Biostatistics 615/815 Lecture 12: Hidden Markov Models

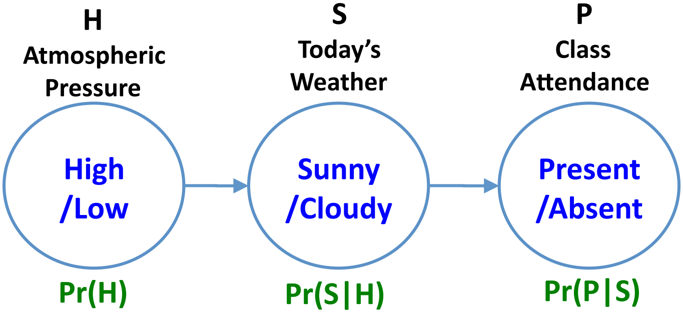
Hyun Min Kang

February 15th, 2011

Graphical Models 101

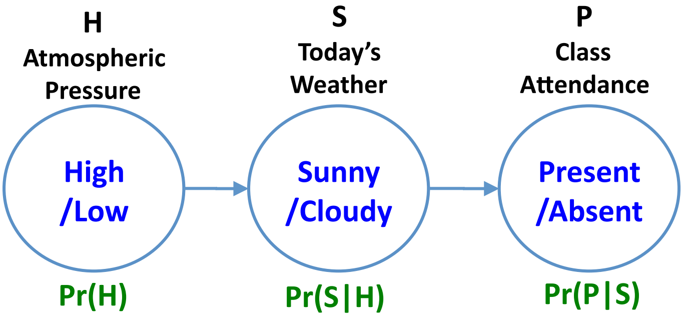
- Marriage between probability theory and graph theory
- Each random variable is represented as vertex
- Dependency between random variables is modeled as edge
    - Directed edge : conditional distribution
    - Undirected edge : joint distribution
- Unconnected pair of vertices (without path from one to another) is independent
- A powerful tool to represent complex structure of dependence / independence between random variables.

An example graphical model



- Are H and P independent?

An example graphical model



- Are H and P independent?
- Are H and P independent given S?

Example probability distribution

$\Pr(H)$

Value (H)	Description (H)	$\Pr(H)$
0	Low	0.3
1	High	0.7

$\Pr(S\|H)$

S	Description (S)	H	Description (H)	$\Pr(S\|H)$
0	Cloudy	0	Low	0.7
1	Sunny	0	Low	0.3
0	Cloudy	1	High	0.1
1	Sunny	1	High	0.9

Probability distribution (cont'd)

$\Pr(P|S)$

P	Description (P)	S	Description (S)	$\Pr(P\|S)$
0	Absent	0	Cloudy	0.5
1	Present	0	Cloudy	0.5
0	Absent	1	Sunny	0.1
1	Present	1	Sunny	0.9

Full joint distribution

$\Pr(H, S, P)$

H	S	P	$\Pr(H, S, P)$
0	0	0	0.105
0	0	1	0.105
0	1	0	0.009
0	1	1	0.081
1	0	0	0.035
1	0	1	0.035
1	1	0	0.063
1	1	1	0.567

- With a full join distribution, any type of inference is possible
- As the number of variables grows, the size of full distribution table increases exponentially

$\Pr(H, P|S) = \Pr(H|S)\Pr(P|S)$

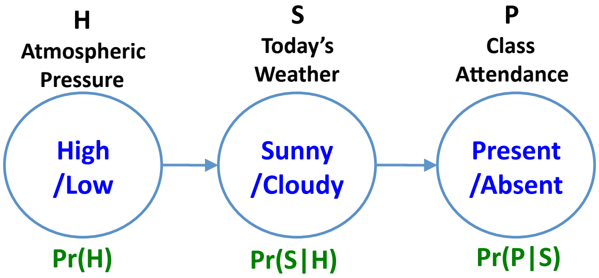
$\Pr(H, P|S)$

H	P	S	$\Pr(H, P\|S)$
0	0	0	0.3750
0	1	0	0.3750
1	0	0	0.1250
1	1	0	0.1250
0	0	1	0.0125
0	1	1	0.1125
1	0	1	0.0875
1	1	1	0.7875

