baum Welch)

Idea: Using current guess to complete data and re-estimate



Thm: Likelihood of observables never decreases!!!
(to be proved later in the course)

Problems: Gets stuck at sub-optimal solutions

Parameter Estimation

We define the expected number of transitions from state i to j at time t:

$$p_t(i,j) = P(X_t = i, X_{t+1} = j \mid Y) = \frac{\alpha_t(i) A_{ij} B_{jy_{t+1}} \beta_{t+1}(j)}{\sum_m \sum_n \alpha_t(m) A_{mn} B_{ny_{t+1}} \beta_{t+1}(n)}$$

the expected # of transition from i to j in Y is then

$$\sum_{t=1}^{T} p_t(i,j)$$

Parameter Esimation (cont.)

We use EM re-estimation formulas using the expected counts we already have:

$$\bar{q}_i = \text{expected frequency in state } i \text{ at time } 1 = \sum_j p_1(i,j)$$

$$\bar{a}_{ij} = \frac{\text{expected \# of transitions from state } i \text{ to } j}{\text{expected \# of transitions from state } i} = \frac{\sum_t p_t(i,j)}{\sum_t \sum_j p_t(i,j)}$$

$$\bar{b}_{ik} = \frac{\text{expected \# of emissions of } k \text{ from state } i}{\text{expected \# of emissions from state } i} = \frac{\sum_j \sum_{t:y_t=k} p_t(i,j)}{\sum_t \sum_j p_t(i,j)}$$