HMKANG@UMICH.EDU

MODULE 1 / UNIT 9

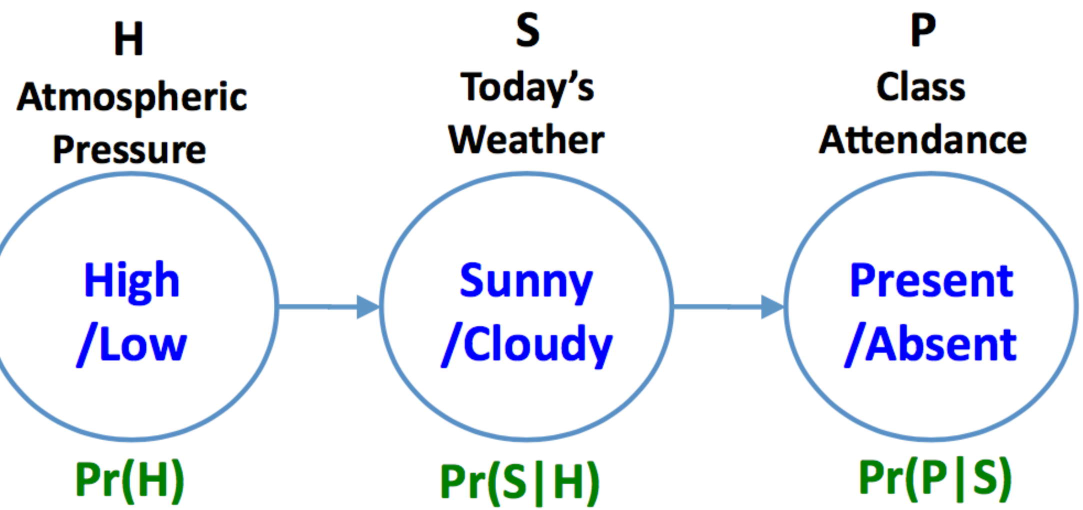
HIDDEN MARKOV MODELS

Outline

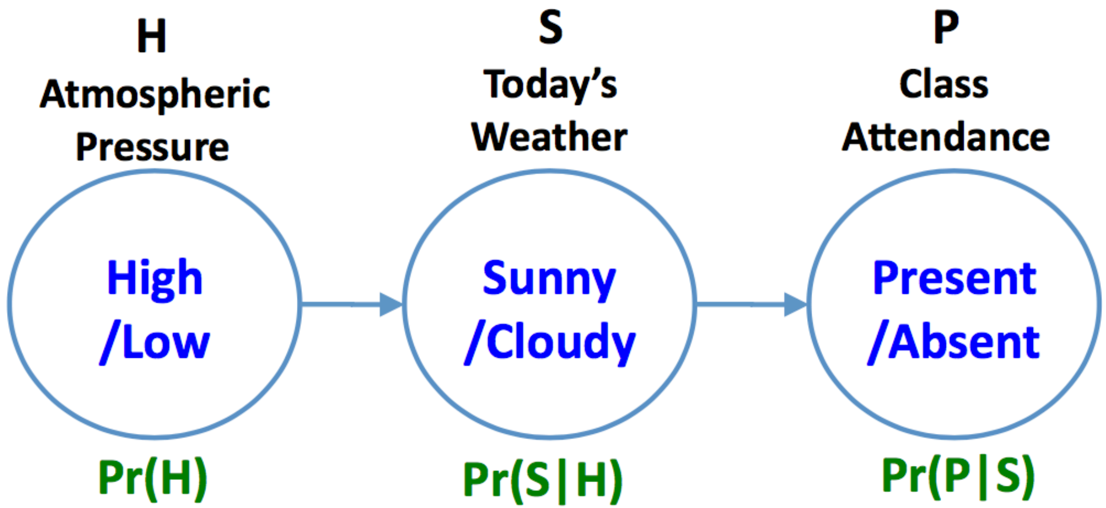
- Graphical models
- Markov process
- Hidden Markov Models (HMM)
- Forward-backward algorithm
- Viterbi algorithm

Graphical Models 101

- Marriage between probability theory and graph theory
- An effective tool to represent complex structure of dependence/independence between random variable
- Vertex : a random variable
- Edge : dependency between random variables



An example graphical model



Is $H \perp P$? (marginal independence)

Is $H \perp P \mid S$? (conditional independence)

An example probability distribution

	H	Pr(*H*)
0	Low	0.3
1	High	0.7

	S		*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

	P		*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

The full joint distribution

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

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

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

	P		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

Conditional distribution

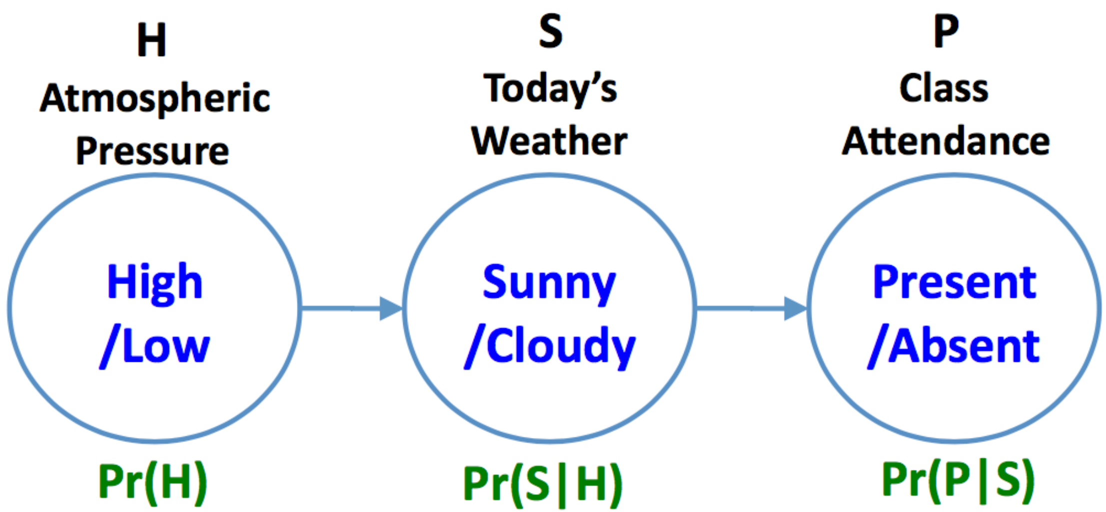
Is $H \perp P \mid S$?

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

H		S		$\Pr(H/S)$
0	Low	0	Cloudy	0.750
1	High	0	Cloudy	0.250
0	Low	1	Sunny	0.125
1	High	1	Sunny	0.875

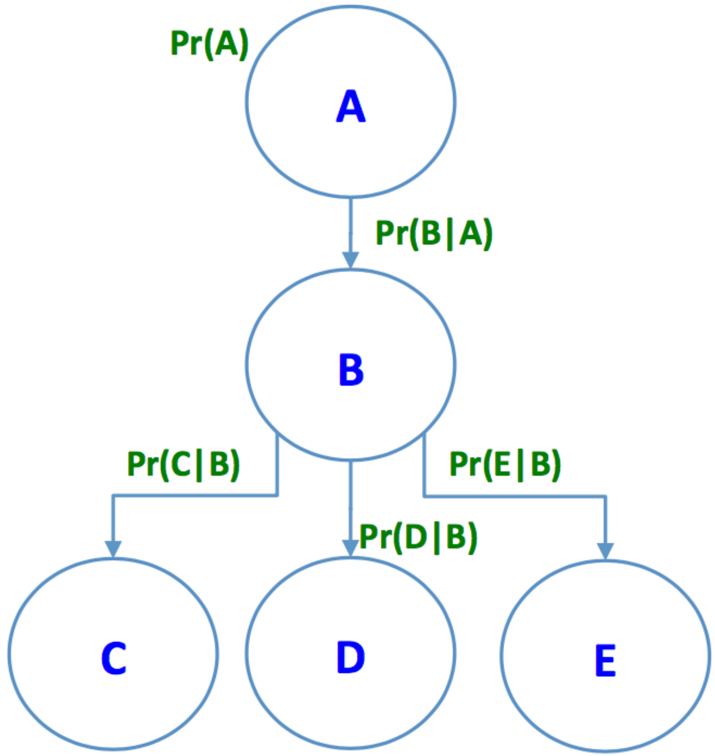
P		S		$\Pr(P/S)$
0	Absent	0	Cloudy	0.5
1	Present	0	Cloudy	0.5
0	Absent	1	Sunny	0.1
1	Present	1	Sunny	0.9