$\Pr(H|S)$, $\Pr(P|S)$

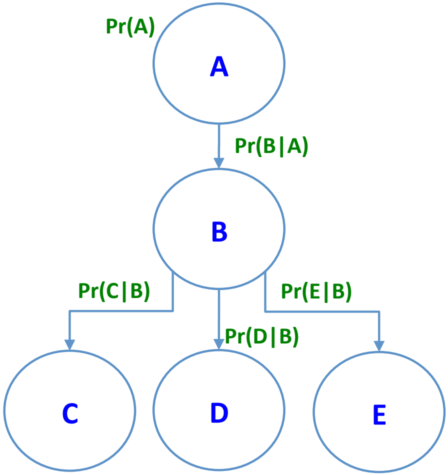
H	S	$\Pr(H\|S)$	P	S	$\Pr(P\|S)$
0	0	0.750	0	0	0.500
1	0	0.250	1	0	0.500
0	1	0.125	0	1	0.100
1	1	0.875	1	1	0.900

H and P are conditionally independent given S



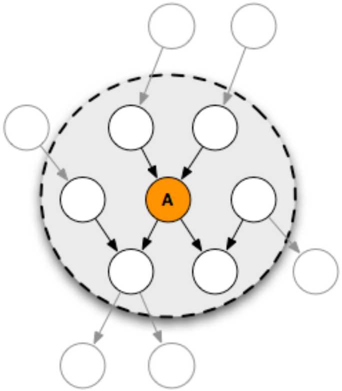
- H and P do not have direct path one from another
- All path from H to P is connected thru S.
- Conditioning on S separates H and P

Conditional independence in graphical models



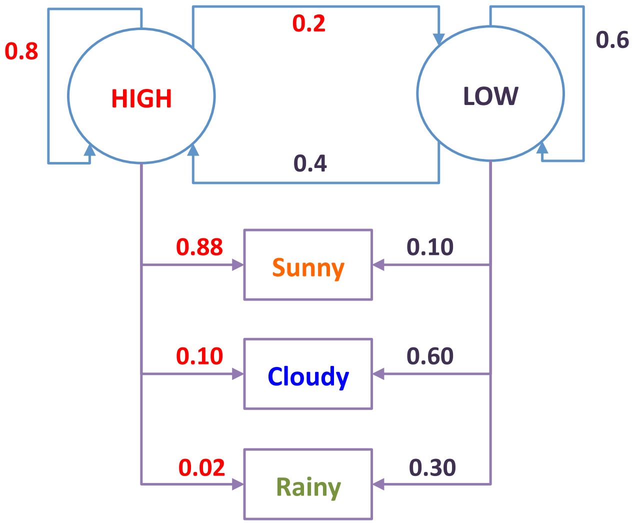
- $\Pr(A, C, D, E|B) = \Pr(A|B)\Pr(C|B)\Pr(D|B)\Pr(E|B)$

Markov Blanket

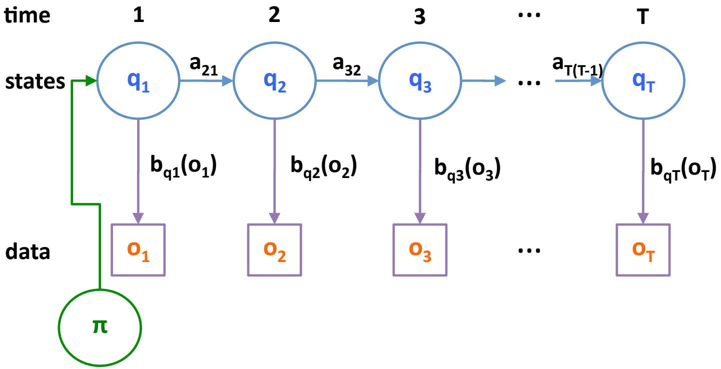


- If conditioned on the variables in the gray area (variables with direct dependency), A is independent of all the other nodes.
- $A \perp (U - A - \pi_A) | \pi_A$

