隐马科夫链

七月在线 加号
微博：@翻滚吧_加号

马科夫链

马尔可夫链，因安德烈·马尔可夫（A.A.Markov，1856 - 1922）得名，
是指数学中具有马尔可夫性质的离散事件随机过程。
在给定当前知识或信息的情况下，过去（即当前以前的历史状态）
对于预测将来（即当前以后的未来状态）是无关的。

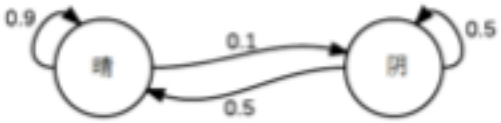
马科夫链

每个状态的转移只依赖于之前的n个状态，这个过程被称为1个n阶的模型，
其中n是影响转移状态的数目。最简单的马尔科夫过程就是一阶过程，
每一个状态的转移只依赖于其之前的那一个状态。
用数学表达式表示就是下面的样子：

$$P(X_{n+1} = x | X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = P(X_{n+1} = x | X_n = x_n).$$

马科夫链

- 假设天气服从马尔可夫链



- 转移矩阵 $P = \begin{pmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{pmatrix}$
- 从今天(晴或阴)开始，在遥远未来的某天，晴阴的概率分布是什么？

$$q = \lim_{n\to\infty} (1 \quad 0) P^n = (0.833\ 0.167)$$

马科夫链

- $P^{\infty} = \begin{pmatrix} 0.833 & 0.167 \\ 0.833 & 0.167 \end{pmatrix}$
- $(1\ \ 0)P^{\infty} = (0\ \ 1)P^{\infty}$
- 不管今天晴阴，很多天之后的晴阴分布收敛到一个固定分布
- 这个固定分布叫稳态分布
- 很久的未来，每一天天气都是 $q = (0.833\ 0.167)$ 的一个样本点

马科夫链

至此，我们就为上面的一阶马尔科夫过程定义了以下三个部分：

1. 状态：晴天、阴天。
2. 初始向量：定义系统在时间为0的时候的状态的概率。
3. 状态转移矩阵：每种天气转换的概率。所有的能被这样描述的系统都是一个马尔科夫过程。

马科夫链的缺陷

很明显，前后关系的缺失，带来了信息的缺失：

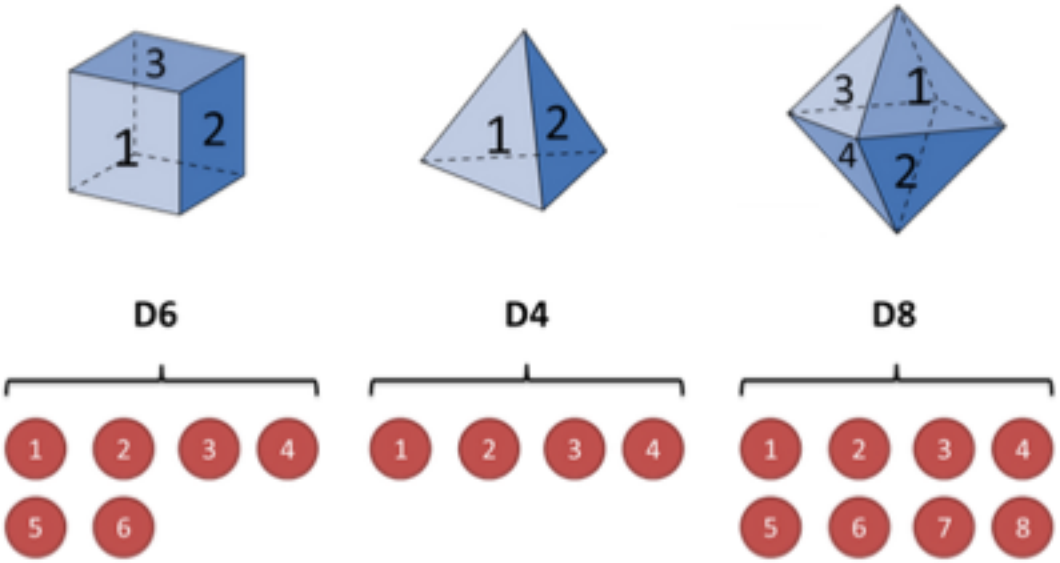
比如我们的股市，如果只是观测市场，我们只能知道当天的价格、成交量等信息，
但是并不知道当前股市处于什么样的状态（牛市、熊市、震荡、反弹等等），
在这种情况下我们有两个状态集合，一个可以观察到的状态集合
（股市价格成交量状态等）和一个隐藏的状态集合（股市状况）。

我们希望能找到一个算法可以根据股市价格成交量状况和马尔科夫假设来预测股市的状况。
在上面的这些情况下，可以观察到的状态序列和隐藏的状态序列是概率相关的。
于是我们可以将这种类型的过程建模为有一个隐藏的马尔科夫过程和
一个与这个隐藏马尔科夫过程概率相关的并且可以观察到的状态集合，就是隐马尔可夫模型。