H and *P* are conditionally independent given *S*



- *H* and *P* does not have direct path in the graph
- All paths from *H* to *P* are connected through *S*
- In such cases, conditioning on *S* separates *H* and *P*

More conditional independence



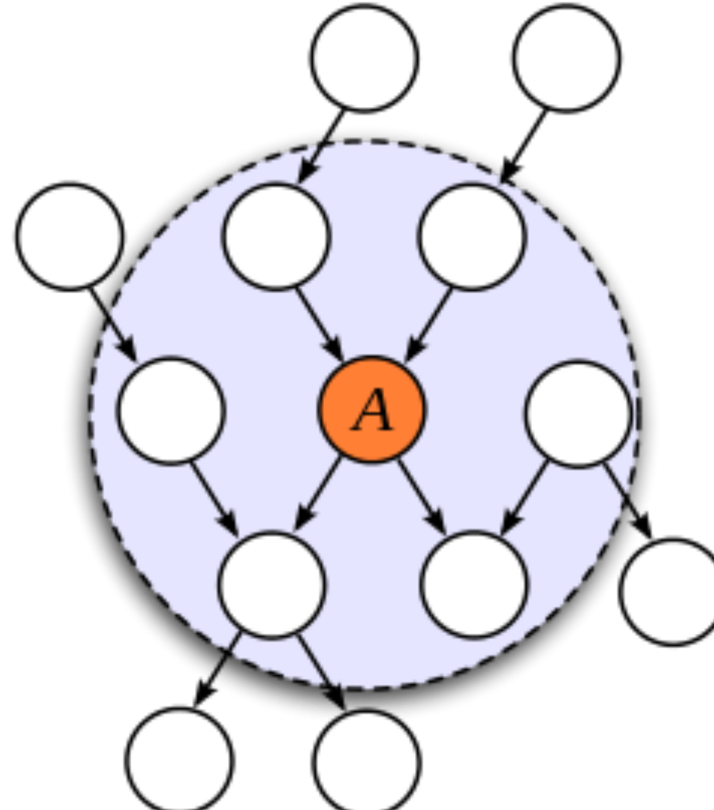
$$\Pr(A, C, D, E|B) = \Pr(A|B)\Pr(C|B)\Pr(D|B)\Pr(E|B)$$

Markov blanket

- Markov blanket ∂A for node A is a set of nodes composed of A's parents, children, and the other parents of its children.

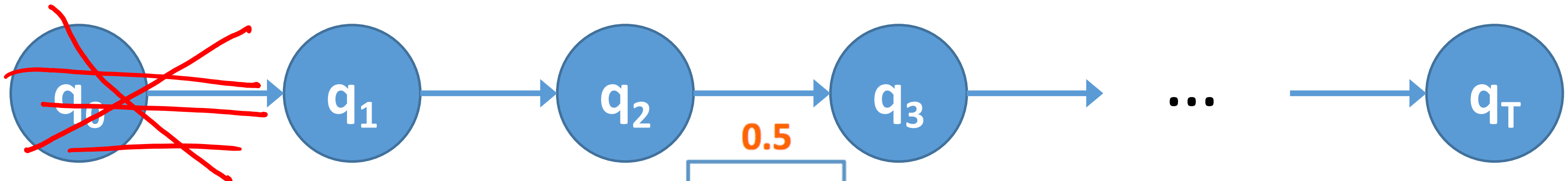
- Fact : Node A is conditionally independent given its Markov blanket ∂A.

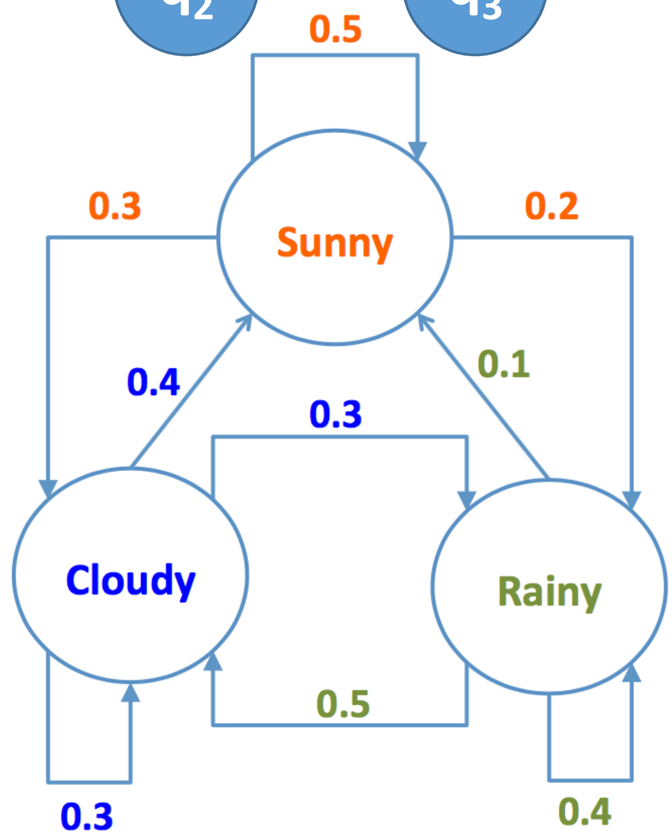
$$A \perp (U - A - \partial A) \mid \partial A$$



https://en.wikipedia.org/wiki/Markov_blanket

Markov process





State diagram

$\Pr(q_{t+1}|q_t)$

Markov process : mathematical representation

$$\pi = \begin{pmatrix} \Pr(q_1 = S_1 = \text{Sunny}) \\ \Pr(q_1 = S_2 = \text{Cloudy}) \\ \Pr(q_1 = S_3 = \text{Rainy}) \end{pmatrix} = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix}$$

$$A_{ij} = \Pr(q_{t+1} = S_j | q_t = S_i)$$

$$A = \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0.4 & 0.3 & 0.3 \\ 0.1 & 0.5 & 0.4 \end{pmatrix}$$

Markov process : example questions

- What is the chance of rain in day 2?

- If it rains today, what is the chance of rain on the day after tomorrow?

- What is the stationary distribution when T

Markov process : example questions

- What is the chance of rain in day 2?

$$\Pr(q_2 = S_3) = (A^T\pi)_3 = 0.24$$

- If it rains today, what is the chance of rain on the day after tomorrow?

$$\Pr(q_3 = S_3 | q_1 = S_3) = \left[(A^T)^2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right]_3 = 0.33$$

- What is the stationary distribution when t is very large?

$$\begin{aligned} \mathbf{p} &= A^T\mathbf{p} \\ p &= (0.346, 0.359, 0.295)^T \end{aligned}$$

Markov process dependencies

- If it rains today, what is the chance of rain on the day after tomorrow?

$$\Pr(q_3 = S_3 | q_1 = S_3) = \left[(A^T)^2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right]_3 = 0.33$$

- If it has rained for the past 3 days in a row, what is the chance of rain on the day after tomorrow?

Markov process dependencies

- If it rains today, what is the chance of rain on the day after tomorrow?

$$\Pr(q_3 = S_3 | q_1 = S_3) = \left[(A^T)^2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right]_3 = 0.33$$

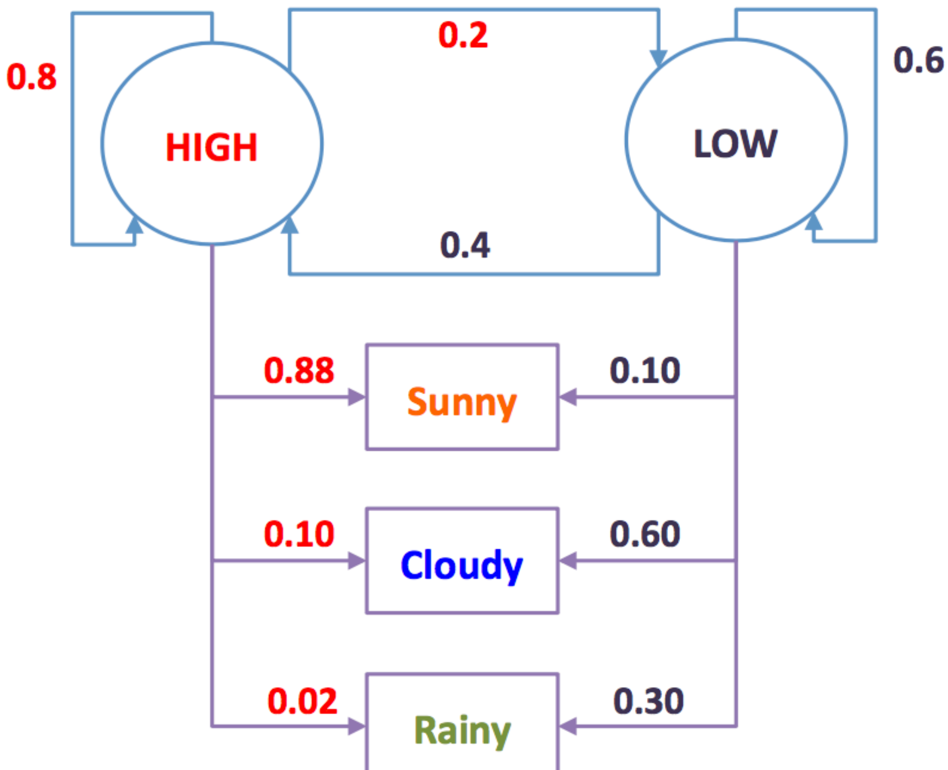
- If it has rained for the past 3 days in a row, what is the chance of rain on the day after tomorrow?

