Markov Models

Kevin P. Murphy

Last updated November 26, 2007

1 Stochastic processes

A **stochastic process** is an indexed collection of random variables, $\{X_t\}$, $t \in \mathcal{T}$. If the index set $\mathcal{T}$ is discrete, we will often write $t \in \{1, 2, \ldots\}$, to represent discrete time steps. For a finite number of variables, we will assume $t \in 1 : d$ as usual, where d is the length of the sequence. If the state space $\mathcal{X}$ is finite, we will write $X_t \in \{1, 2, \ldots, K\}$, where K is the number of states. If the state space is countably infinite, we will write $X_t \in \{0, 1, 2, \ldots\}$. If the state space is continuous, we will write $X_t \in \mathbb{R}$, although X_t could also be a vector.

Here are some examples of stochastic processes:

- A finite sequence of i.i.d. discrete random variables, $\{X_1, X_2, \ldots, X_n\}$, where $X_t \in \{1, \ldots, K\}$. This is discrete (finite) time and discrete (finite) state.
- An infinite sequence of non i.i.d. random variables $\{X_1, X_2, \ldots\}$, $X_t \in \mathbb{R}$, representing, for example, the daily temperature or stock price. This is discrete time but continuous state.
- An infinite sequence of non i.i.d. random variables $\{X_1, X_2, \ldots\}$, $X_t \in \{0, 1, 2, \ldots\}$, representing, for example, the number of people in a queue at time t. This is discrete time and discrete state.
- **Brownian motion**, which models a particle performing a Gaussian random walk along the real line. This is continuous-time and continuous-state.

For the rest of this Chapter, we shall restrict attention to discrete-time, discrete-state stochastic processes.

2 Markov chains

Recall that for any set of random variables $X_1, \ldots, X_d$, we can write the joint density using the chain rule as

$$p(X_1, \ldots, X_d) = p(X_1) \prod_{t=2}^{d} p(X_t | X_{1:t-1}) \tag{1}$$

A (first order, discrete-time) **Markov chain** is a stochastic process in which X_t only depends on X_{t-1}, not the whole past:

$$p(X_t | X_1, \ldots, X_{t-1}) = p(X_t | X_{t-1}) \tag{2}$$

We say that X_{t-1} is a sufficient statistic of the past history for predicting X_t. Under this assumption, the joint becomes

$$p(X_{1:d}) = p(X_1)p(X_2|X_1)p(X_3|X_2)\ldots = p(X_1) \prod_{t=2}^{d} p(X_t | X_{t-1}) \tag{3}$$

Define $p(X_1 = j) = \pi_j$ is the **initial state distribution** and $p(X_t = k | X_{t-1} = j) = T_{jk}^{(t)}$ is the **state transition kernel** at time t. If the state space is finite, T can be represented as a $K \times K$ matrix called the **transition matrix**. Each row of the matrix sums to one, $\sum_k T_{jk}^{(t)} = 1$, so this is called a **stochastic matrix**. If we assume the transition matrix $T^{(t)}$ is independent of time, then the chain is called **homogeneous**, **stationary**, or **time-invariant**. This assumption is illustrated in Figure 1, where we see that the T parameter node is a parent of all the X_{it} nodes for $t > 1$; this is called **parameter tying**. (Recall that $i = 1 : n$ indexes the training sequences.) A stationary, finite-state Markov chain is equivalent to a **stochastic automaton**.

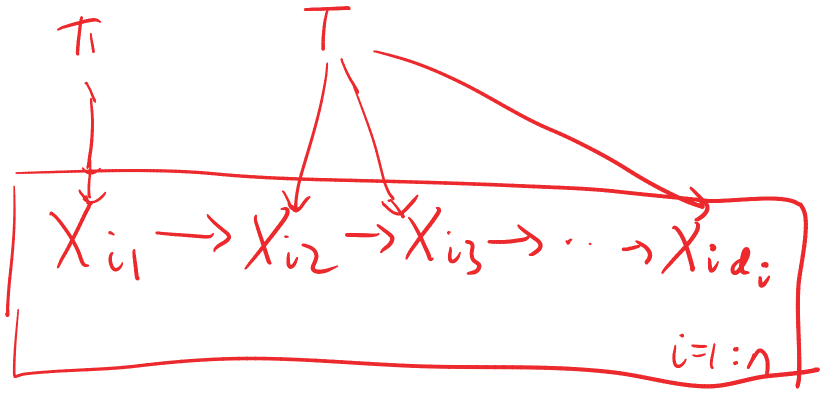

Figure 1: Markov chain as a DGM. π is the initial state distribution and T is the (tied) transition matrix. d_i is the length of the i'th sequence.