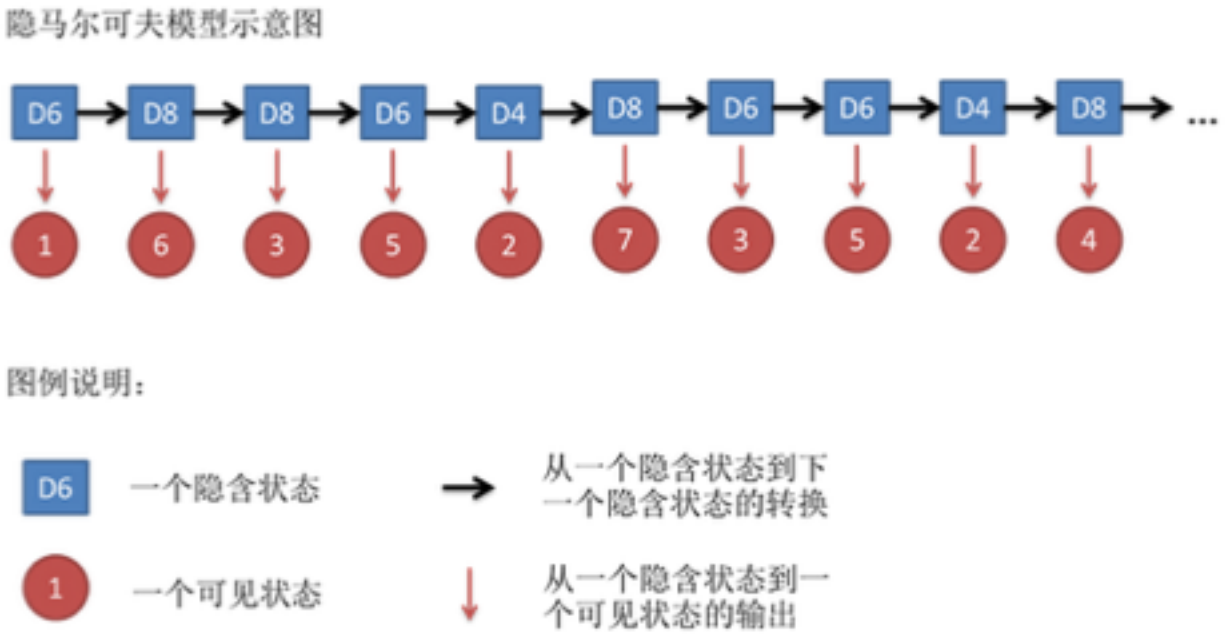
隐马科夫链

隐马尔可夫模型(Hidden Markov Model) 是一种统计模型，
用来描述一个含有隐含未知参数的马尔可夫过程。
其难点是从可观察的参数中确定该过程的隐含参数，
然后利用这些参数来作进一步的分析。

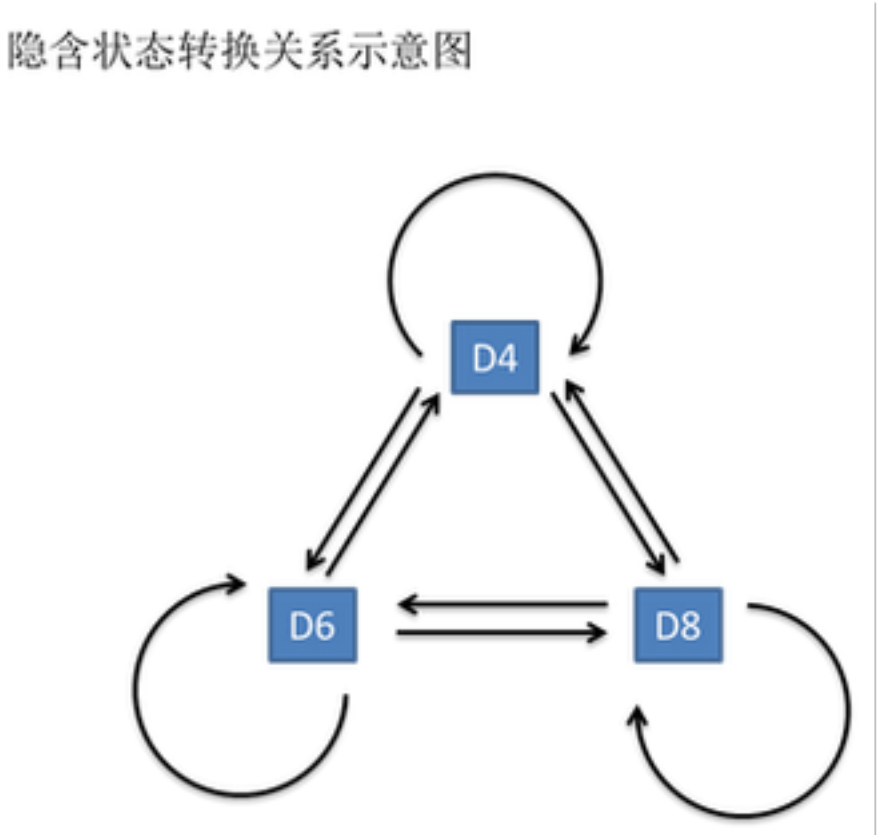
隐马科夫链



隐马科夫链


隐马尔可夫模型示意图
D6
D8
D8
D6
D4
D8
D6
D6
D4
D8
...
1
6
3
5
2
7
3
5
2
4
图例说明：
D6
一个隐含状态
从一个隐含状态到下一个隐含状态的转换
1
一个可见状态
从一个隐含状态到一个可见状态的输出

隐马科夫链



隐马科夫链三大问题

1. 知道骰子有几种（隐含状态数量），每种骰子是什么（转换概率），
   根据掷骰子掷出的结果（可见状态链），我想知道每次掷出来的都是哪种骰子（隐含状态链）。
2. 还是知道骰子有几种（隐含状态数量），每种骰子是什么（转换概率），
   根据掷骰子掷出的结果（可见状态链），我想知道掷出这个结果的概率。
3. 知道骰子有几种（隐含状态数量），不知道每种骰子是什么（转换概率），
   观测到很多次掷骰子的结果（可见状态链），我想反推出每种骰子是什么（转换概率）。