$$\Pr(q_5 = S_3 | q_1 = q_2 = q_3 = S_3) = \Pr(q_5 = S_3 | q_3 = S_3) = 0.33$$

Hidden Markov Models (HMMs)

- A graphical model composed of hidden states and observed data.
- The hidden states follow a Markov process
- The distribution of observed data depends only on the hidden state.

An example of HMM



- **Hidden states:**
  - HIGH
  - LOW
- **Observed data**
  - Sunny
  - Cloudy
  - Rainy

Mathematical representation of the example

- States $S = \{S_1, S_2\} = (\text{HIGH, LOW})$
- Observed Data $O = \{O_1, O_2, O_3\} = (\text{SUNNY, CLOUDY, RAINY})$
- Initial States $\pi_i = \Pr(q_1 = S_i)$, $\pi = \{0.7, 0.3\}$
- Transition $A_{ij} = \Pr(q_{t+1} = S_j | q_t = S_i)$

$$A = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}$$

- Emission $B_{ij} = b_{q_t}(o_t) = b_{S_i}(O_j) = \Pr(o_t = O_j | q_t = S_i)$

$$B = \begin{pmatrix} 0.88 & 0.10 & 0.02 \\ 0.10 & 0.60 & 0.30 \end{pmatrix}$$

Marginal probabilities

- **What is the chance of rain in the day 4?**

0.669×0.88
$+0.331 \times 0.10$

$0.\cancel{331}669 \times 0.02$
$+0.331 \times 0.30$

0.099
$+$
0.013

$$\begin{pmatrix} \Pr(q_4 = S_1) \\ \Pr(q_4 = S_2) \end{pmatrix} = (A^T)^3 \pi = \begin{pmatrix} 0.669 \\ 0.331 \end{pmatrix}$$

$$\begin{pmatrix} \Pr(o_4 = O_1) \\ \Pr(o_4 = O_2) \\ \Pr(o_4 = O_3) \end{pmatrix} = B^T (A^T)^3 \pi = \begin{pmatrix} 0.621 \\ 0.266 \\ 0.\cancel{233} \end{pmatrix}$$

0.113

Conditional probabilities

- **If the observation was** (SUNNY, SUNNY, CLOUDY, RAINY, RAINY) = $(O_1, O_1, O_2, O_3, O_3)$ **from day 1 to day 5, what is the distribution of the hidden states on each day?**