Hidden Markov Models - An Example



An alternative representation of HMM



Marginal likelihood of data in HMM

- Let $\lambda = (A, B, \pi)$
- For a sequence of observation $\mathbf{o} = \{o_1, \cdots, o_t\}$,

$$\begin{aligned}
\Pr(\mathbf{o}|\lambda) &= \sum_{\mathbf{q}} \Pr(\mathbf{o}|\mathbf{q}, \lambda) \Pr(\mathbf{q}|\lambda) \\
\Pr(\mathbf{o}|\mathbf{q}, \lambda) &= \prod_{i=1}^{t} \Pr(o_i|q_i, \lambda) = \prod_{i=1}^{t} b_{q_i}(o_i) \\
\Pr(\mathbf{q}|\lambda) &= \pi_{q_1} \sum_{i=2}^{t} a_{q_i q_{i-1}} \\
\Pr(\mathbf{o}|\lambda) &= \sum_{\mathbf{q}} \pi_{q_1} b_{q_1}(o_{q_1}) \prod_{i=2}^{t} a_{q_i q_{i-1}} b_{q_i}(o_{q_i})
\end{aligned}$$

Forward and backward probabilities

$$
\begin{aligned}
\mathbf{q}_t^- &= (q_1, \cdots, q_{t-1}), \qquad \mathbf{q}_t^+ = (q_{t+1}, \cdots, q_T) \\
\mathbf{o}_t^- &= (o_1, \cdots, o_{t-1}), \qquad \mathbf{o}_t^+ = (o_{t+1}, \cdots, o_T) \\
\Pr(q_t = i|\mathbf{o}, \lambda) &= \frac{\Pr(q_t = i, \mathbf{o}|\lambda)}{\Pr(\mathbf{o}|\lambda)} = \frac{\Pr(q_t = i, \mathbf{o}|\lambda)}{\sum_{j=1}^n \Pr(q_t = j, \mathbf{o}|\lambda)} \\
\Pr(q_t, \mathbf{o}|\lambda) &= \Pr(q_t, \mathbf{o}_t^-, o_t, \mathbf{o}_t^+|\lambda) \\
&= \Pr(\mathbf{o}_t^+|q_t, \lambda)\Pr(\mathbf{o}_t^-|q_t, \lambda)\Pr(o_t|q_t, \lambda)\Pr(q_t|\lambda) \\
&= \Pr(\mathbf{o}_t^+|q_t, \lambda)\Pr(\mathbf{o}_t^-, o_t, q_t|\lambda) \\
&= \beta_t(q_t)\alpha_t(q_t)
\end{aligned}
$$

If $\alpha_t(q_t)$ and $\beta_t(q_t)$ is known, $\Pr(q_t|\mathbf{o}, \lambda)$ can be computed in a linear time.

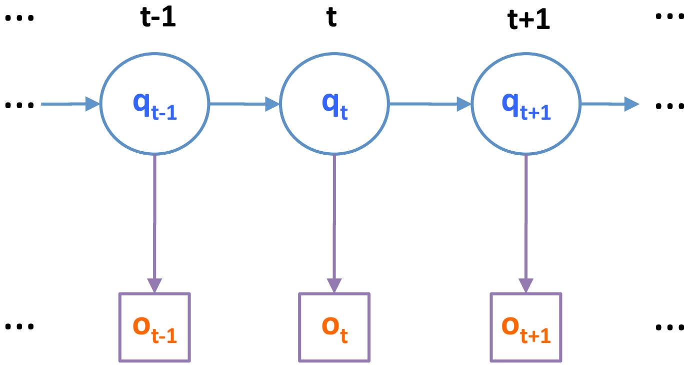
DP algorithm for calculating forward probability

- Key idea is to use $(q_t, o_t) \perp \mathbf{o}_t^- | \mathbf{q}_{t-1}$.

$$\begin{aligned}
\alpha_t(i) &= \Pr(o_1, \cdots, o_t, q_t = i|\lambda) \\
&= \sum_{j=1}^{n} \Pr(\mathbf{o}_t^-, o_t, q_{t-1} = j, q_t = i|\lambda) \\
&= \sum_{j=1}^{n} \Pr(\mathbf{o}_t^-, q_{t-1} = j|\lambda) \Pr(q_t = i|q_{t-1} = j, \lambda) \Pr(o_t|q_t = i, \lambda) \\
&= \sum_{j=1}^{n} \alpha_{t-1}(j) a_{ij} b_i(o_t) \\
\alpha_1(i) &= \pi_i b_i(o_1)
\end{aligned}$$

Conditional dependency in forward-backward algorithms

- Forward : $(q_t, o_t) \perp \mathbf{o}_t^- | \mathbf{q}_{t-1}$.
- Backward : $o_{t+1} \perp \mathbf{o}_{t+1}^+ | \mathbf{q}_{t+1}$.