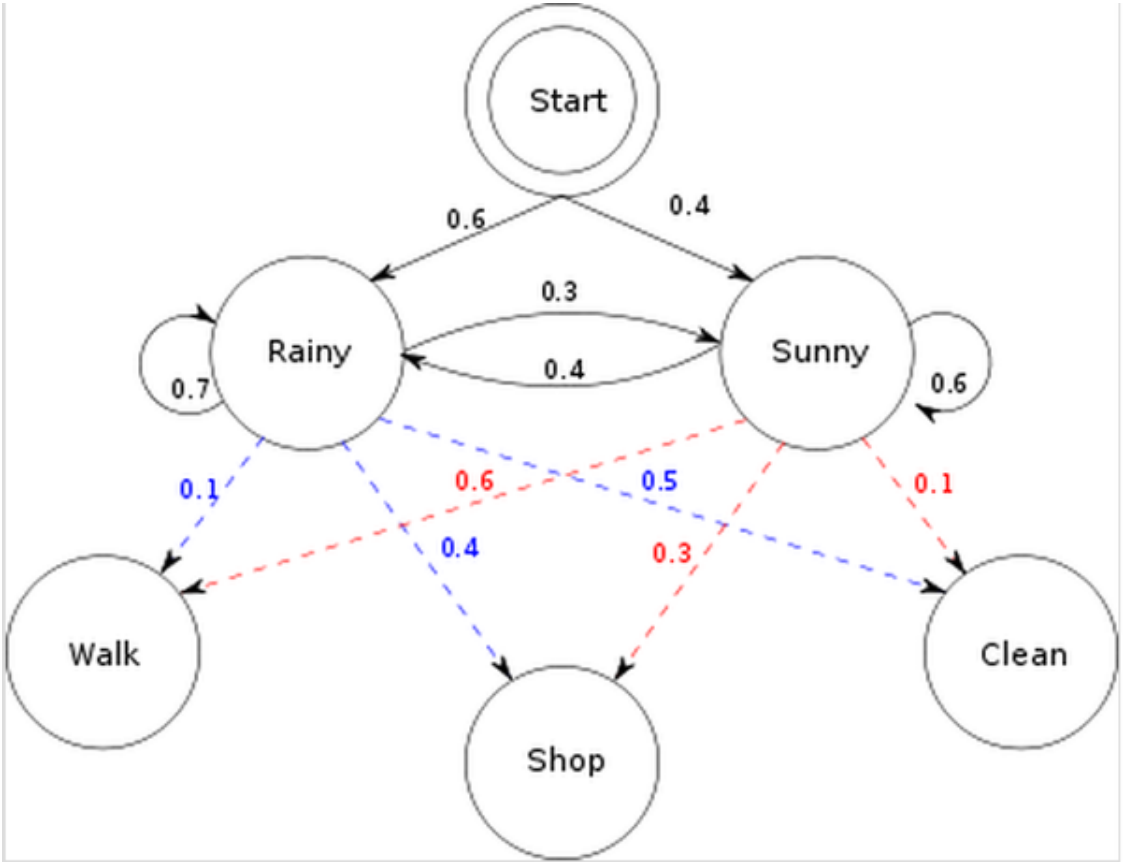
隐马科夫链三大问题

Evaluation Given the observation sequence $O = O_1 O_2 ... O_T$ and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, i.e., the probability of the observation sequence given the model

Recognition Given the observation sequence $O = O_1 O_2 ... O_T$ and a model $\lambda = (A, B, \pi)$, how do we choose a corresponding state sequence $Q = q_1 q_2 ... q_T$ which is optimal in some sense, i.e., best explains the observations

Training Given the observation sequence $O = O_1 O_2 ... O_T$, how do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$

隐马科夫链例子



课堂问答：

初始概率分布 π：
状态转移矩阵 A：
观测量的概率分布 B：

同时有两个状态：
三种可能的观测值：

隐马科夫链例子

问题1，已知整个模型，我观测到连续三天做的事情是：**散步，购物，收拾**。
那么，根据模型，计算产生这些行为的概率是多少。

问题2，同样知晓这个模型，同样是这三件事，我想猜，这三天的天气是怎么样的。

问题3，最复杂的，我只知道这三天做了这三件事儿，而其他什么信息都没有。
我得建立一个模型，晴雨转换概率，第一天天气情况的概率分布，
根据天气情况选择做某事的概率分布。