$$\Pr\left(q_i | o_1, o_2, o_3, o_4, o_5; \lambda = (\pi, A, B)\right)$$

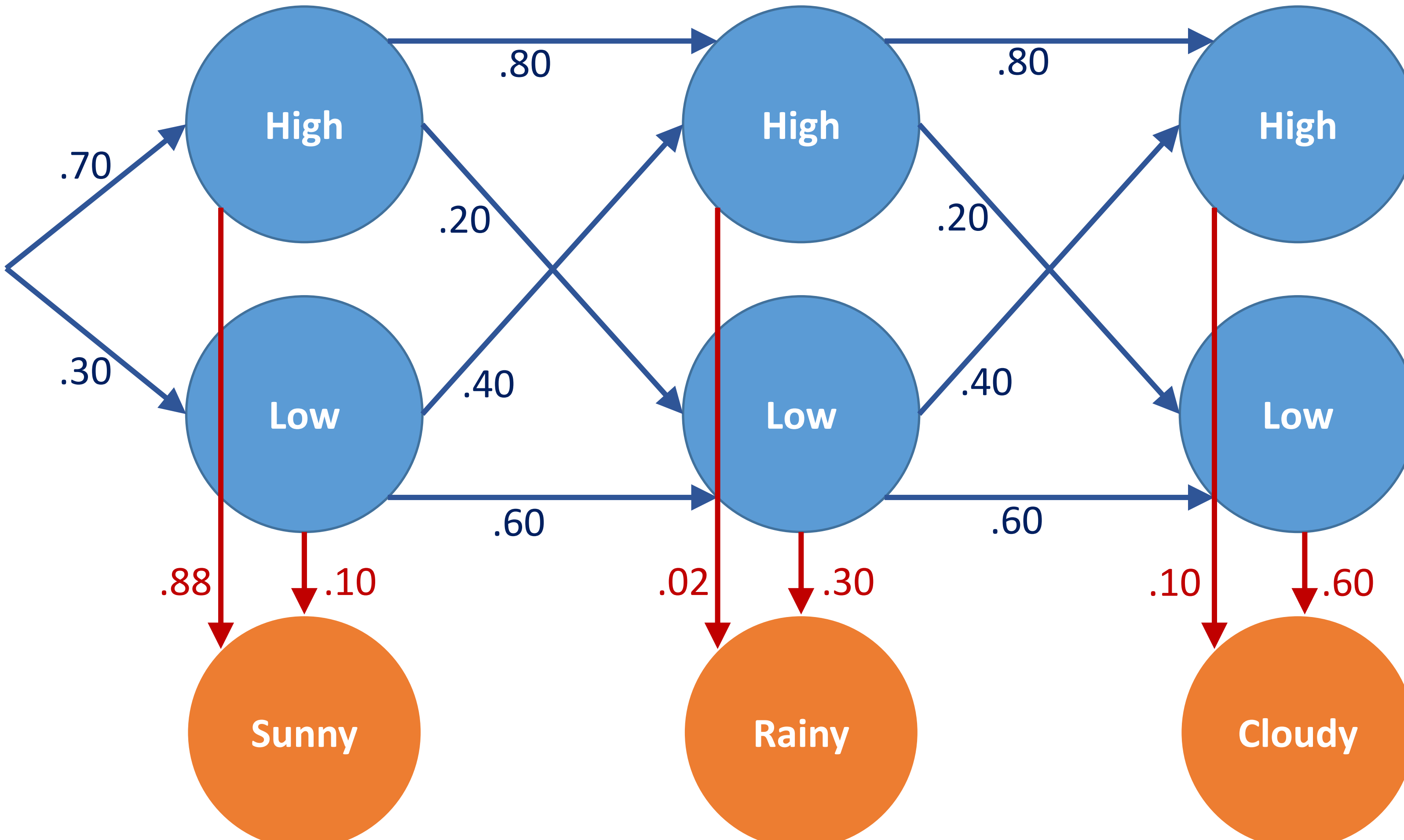
$$(1 \leq i \leq 5)$$

Most likely states

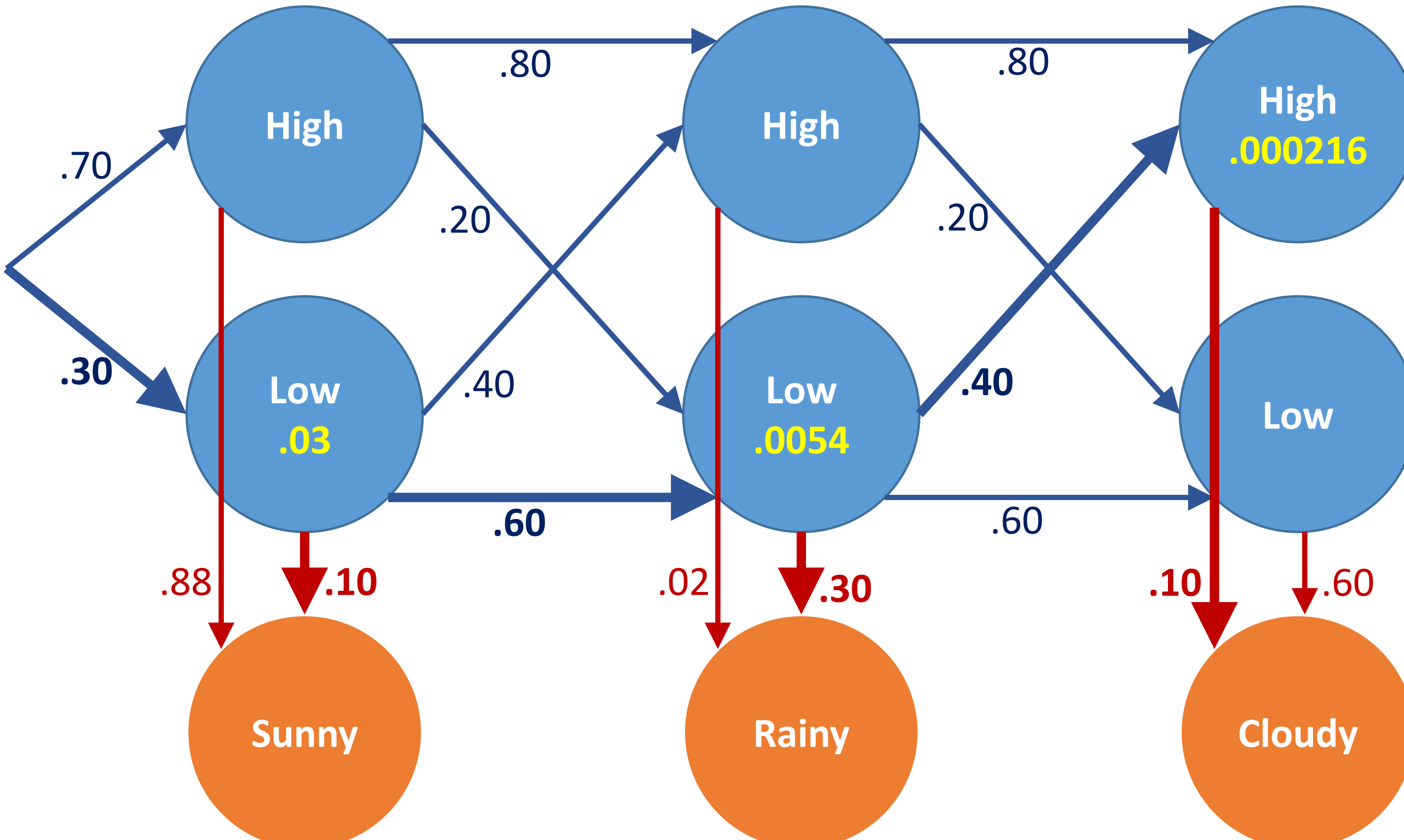
- **If the observation was** (SUNNY, SUNNY, CLOUDY, RAINY, RAINY) = $(O_1, O_1, O_2, O_3, O_3)$ **from day 1 to day 5, what is the most likely combinations of the hidden states?**

$$\arg \max_{(q_1,\ldots,q_5)} \Pr\left(q_1, \ldots, q_5 \middle| o_1, \ldots, o_5; \lambda = (\pi, A, B)\right)$$

Finding MLE states



Enumerating a possible state



Enumerating all possible states

q_{111}	q_2	q_3	Pr(q)	$Pr(o_1 \mid q_1)$	$Pr(o_2 \mid q_2)$	$Pr(o_3 \mid q_3)$	Pr(q,o)
H	H	H	$0.7 \times 0.8 \times 0.8$ .448	.88	.02	.10	.0008
H	H	L	$0.7 \times 0.8 \times 0.2$ .112	.88	.02	.60	.0012
H	L	H	$0.7 \times 0.2 \times 0.4$ .056	.88	.30	.10	.0015
H	**L**	**L**	**.084**	**.88**	**.30**	**.60**	**.0133**
L	H	H	.096	.10	.02	.10	<.0001
L	H	L	.024	.10	.02	.60	<.0001
L	L	H	.072	.10	.30	.10	.0002
L	L	L	.108	.10	.30	.60	.0019

Brute-force calculation of MLE states

- Enumerating all possible combination of states takes $O(2^T)$.

- Is this reasonable fast or too slow? Why?

- Is there a way to reduce the time complexity? How?

Naïve calculation of the conditional probabilities

$$\Pr(q_t = S_i | \boldsymbol{o}; \lambda) = \frac{\Pr(q_t = S_i, \boldsymbol{o}; \lambda)}{\Pr(\boldsymbol{o}; \lambda)}$$

$$= \frac{\sum_{\boldsymbol{q}_{-t}} \Pr(\boldsymbol{o} | \boldsymbol{q}_{-t}, q_t = S_i; \lambda)}{\sum_{x \in \{S_1, S_2\}} \sum_{\boldsymbol{q}_{-t}} \Pr(\boldsymbol{o} | \boldsymbol{q}_{-t}, q_t = x; \lambda)}$$

- **What is the time complexity?**

Naïve calculation of the conditional probabilities

$$\Pr(q_t = S_i | \boldsymbol{o}; \lambda) = \frac{\Pr(q_t = S_i, \boldsymbol{o}; \lambda)}{\Pr(\boldsymbol{o}; \lambda)}$$

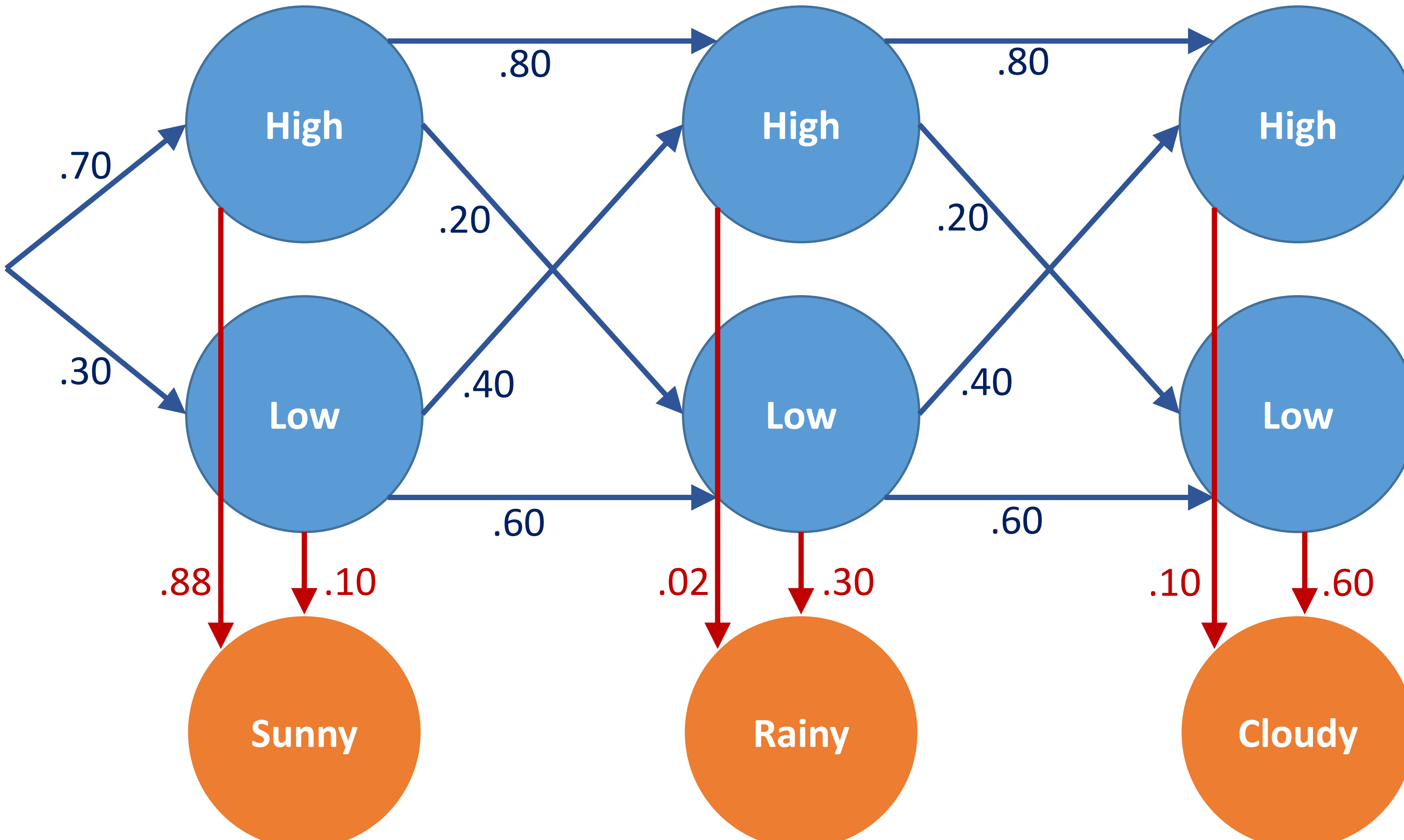
$$= \frac{\sum_{\boldsymbol{q}_{-t}} \Pr(\boldsymbol{o} | \boldsymbol{q}_{-t}, q_t = S_i; \lambda)}{\sum_{x \in \{S_1, S_2\}} \sum_{\boldsymbol{q}_{-t}} \Pr(\boldsymbol{o} | \boldsymbol{q}_{-t}, q_t = x; \lambda)}$$