DP algorithm for calculating backward probability

- Key idea is to use $o_{t+1} \perp \mathbf{o}^{+}_{t+1} | \mathbf{q}_{t+1}$.

$$
\begin{aligned}
\beta_t(i) &= \Pr(o_{t+1}, \cdots, o_T | q_t = i, \lambda) \\
&= \sum_{j=1}^{n} \Pr(o_{t+1}, \mathbf{o}^{+}_{t+1}, q_{t+1} = j | q_t = i, \lambda) \\
&= \sum_{j=1}^{n} \Pr(o_{t+1} | q_{t+1}, \lambda) \Pr(\mathbf{o}^{+}_{t+1} | q_{t+1} = j, \lambda) \Pr(q_{t+1} = j | q_t = i, \lambda) \\
&= \sum_{j=1}^{n} \beta_{t+1}(j) a_{ji} b_j(o_{t+1}) \\
\beta_T(i) &= 1
\end{aligned}
$$

Putting forward and backward probabilities together

- Conditional probability of states given data

$$\Pr(q_t = i|\mathbf{o}, \lambda) = \frac{\Pr(\mathbf{o}, q_t = S_i|\lambda)}{\sum_{j=1}^{n} \Pr(\mathbf{o}, q_t = S_j|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{n} \alpha_t(j)\beta_t(j)}$$

- Time complexity is $\Theta(n^2 T)$.

Finding the most likely trajectory of hidden states

- Given a series of observations, we want to compute

$$\arg\max_{\mathbf{q}} \Pr(\mathbf{q}|\mathbf{o}, \lambda)$$

- Define $\delta_t(i)$ as

$$\delta_t(i) = \max_{\mathbf{q}} \Pr(\mathbf{q}, \mathbf{o}|\lambda)$$

- Use dynamic programming algorithm to find the 'most likely' path

The Viterbi algorithm

Initialization $\delta_1(i) = \pi b_i(o_1)$ for $1 \leq i \leq n$.

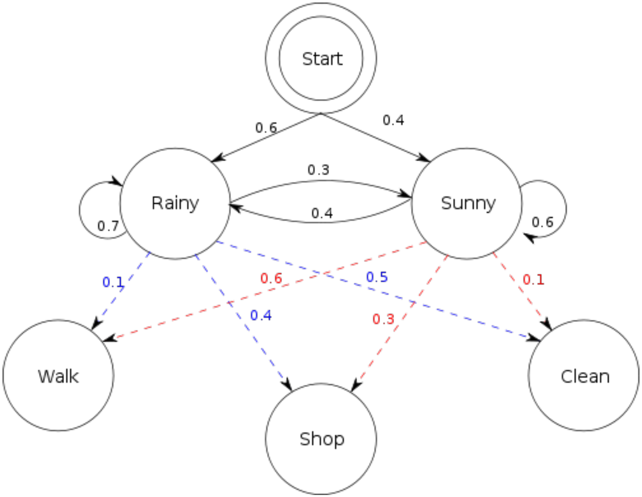
Maintenance $\delta_t(i) = \max_j \delta_{t-1}(j) a_{ij} b_i(o_t)$