隐马科夫链解法

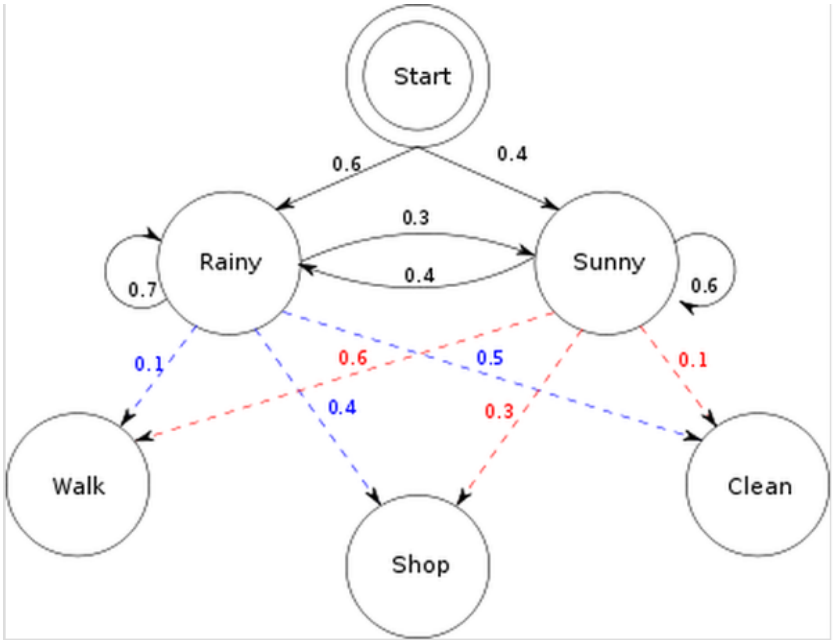
问题一，Forward Algorithm，向前算法，或者 Backward Algo，向后算法。

问题二，Viterbi Algo，维特比算法。

问题三，Baum-Welch Algo，鲍姆-韦尔奇算法

隐马科夫链解法

问题1的解决1：遍历算法



隐马科夫链解法：遍历算法

- We need $P(O|\lambda)$, i.e., the probability of the observation sequence $O = O_1 O_2 ... O_T$ given the model λ
- So we can enumerate every possible state sequence $Q = q_1 q_2 ... q_T$
- For a sample sequence Q

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_t,\lambda) = \prod_{t=1}^{T} b_{q_t O_t}$$

- The probability of such a state sequence Q is

$$P(Q|\lambda) = P(q_1) \prod_{t=2}^{T} P(q_t|q_{t-1}) = \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1} q_t}$$

隐马科夫链解法：遍历算法

- Therefore the joint probability

$$P(O,Q|\lambda) = P(Q|\lambda)P(O|Q,\lambda) = \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1}q_t} \prod_{t=1}^{T} b_{q_t O_t}$$

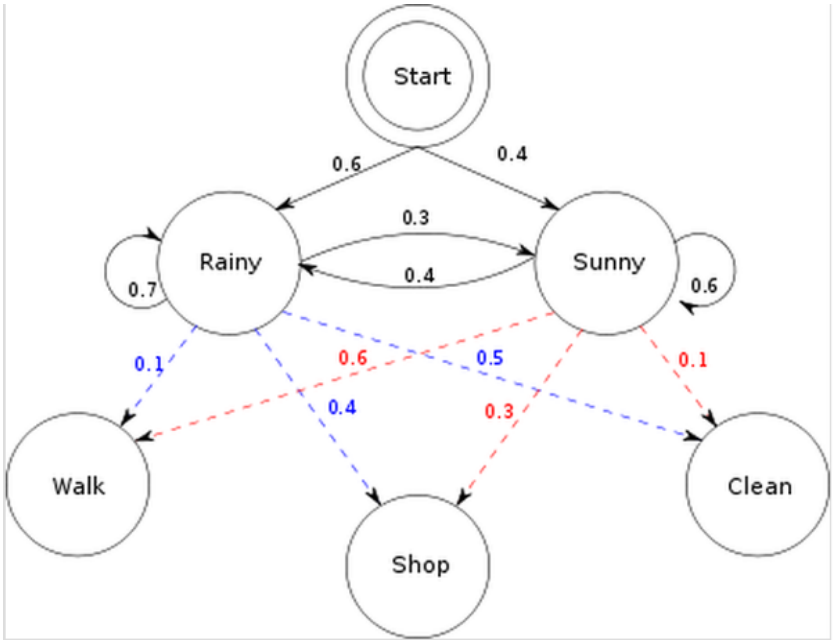
- By considering all possible state sequences

$$P(O|\lambda) = \sum_{Q} \pi_{q_1} b_{q_1 O_1} \prod_{t=2}^{T} a_{q_{t-1}q_t} b_{q_t O_t}$$

- Problem: order of $2TN^T$ calculations
  - N^T possible state sequences
  - about $2T$ calculations for each sequence

隐马科夫链解法

问题1的解决2：向前算法



隐马科夫链解法：前向算法

- We define a forward variable $\alpha_j(t)$ as the probability of the partial observation seq. until time t, with state S_j at time t