- **What is the time complexity?**

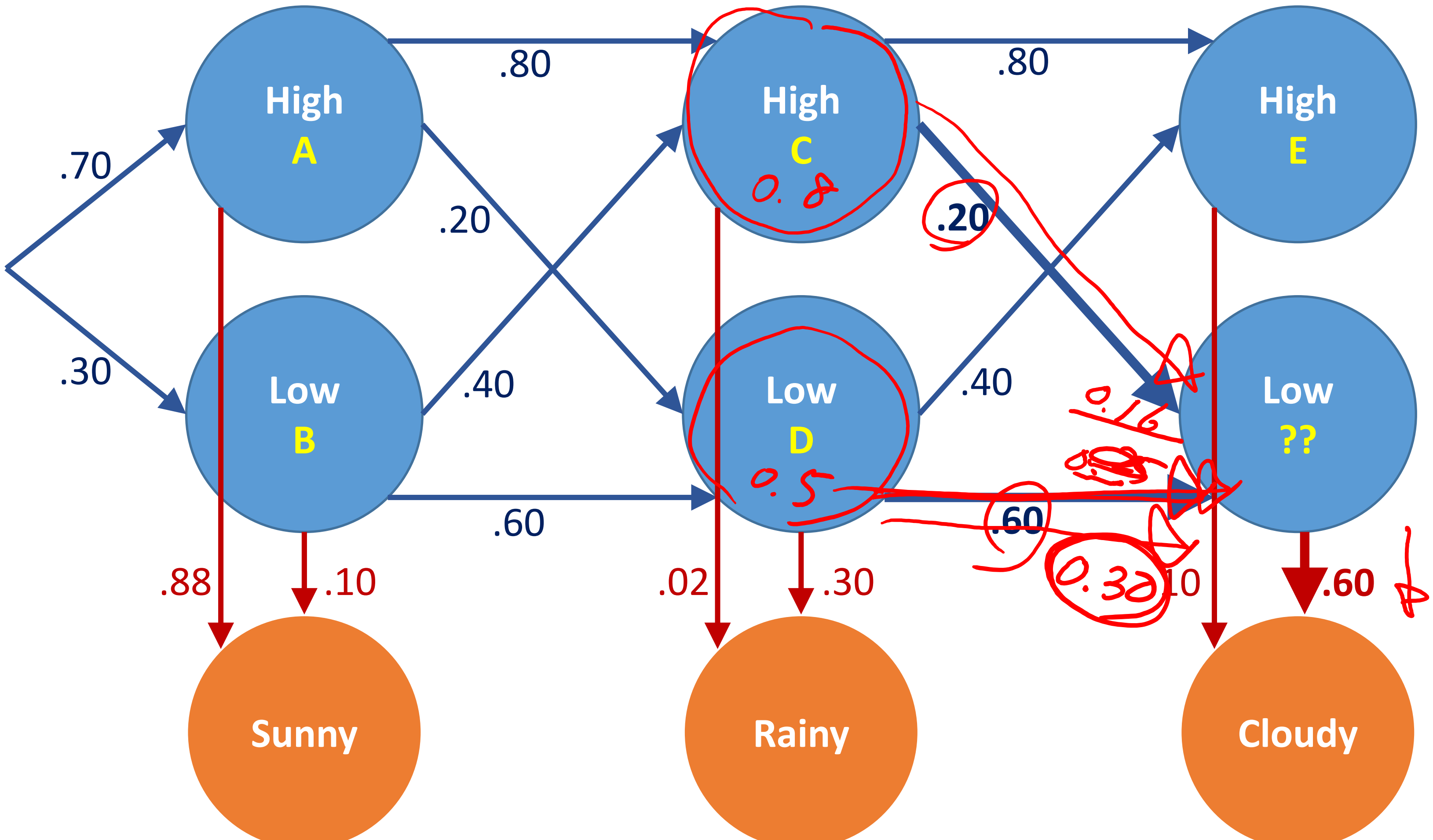
$$f(T) = \Theta(2^T)$$

- **Can we make it faster? How?**

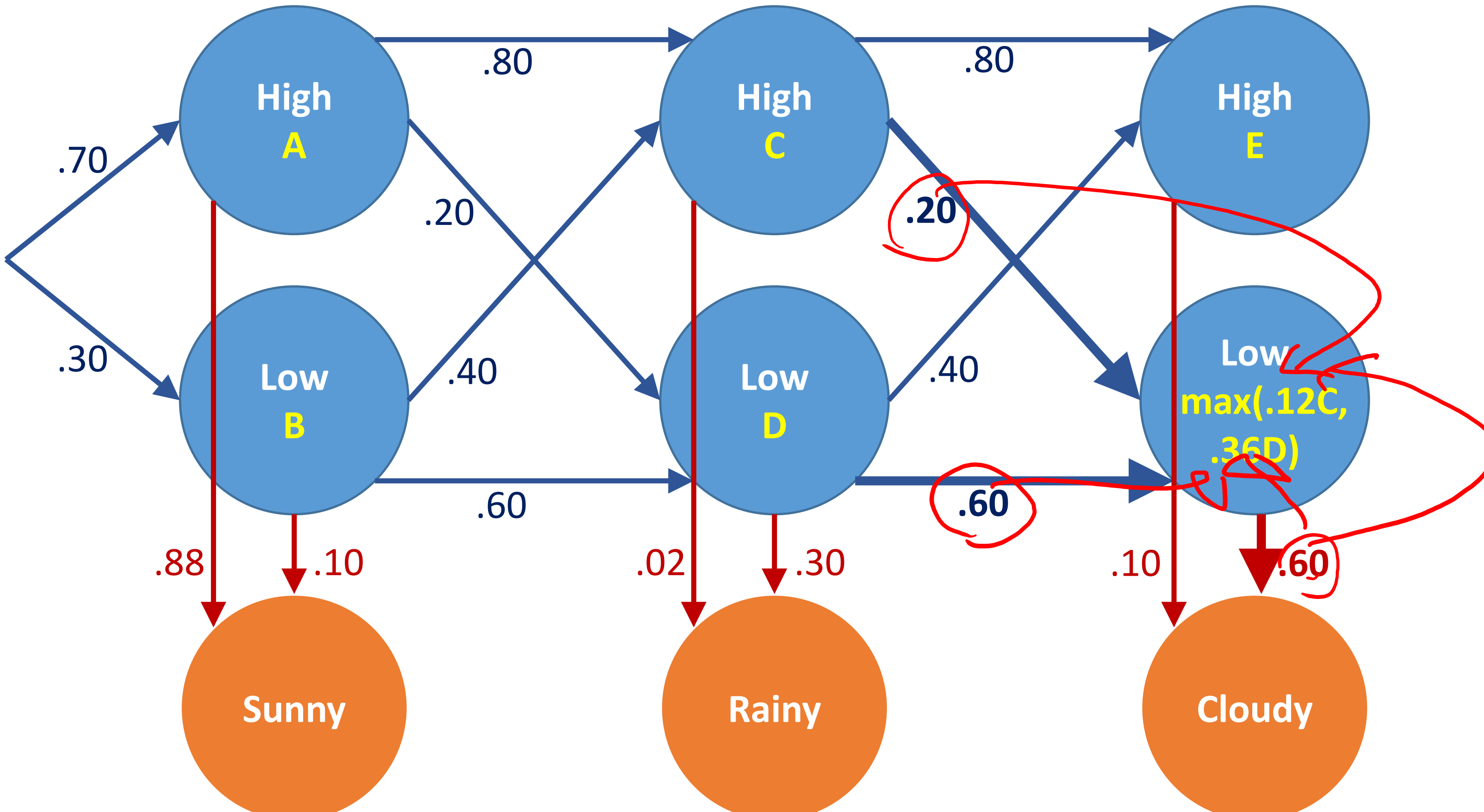
Dynamic programming solution



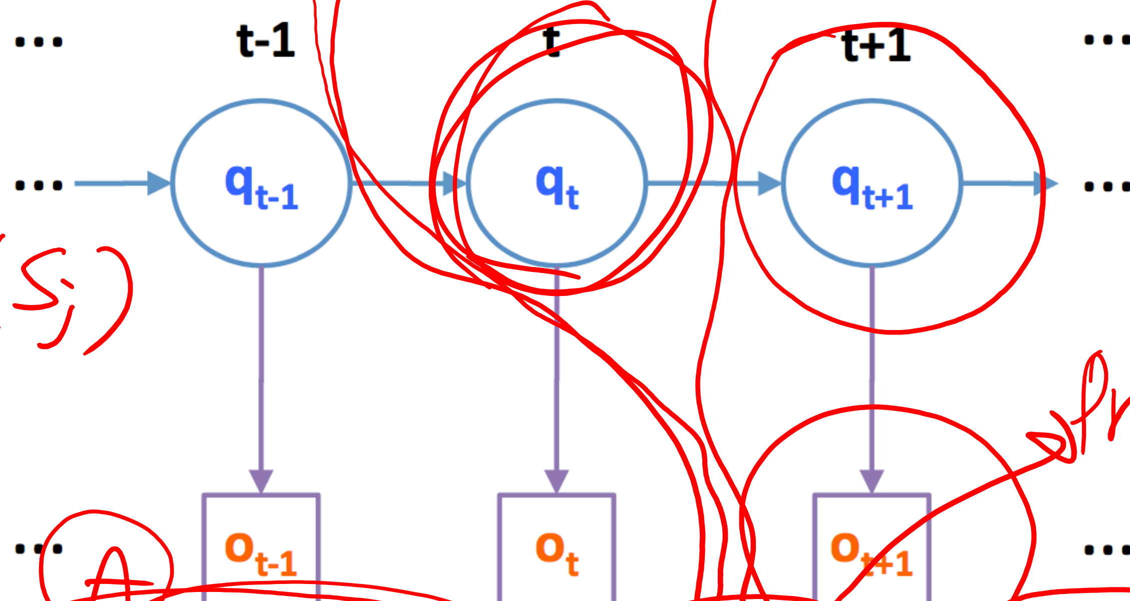
Dynamic programming solution



Dynamic programming solution



Identifying conditional independence



$\delta_{t-1}(S_j)$

$Pr(A,B,C) = Pr(A)\,Pr(B|A)\,Pr(C|A,B)$

$Pr(B|A)$

$Pr(C|B) = Pr(C|A,B)$

A

$$\Pr(q_1,\cdots,q_t,o_1,\cdots,o_t)\Pr(q_{t+1}|q_t)\Pr(o_{t+1}|q_{t+1})$$
$$= \Pr(q_1,\cdots,q_{t+1},o_1,\cdots,o_{t+1})$$

$Pr(A,B,C)$

Viterbi algorithm

- **Calculating MLE**

$$\delta_t(S_i) = \max_{(q_1,\cdots,q_{t-1})} \Pr(q_1,\cdots,q_{t-1},q_t = S_i, o_1,\cdots,o_{t-1},o_t)$$

$$= \max_j \left[\delta_{t-1}(S_j)\Pr(q_t = S_i | q_{t-1} = S_j)\Pr(o_t | q_t = S_i)\right]$$

$$\max_{\boldsymbol{q}} \Pr(\boldsymbol{q},\boldsymbol{o}) = \max_j \delta_T(S_j)$$

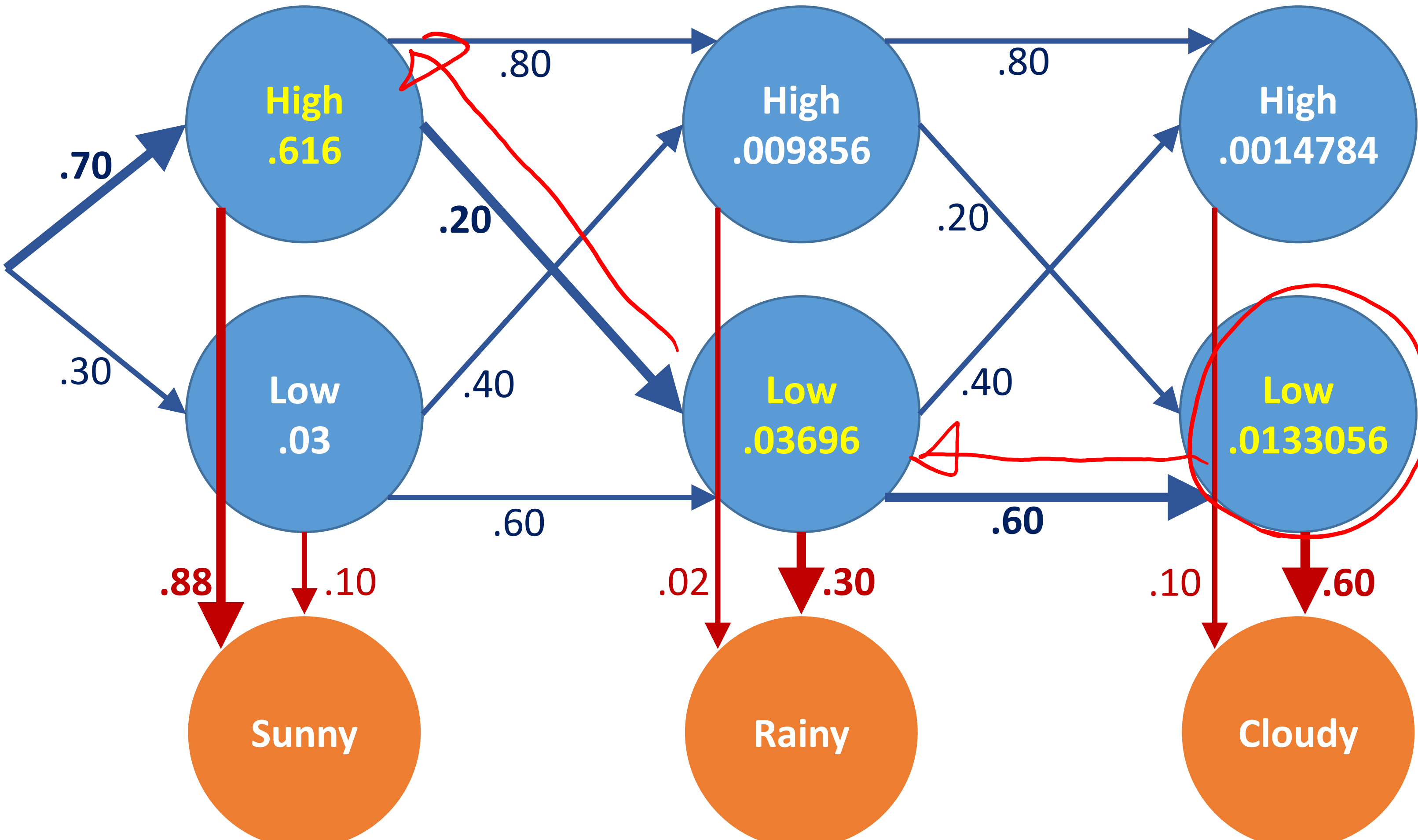
not relevant to j

→ does not affect the result

- **Backtracking Viterbi (MLE) path**

$$\phi_t(S_i) = \underset{j}{\operatorname{argmax}}\left[\delta_{t-1}(S_j)\Pr(q_t = S_i | q_{t-1} = S_j)\right]$$

Results from dynamic programming



A C++ implementation of Viterbi algorithm

```cpp
double viterbiDP(IntegerVector& obs, NumericMatrix& transMtx,
                 NumericMatrix& emisMtx, NumericVector& pi,
                 int t, int s,
                 NumericMatrix& delta, IntegerMatrix& phi) {
  if ( delta(s, t) < 0 ) {
    if ( t == 0 ) delta(s, t) = pi[s];
    else {
      int ns = (int)pi.size();
      for(int i=0; i < ns; ++i) {
        double v = viterbiDP(obs, transMtx, emisMtx, pi, t-1, i, delta, phi)
 * transMtx(i, s);
        if ( delta(s, t) < v ) {
          delta(s, t) = v;
          phi(s, t) = i;
        }
      }
    }
    delta(s, t) *= emisMtx(s, obs[t]);
  }
  return delta(s,t);
}
```