$\phi_t(i) = \arg\max_j \delta_{t-1}(j) a_{ij}$

Termination Max likelihood is $\max_i \delta_T(i)$

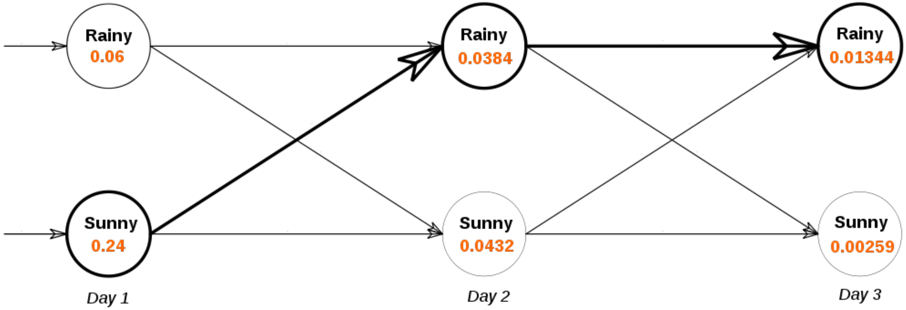
Optimal path can be backtracked using $\phi_t(i)$

An HMM example



An example Viterbi path

- When observations were (walk, shop, clean)
- Similar to Dijkstra's or Manhattan tourist algorithm



A working example : Occasionally biased coin

A generative HMM

- Observations : $O = \{1(Head), 2(Tail)\}$
- Hidden states : $S = \{1(Fair), 2(Biased)\}$
- Initial states : $\pi = \{0.9, 0.1\}$
- Transition probability : $A(i,j) = a_{ij} = \begin{pmatrix} 0.95 & 0.2 \\ 0.05 & 0.8 \end{pmatrix}$
- Emission probability : $B(i,j) = b_j(i) = \begin{pmatrix} 0.5 & 0.9 \\ 0.5 & 0.1 \end{pmatrix}$

Questions

- Given coin toss observations, estimate the probability of each state
- Given coin toss observations, what is the most likely series of states?

Example HMM implementations

```
// assume that T is # of states, and o is array of coin toss (0/1)
double pi[2] = {0.9,0.1};                                  // initial 0/1 probability
double trans[2][2] = { {0.95,0.2}, {0.05,0.8} }; // trans[i][j] : j->i transitio
double emis[2][2] = { {0.5,0.9}, {0.5,0.1} };    // emis[i][j] : b_j(o_i)
double* alphas = new double[T*2];  // forward probability  (i,j)->(2*i+j)
double* betas = new double[T*2];   // backward probability (i,j)->(2*i+j)

// forward algorithm
alphas[0] = pi[0] * emis[o[0]][0];
alphas[1] = pi[1] * emis[o[0]][1];
for(int i=1; i < T; ++i) {
  alphas[i*2] = (alphas[(i-1)*2] * trans[0][0] +
                 alphas[(i-1)*2+1] * trans[0][1]) * emis[o[i]][0];
  alphas[i*2+1] = (alphas[(i-1)*2] * trans[1][0] +
                   alphas[(i-1)*2+1] * trans[1][1]) * emis[o[i]][1];
}
```

Example HMM implmentations

```
// backward algorithm
betas[(T-1)*2] = 1;  betas[(T-1)*2+1] = 1;
for(int i=T-2; i >= 0; --i) {
  betas[i*2] = betas[(i+1)*2] * trans[0][0] * emis[o[i+1]][0]
             + betas[(i+1)*2+1] * trans[0][1] * emis[o[i+1]][1];
  betas[i*2+1] = betas[(i+1)*2] * trans[1][0] * emis[o[i+1]][0]
             + betas[(i+1)*2+1] * trans[1][1] * emis[o[i+1]][1];
}

// summing forward-backward probabilities
double* gammas = new double[T*2];
for(int i=0; i < T; ++i) {
  gammas[i*2] = (alphas[i*2]*betas[i*2]);
  gammas[i*2+1] = (alphas[i*2+1]*betas[i*2+1]);
  double z = gammas[i*2]+gammas[i*2+1];
  gammas[i*2] /= z;
  gammas[i*2+1] /= z;
}
```

More HMMs and beyond

Baum-Welch algorithm

- Estimate the transition and emission probabilities from data
- Iterative procedure to calculate the frequencies using E-M algorithm
- Will be introduced later

Advanced graphical models

- Conditional random field - inference using undirected graphical model
- Bayesian network - inference from generalized graphical models

Summary

Today - Hidden Markov Models

- Graphical models and conditional independence
- Forward-backward algorithm
- Viterbi algorithm
- Implementations

Next lectures

- Linear algebra
- Matrix decomposition
- Efficient matrix operations