$$\alpha_j(t) = P(O_1 O_2 ... O_t, q_t = S_j|\lambda)$$

- This can be computed inductively

$$\alpha_j(1) = \pi_j b_{jO_1} \qquad 1 \le j \le N$$

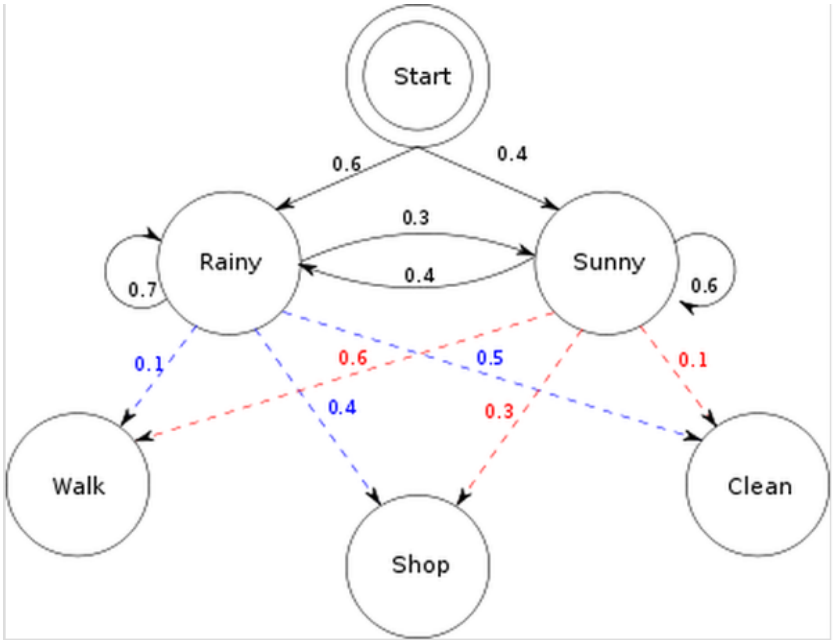
$$\alpha_j(t+1) = (\sum_{i=1}^{N} \alpha_i(t) a_{ij}) b_{jO_{t+1}} \qquad 1 \le t \le T-1$$

- Then with $N^2 T$ operations:

$$P(O|\lambda) = \sum_{i=1}^{N} P(O, q_T = S_i|\lambda) = \sum_{i=1}^{N} \alpha_i(T)$$

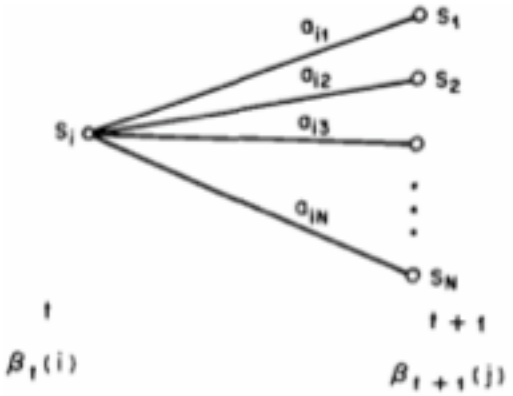
隐马科夫链解法：后向算法

问题1的解决3：向后算法



隐马科夫链解法：后向算法

Figure: Operations for computing the backward variable $\beta_i(t)$



- We define a backward variable $\beta_i(t)$ as the probability of the partial observation seq. after time t, given state S_i at time t

$$\beta_i(t) = P(O_{t+1}Ot + 2...O_T|q_t = S_i, \lambda)$$

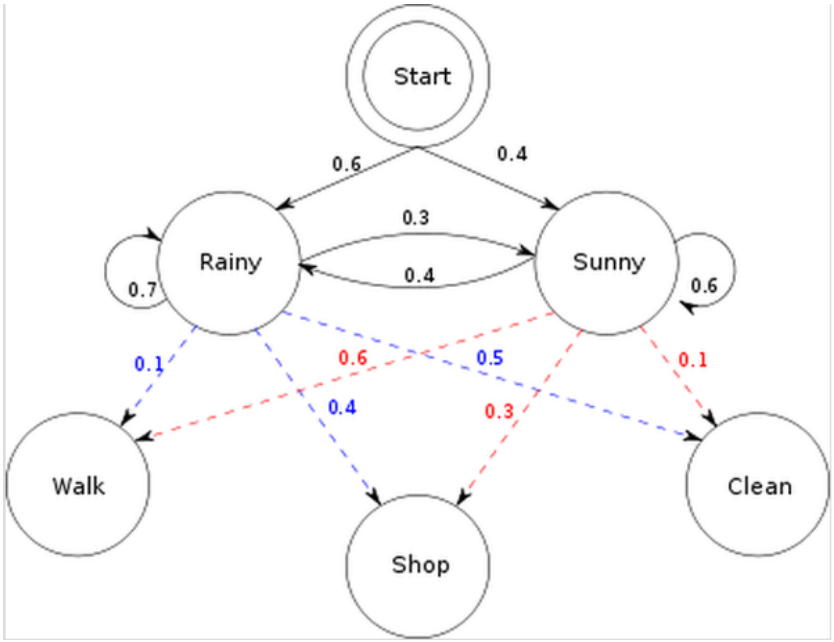
- This can be computed inductively as well

$$\beta_i(T) = 1 \qquad 1 \leq i \leq N$$

$$\beta_i(t-1) = \sum_{j=1}^{N} a_{ij} b_{jO_t} \beta_j(t) \qquad 2 \leq t \leq T$$

隐马科夫链解法：Viterbi算法

问题2的解决：Viterbi算法



隐马科夫链解法：Viterbi算法

- Finding the best single sequence means computing $\text{argmax}_Q P(Q|O,\lambda)$, equivalent to $\text{argmax}_Q P(Q,O|\lambda)$
- The Viterbi algorithm (dynamic programming) defines $\delta_j(t)$, i.e., the highest probability of a single path of length t which accounts for the observations and ends in state S_j

$$\delta_j(t) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1 q_2 \ldots q_t = j, O_1 O_2 \ldots O_t|\lambda)$$

- By induction

$$\begin{aligned} \delta_j(1) &= \pi_j b_{jO_1} & 1 \le j \le N \\ \delta_j(t+1) &= \left(\max_i \delta_i(t) a_{ij}\right) b_{jO_{t+1}} & 1 \le t \le T-1 \end{aligned}$$

- With backtracking (keeping the maximizing argument for each t and j) we find the optimal solution