$\times Pr(o_1|s_1)$

ML until $(t-1, j)$

$delta(j, t-1)$

```
// [[Rcpp::export]]
List viterbiHMM(IntegerVector obs, NumericMatrix transMtx, NumericMatrix emisMtx, NumericVector pi) {
  int T = (int)obs.size();
  int ns = (int)pi.size();
  NumericMatrix delta(ns, T);
  IntegerMatrix phi(ns, T);
  std::fill(delta.begin(), delta.end(), -1);

  double ml = -1;
  IntegerVector paths(T);
  for(int i=0; i < ns; ++i) {
    double v = viterbiDP(obs, transMtx, emisMtx, pi, T-1, i, delta, phi);
    if ( ml < v ) {
      ml = v;
      paths[T-1] = i;
    }
  }
  for(int i=T-1; i > 0; --i)
    paths[i-1] = phi(paths[i],i);
  return ( List::create(Named("ML") = ml,
                        Named("path") = paths,
                        Named("delta") = delta,
                        Named("phi") = phi
  ) );
}
```

1

0

0 H

1 BL

Rcpp interface

An Example Result

```r
A <- matrix(c(0.8, 0.2, 0.4, 0.6),2,2,byrow=TRUE)
B <- matrix(c(0.88, 0.10, 0.02, 0.10, 0.60, 0.30),2,3,byrow=TRUE)
pi <- c(0.7,0.3)
obs <- c(0,2,1) # SUNNY, SUNNY, RAINY, RAINY, CLOUDY
viterbiHMM(obs, A, B, pi)
```

```
$ML
[1] 0.0133056

$path
[1] 0 1 1
```

HLL

```
$delta
      [,1]     [,2]      [,3]
[1,] 0.616 0.009856 0.0014784
[2,] 0.030 0.036960 0.0133056

$phi
     [,1] [,2] [,3]
[1,]    0    0    1
[2,]    0    0    1
```

Another example result

```
viterbiHMM(c(0,0,2,2,1), A, B, pi)
```

S S R R C

```
$ML
[1] 0.001686086

$path
[1] 0 0 1 1 1

$delta
      [,1]     [,2]        [,3]         [,4]         [,5]
[1,] 0.616 0.433664 0.006938624 0.0002081587 0.0001873428
[2,] 0.030 0.012320 0.026019840 0.0046835712 0.0016860856

$phi
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    1    1
[2,]    0    0    0    1    1
```

H H L L L

Viterbi Implementation using loop

solve $t=t$ from $t=1,\dots,T-1$

objective

delta(t, i)

not necessary

```
double viterbiLoop(IntegerVector& obs, NumericMatrix& transMtx,
                   NumericMatrix& emisMtx, NumericVector& pi,
                   NumericMatrix& delta, IntegerMatrix& phi) {
  int T = (int)obs.size();
  int ns = (int)pi.size();
  for(int i=0; i < ns; ++i)
    delta(i,0) = pi(i) * emisMtx(i, obs[0]);
  for(int t=1; t < T; ++t) {
    for(int i=0; i < ns; ++i) {
      for(int j=0; j < ns; ++j) {
        double v = delta(j,t-1) * transMtx(j, i);
        if ( v > delta(i,t) ) {
          delta(i,t) = v;
          phi(i,t) = j;
        }
      }
      delta(i,t) *= emisMtx(i, obs[t]);
    }
  }
  double ml = -1;
  for(int i=0; i < ns; ++i) {
    if ( delta(i,T-1) > ml ) ml = delta(i,T-1);
  }
  return ml;
}
```

fills the first column in delta

solve $i=0,\dots,ns-1$

here just for reducing computation

choose the ML path

Hyun Min Kang hmkang@umich.edu

Further thoughts

- When the problem scales to larger T and n, do you see any precision problem in the current implementations of Viterbi algorithm? How can you solve it?

- What is the time complexity of the current Viterbi algorithm?

- If there are only two transition probabilities parameters, diagonal (θ) and off-diagnoal $\left(\frac{1-\theta}{n-1}\right)$, can we reduce the time complexity?

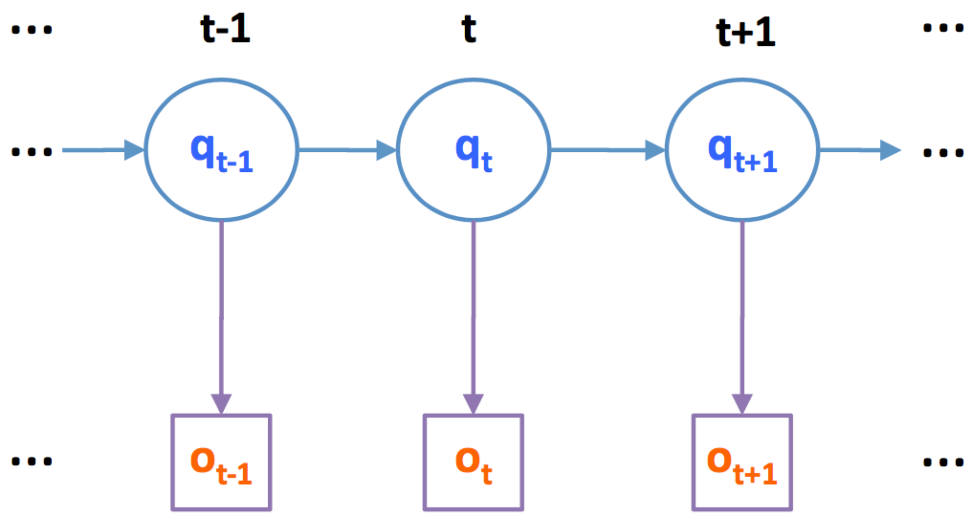
Back to the conditional probability problem

- **If the observation was**
  (SUNNY, SUNNY, CLOUDY, RAINY, RAINY) = (O_1, O_1, O_2, O_3, O_3)
  from day 1 to day 5, what is the distribution of the hidden states on each day?

$$\Pr\left(q_i | o_1, o_2, o_3, o_4, o_5; \lambda = (\pi, A, B)\right)$$

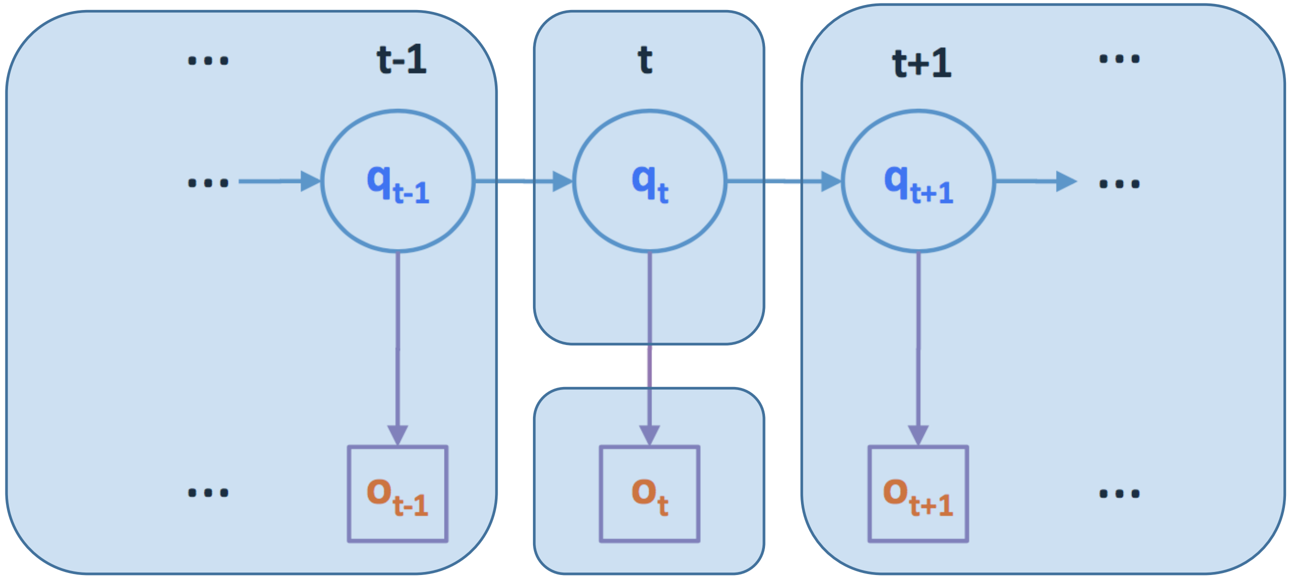
$$(1 \leq i \leq 5)$$

Identifying conditional independence



$$\Pr(q_t = S_i | \boldsymbol{o}; \lambda) = \frac{\Pr(q_t = S_i, \boldsymbol{o}; \lambda)}{\Pr(\boldsymbol{o}; \lambda)} = \frac{\Pr(q_t = S_i, \boldsymbol{o}; \lambda)}{\sum_j \Pr(q_t = S_j, \boldsymbol{o}; \lambda)}$$