隐马科夫链解法：Baum-Welch算法

- There is no known way to analytically solve for the model which maximizes the probability of the observation sequence
- We can choose $\lambda = (A, B, \pi)$ which locally maximizes $P(O|\lambda)$
  - gradient techniques
  - Baum-Welch reestimation (equivalent to EM)
- We need to define $\xi_{ij}(t)$, i.e., the probability of being in state S_i at time t and in state S_j at time $t+1$

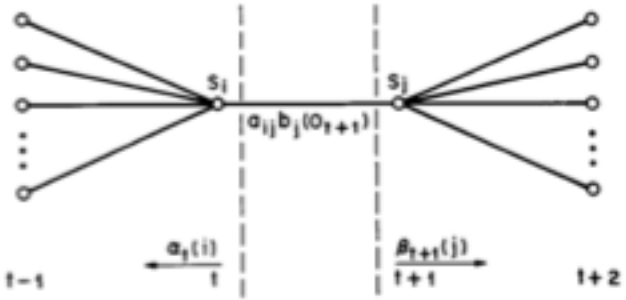
$$\xi_{ij}(t) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

$$\xi_{ij}(t) = \frac{\alpha_i(t) a_{ij} b_{jO_{t+1}} \beta_j(t+1)}{P(O|\lambda)} =$$

$$= \frac{\alpha_i(t) a_{ij} b_{jO_{t+1}} \beta_j(t+1)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i(t) a_{ij} b_{jO_{t+1}} \beta_j(t+1)}$$

隐马科夫链解法：Baum-Welch算法

Figure: Operations for computing the $\xi_{ij}(t)$



- Recall that $\gamma_i(t)$ is a probability of state S_i at time t, hence

$$\gamma_i(t) = \sum_{j=1}^{N} \xi_{ij}(t)$$

- Now if we sum over the time index t
  - $\sum_{t=1}^{T-1} \gamma_i(t)$ = expected number of times that S_i is visited*
    = expected number of transitions from state S_i
  - $\sum_{t=1}^{T-1} \xi_{ij}(t)$ = expected number of transitions from S_i to S_j

隐马科夫链解法：Baum-Welch算法

- Reestimation formulas

$$\bar{\pi}_i = \gamma_i(1) \qquad \bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \qquad \bar{b}_{jk} = \frac{\sum_{O_t=v_k} \gamma_j(t)}{\sum_{t=1}^{T} \gamma_j(t)}$$

- Baum et al. proved that if current model is $\lambda = (A, B, \pi)$ and we use the above to compute $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ then either
  - $\bar{\lambda} = \lambda$ – we are in a critical point of the likelihood function
  - $P(O|\bar{\lambda}) > P(O|\lambda)$ – model $\bar{\lambda}$ is more likely
- If we iteratively reestimate the parameters we obtain a maximum likelihood estimate of the HMM
- Unfortunately this finds a local maximum and the surface can be very complex

隐马科夫链应用：词性标注

What is it?

Automatically assigning a part of speech to the words in a sentence:
e.g. the/DET cat/N sat/V on/P the/DET mat/N ./.

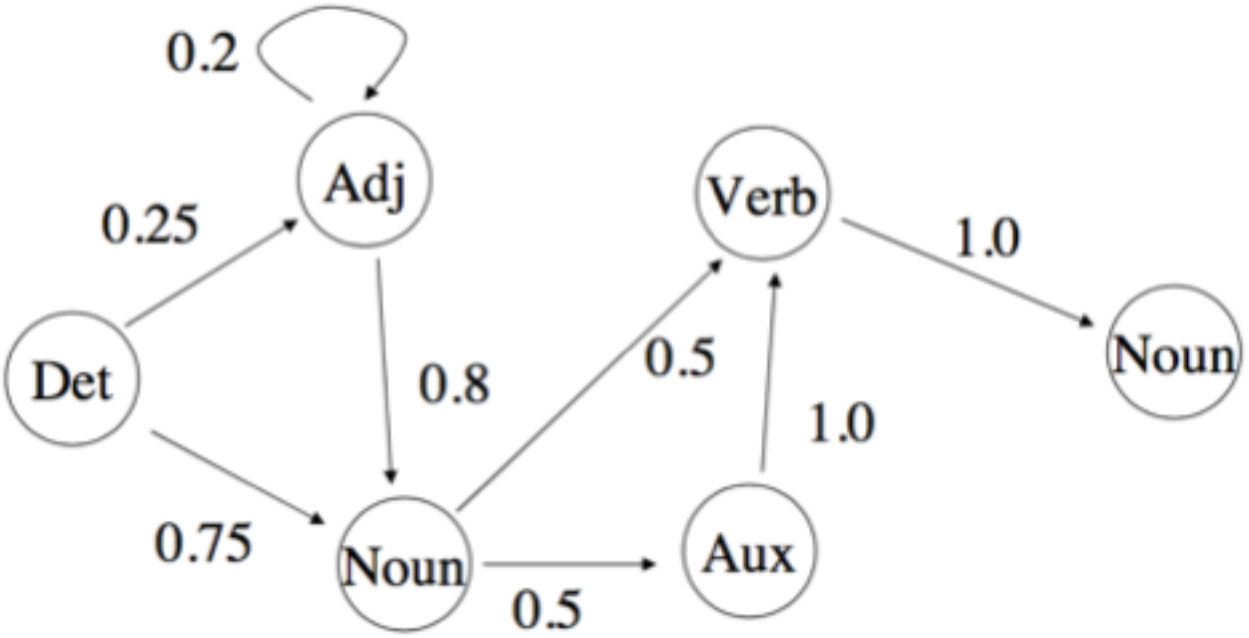
Why?

- useful preprocessing step for later parsing, since it reduces ambiguity: How the time/N flies vs. I want you to time/V flies
- enrich a corpus with useful information: 'find all occurrences of 'time' as a verb, followed by a noun'
- can help guess categories of unknown words:

'Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

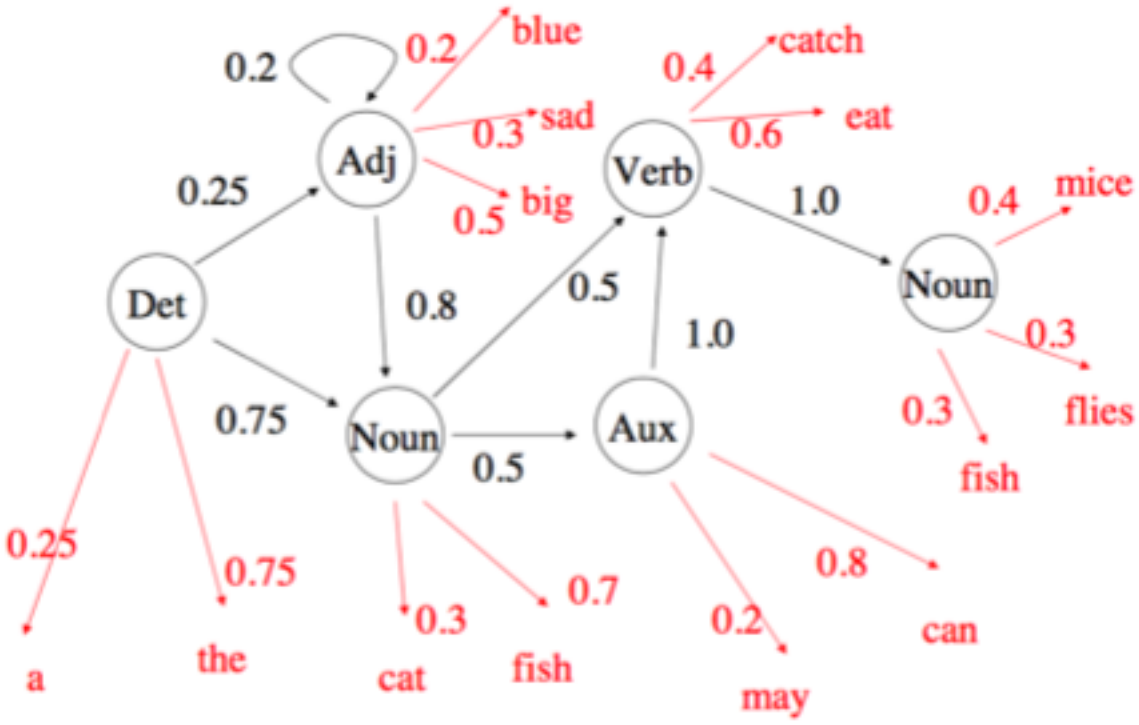
隐马科夫链应用：词性标注

A Markov Model is essentially a finite state machine with probabilities on the transitions:



隐马科夫链应用：词性标注

An HMM is a Markov model which also emits symbols when in a particular state: each symbol having a probability of being emitted:



隐马科夫链应用：词性标注

We want to find $P(t_1,\ldots,t_n \mid w_1,\ldots,w_n)$. **Bayes' Theorem** tells us that this is the same as:

$$P(t_1 \ldots t_N | w_1 \ldots w_N) = \frac{P(t_1 \ldots t_N) \times P(w_1 \ldots w_N | t_1 \ldots t_N)}{P(w_1 \ldots w_N)}$$

We want to find the sequence of ts that give us the highest value in this equation. Note $P(w_1 \ldots w_n)$ will be the same for all sequences if the words are given so we can ignore it. But the top line is still too much to calculate directly, since we would never have enough data to estimate the probabilities from.
The chain rule:

$$\begin{aligned}
P(A, B) &= P(A|B) \times P(B) \\
P(A, B, C) &= P(A|B, C) \times P(B|C) \times P(C) \\
P(A, B, C, D) &= P(A|B, C, D) \times P(B|C, D) \times P(C|D) \times P(D) \\
P(C_1 \ldots C_n) &= P(C_1|C_2 \ldots C_n) \times P(C_2|C_3 \ldots C_n) \times \cdots \times P(C_n)
\end{aligned}$$

隐马科夫链应用：词性标注

Tag transition distribution:

$$P(t_1 \dots t_N) = \prod_{i=1}^{N} P(t_i | \dots t_{i-1} \dots t_1) \approx \prod_{i=1}^{N} P(t_i | t_{i-1})$$

Word emission distribution:

$$P(w_1 \dots w_N | t_1 \dots t_N) = \prod_{i=1}^{N} P(w_i | w_{i-1} \dots w_1, t_1 \dots t_N)$$

$$\approx \prod_{i=1}^{N} P(w_i | t_i)$$

NB: we invent a start (and end) of sentence marker or dummy word (e.g. **start**, **end**) so that t_{i-1} exists even where i=1.

Example: Part-of-Speech Tagging

- $t_1 \ldots t_N$ tags correspond to states (z_n),
- $w_1 \ldots w_N$ words correspond to observations (x_n).
- by counting words and tags we can calculate MLEs for the model's parameters

$$A_{ij} = p(t_{n+1} = j | t_n = i) = \frac{\text{Count}(t = i, t = j)}{\text{Count}(t = i)},$$

$$p(w_n = \text{word} | t_n = k) = \frac{\text{Count}(w = \text{word}, t = k)}{\text{Count}(t = k)},$$

There are various suitable corpora for this, e.g. the Penn Treebank (UPenn) and the British National Corpus (Oxford Text Archive).