Identifying conditional independence



$$\Pr(q_t = S_i | \boldsymbol{o}; \lambda) = \frac{\Pr(q_t{=}S_i, \boldsymbol{o}; \lambda)}{\Pr(\boldsymbol{o}; \lambda)} = \frac{\Pr(q_t{=}S_i, \boldsymbol{o}; \lambda)}{\sum_j \Pr\left(q_t{=}S_j, \boldsymbol{o}; \lambda\right)}$$

Forward and backward probabilities

- **Define** $\boldsymbol{o}_{t-1}^{-} = (o_1, \cdots, o_{t-1})$ $\boldsymbol{o}_{t+1}^{+} = (o_{t+1}, \cdots, o_T)$
- **We need to compute..**

$$\Pr(q_t = S_i | \boldsymbol{o}; \lambda) = \frac{\Pr(q_t = S_i, \boldsymbol{o}; \lambda)}{\Pr(\boldsymbol{o}; \lambda)} = \frac{\Pr(q_t = S_i, \boldsymbol{o}; \lambda)}{\sum_j \Pr(q_t = S_j, \boldsymbol{o}; \lambda)}$$

- **From the conditional independence..**

$$\begin{aligned}
\Pr(q_t, \boldsymbol{o}; \lambda) &= \Pr(q_t, \boldsymbol{o}_{t-1}^{-}, o_t, \boldsymbol{o}_{t+1}^{+}; \lambda) \\
&= \Pr(\boldsymbol{o}_{t+1}^{+} | q_t; \lambda) \Pr(\boldsymbol{o}_{t-1}^{-} | q_t; \lambda) \Pr(o_t | q_t; \lambda) \Pr(q_t | \lambda) \\
&= \Pr(\boldsymbol{o}_{t+1}^{+} | q_t; \lambda) \Pr(\boldsymbol{o}_{t-1}^{-}, o_t, q_t; \lambda) \\
&= \beta_t(q_t) \alpha_t(q_t)
\end{aligned}$$

Forward algorithm for $\alpha_t(q_t)$

- **Key idea : Separate out o_t, q_t to make a recursive formula**

$$(o_t, q_t) \perp \boldsymbol{o_{t-1}^{-}} \mid q_{t-1}$$

$$\alpha_t(S_i) = \Pr(\boldsymbol{o_{t-1}^{-}}, o_t, q_t = S_i\,; \lambda)$$

$$= \sum_j \Pr\left(\boldsymbol{o_{t-1}^{-}}, o_t, q_t = S_i, q_{t-1} = S_j; \lambda\right)$$

$$= \sum_j \Pr\left(\boldsymbol{o_{t-1}^{-}}, q_{t-1} = S_j; \lambda\right) \Pr\left(q_t = S_i | q_{t-1} = S_j; \lambda\right) \Pr(o_t | q_t = S_i; \lambda)$$

$$= \sum_j \alpha_{t-1}(S_j) A_{ji} B_{i o_t}$$

Loopy implementation of forward algorithm

```
void forwardLoop(IntegerVector& obs, NumericMatrix& transMtx,
                 NumericMatrix& emisMtx, NumericVector& pi,
                 NumericMatrix& alpha) {
  int T = (int)obs.size();
  int ns = (int)pi.size();

  for(int i=0; i < ns; ++i)
    alpha(i,0) = pi(i) * emisMtx(i, obs[0]);

  for(int t=1; t < T; ++t) {
    for(int i=0; i < ns; ++i) {
      alpha(i,t) = 0;
      for(int j=0; j < ns; ++j)
        alpha(i,t) += ( alpha(j,t-1) * transMtx(j,i) );
      alpha(i,t) *= emisMtx(i, obs[t]); // why did I do this?
    }
  }
}
```

Backward algorithm for $\beta_t(q_t)$

- **Key idea : Separate out o_{t+1}, q_t to make a recursive formula**

$$o_{t+1} \perp \boldsymbol{o}_{\boldsymbol{t+2}}^{+} \mid q_{t+1}$$

$$\begin{aligned}
\beta_t(S_i) &= \Pr(\boldsymbol{o}_{\boldsymbol{t+1}}^{+} | q_t = S_i \,; \lambda) \\
&= \sum_j \Pr\left(\boldsymbol{o}_{\boldsymbol{t+2}}^{+}, o_{t+1}, q_{t+1} = S_j | q_t = S_i; \lambda\right) \\
&= \sum_j \Pr\left(\boldsymbol{o}_{\boldsymbol{t+2}}^{+} | q_{t+1} = S_j; \lambda\right) \Pr\left(q_{t+1} = S_j | q_t = S_i; \lambda\right) \Pr\left(o_{t+1} | q_{t+1} = S_j; \lambda\right) \\
&= \sum_j \beta_{t+1}(S_j) A_{ij} B_{j o_{t+1}}
\end{aligned}$$

Loopy implementation of backward algorithm

```
void backwardLoop(IntegerVector& obs, NumericMatrix& transMtx,
                  NumericMatrix& emisMtx, NumericVector& pi,
                  NumericMatrix& beta) {
  int T = (int)obs.size();
  int ns = (int)pi.size();

  for(int i=0; i < ns; ++i)
    beta(i,T-1) = 1;

  for(int t=T-2; t >=0; --t) {
    for(int i=0; i < ns; ++i) {
      beta(i,t) = 0;
      for(int j=0; j < ns; ++j)
        beta(i,t) += ( beta(j,t+1) * transMtx(i,j) * emisMtx(j, obs[t+1]) );
    }
  }
}
```

Putting two algorithms together

$$\Pr(q_t, \boldsymbol{o}; \lambda) = \beta_t(q_t)\alpha_t(q_t)$$

$$\Pr(q_t = S_i | \boldsymbol{o}; \lambda) = \frac{\Pr(q_t = S_i, \boldsymbol{o}; \lambda)}{\sum_j \Pr\left(q_t = S_j, \boldsymbol{o}; \lambda\right)}$$

$$= \frac{\beta_t(S_i)\alpha_t(S_i)}{\sum_j \beta_t\left(S_j\right)\alpha_t(S_j)}$$

```cpp
// [[Rcpp::export]]
NumericMatrix forwardBackwardHMM(IntegerVector obs, NumericMatrix transMtx,
NumericMatrix emisMtx, NumericVector pi) {
  int T = (int)obs.size();
  int ns = (int)pi.size();
  NumericMatrix alpha(ns, T);
  NumericMatrix beta(ns, T);

  forwardLoop(obs, transMtx, emisMtx, pi, alpha);
  backwardLoop(obs, transMtx, emisMtx, pi, beta);
  NumericMatrix condProb(ns, T);

  for(int t=0; t < T; ++t) {
    double sum = 0;

    for(int i=0; i < ns; ++i)
      sum += ( alpha(i,t) * beta(i,t) );
    for(int i=0; i < ns; ++i)
      condProb(i,t) = alpha(i,t) * beta(i,t) / sum;
  }
  return condProb;
}
```

Running examples

```r
A <- matrix(c(0.8, 0.2, 0.4, 0.6),2,2,byrow=TRUE)
B <- matrix(c(0.88, 0.10, 0.02, 0.10, 0.60, 0.30),2,3,byrow=TRUE)
pi <- c(0.7,0.3)
forwardBackwardHMM(c(0,2,1), A, B, pi)
```

```
         [,1]      [,2]     [,3]
[1,] 0.883564 0.1064799 0.131944
[2,] 0.116436 0.8935201 0.868056
```

```r
forwardBackwardHMM(c(0,0,2,2,1), A, B, pi)
```

```
           [,1]      [,2]       [,3]      [,4]      [,5]
[1,] 0.96782013 0.9189246 0.08374666 0.0297711 0.1089313
[2,] 0.03217987 0.0810754 0.91625334 0.9702289 0.8910687
```

(Same) further thoughts

- When the problem scales to larger T and n, do you see any precision problem in the current implementations of Viterbi algorithm? How can you solve it?
- What is the time complexity of the current Viterbi algorithm?
- If there are only two transition probabilities parameters, diagonal (θ) and off-diagnoal $\left(\frac{1-\theta}{n-1}\right)$, can we reduce the time complexity?

Summary

- **Graphical models**
  - A great tool to represent complex relationship between random variables
- **Hidden Markov model**
  - The most widely used graphical models.
  - Current state only depends on previous states
- **Viterbi algorithm**
  - DP algorithm similar to optimal path finding to get MLEs
- **Forward-backward algorithm**
  - DP algorithm to calculate conditional probabilities of hidden states given observations