隐马科夫链应用：词性标注

Counts from an corpus annotated with part-of-speech tags (from BNC):

First tag	AT	BEZ	IN	NN	VB	PERIOD
AT	0	0	0	48636	0	0
BEZ	1973	0	426	187	0	38
IN	43322	0	1352	17314	0	185
NN	1067	3720	42470	11773	614	21392
VB	6072	42	4758	1476	129	1522
PERIOD	8016	75	4656	1329	954	0

(AT=article, BEZ=is/was, IN=prep, NN=sing noun, VB=base form of verb...)

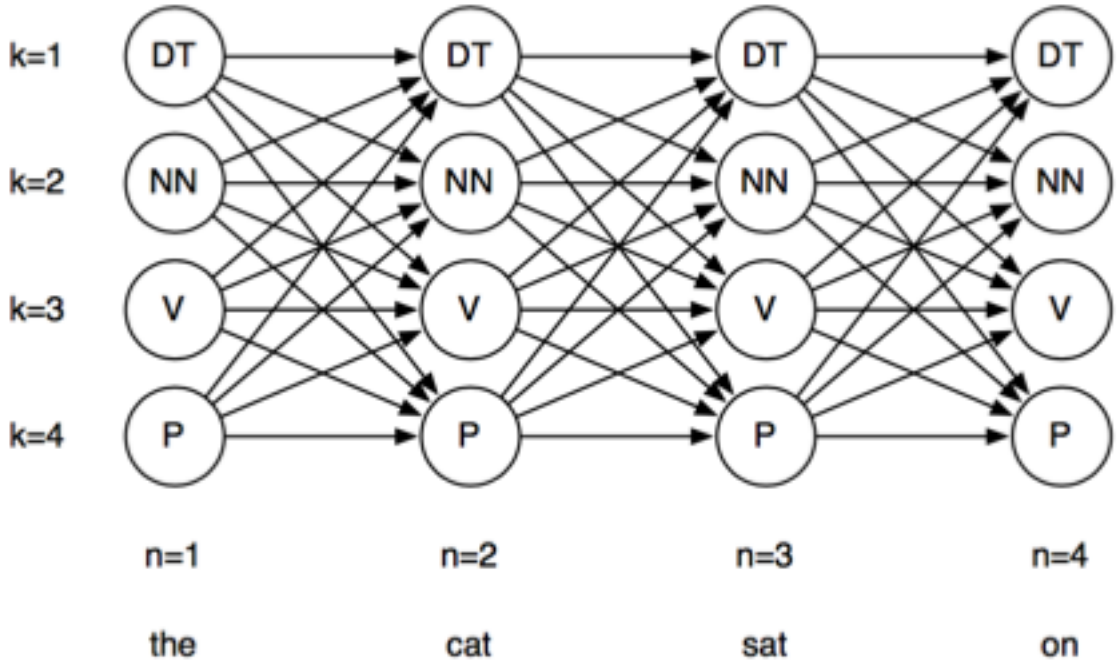
First tag	AT	BEZ	IN	NN	VB	PERIOD
bear	0	0	0	10	43	0
is	0	10065	0	0	0	0
move	0	0	0	36	133	0
on	0	0	5484	0	0	0
president	0	0	0	382	0	0
progress	0	0	0	108	4	0

隐马科夫链应用：词性标注

提问：
如果只给了你一句话
让你给出最符合的tags

这个task属于HMM的问题几？

隐马科夫链应用：词性标注



隐马科夫链应用：词性标注

How can we compute the most likely state sequence efficiently, given that for an N word long sentence, with an average of M POS cats per word, there will be M^N possible sequences? The Viterbi algorithm is a dynamic programming algorithm that allows us to compute the best sequence ('path'), discarding others as we go along.

for each Word $W_{1...n}$
  for each POS category C_j
    for each path ending in some C_k *for* W_{i-1}
      compute $P(C_j|C_k)$*$P(W_i|C_j)$*score of path*
      keep a record of best scoring path to C_j

隐马科夫链应用：词性标注

For a sentence $w_1 \ldots w_N$ of N words, assuming a set $t_1 \ldots t_K$ of K distinct part of speech tags, create an KxN array called 'Score', and another KxN array called 'Backpointer':

Initialise:

for $i = 1 \rightarrow K$ **do**
  Score(i,1) = $P(w_1|t_i) \times P(t_i|\langle start \rangle)$
end for

w1 代表句子中的第一个单词
第i个tag，用在第一个数上的prob

Induction:

for $j = 2 \rightarrow N$ **do**
  for $i = 1 \rightarrow K$ **do**
    Score(i,j) = $\max_{k=1 \ldots K}$(Score(k,j-1) $\times$ $P(t_i|t_k) \times P(w_j|t_i)$)
    Backpointer(i,j) = max k from previous line
  end for
end for

Back tracing the best tagging:

t_N = $\max_i$ Score(i,N)
for $i = N - 1 \rightarrow 1$ **do**
  t_i = Backpointer(t_{i+1},i+1)
end for

简单点说：
初始值是START作为句子的第一个tag
观测值是END出现，成了最后一个tag

隐马科夫链应用：词性标注

【iPyNotebook】

感谢大家！

恳请大家批评指正！