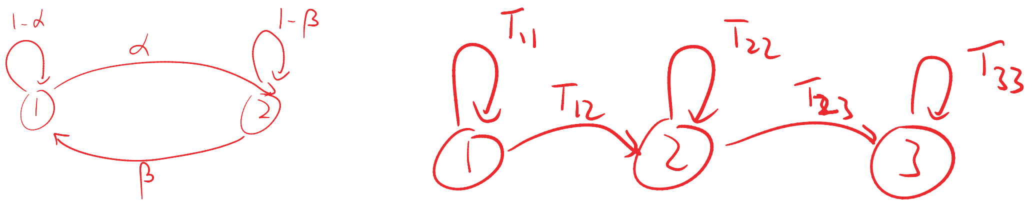

Figure 2: State transition diagrams for some simple Markov chains. Left: a 2-state chain. Right: a 3-state left-to-right chain.

2.1 Examples of Markov chains

The state transition matrix is often visualized by drawing a **state transition diagram**, where nodes represent states and arrows represent legal transitions, i.e., non-zero elements of T. The weights associated with the arcs are the probabilities. For example, the following 2-state chain

$$T = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \tag{4}$$

is illustrated in Figure 2(left). The following 3-state chain

$$T = \begin{pmatrix} T_{11} & T_{12} & 0 \\ 0 & T_{22} & T_{23} \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

is called a **left-to-right transition matrix**, and is illustrated in Figure 2(right). This is commonly used in **speech recognition**: the states represent **phonemes**, and the sequence of phonemes defines a word. Finally, the following 6-state chain

$$T = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

is illustrated in Figure 3. We will use this example later.

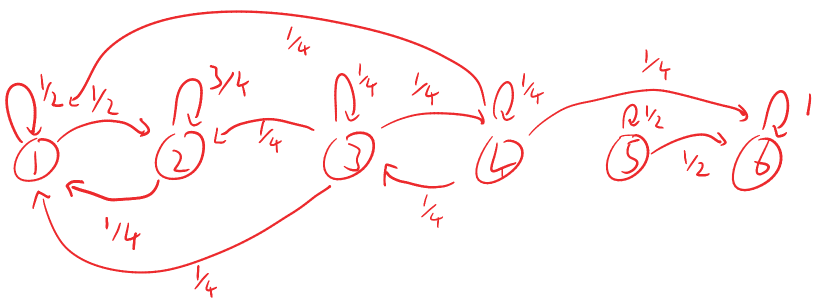
Figure 3: State transition diagram for a 6-state Markov chain.

2.2 Multi-step transition matrix

The T_{jk} element of the transition matrix specifies the probability of getting from j to k in one step. The n-step **transition matrix** $T(n)$ is defined as

$$T_{jk}(n) = p(X_{t+n} = k|X_t = j) \tag{7}$$

which is the probability of getting from j to k in exactly n steps. Obviously $T(1) = T$. The **Chapman-Kolmogorov** equations state that

$$T_{jk}(m+n) = \sum_{l=1}^{K} T_{jl}(m)T_{lk}(n) \tag{8}$$

In words, the probability of getting from j to k in $m+n$ steps is just the probability of getting from j to l in m steps, and then from l to k in n steps, summed up over all l. (Exercise **??** asks you to prove this result.) We can write the above as a matrix multiply

$$T(m+n) = T(m)T(n) \tag{9}$$

Hence

$$T(n) = T\,T(n-1) = T\,T\,T(n-2) = \cdots = T^n \tag{10}$$

Hence we can simulate multiple steps of a Markov chain by **powering up the matrix**.

2.3 Stationary distribution

The probability distribution over states at time t is given by

$$\pi_t(k) = p(X_t = k) = \sum_j p(X_t = k|X_0 = j)p(X_0 = j) \tag{11}$$

$$= \sum_j \pi_0(j)T_{jk}(t) \tag{12}$$

or, in matrix-vector notation,

$$\boldsymbol{\pi}_t = \boldsymbol{\pi}_0 \mathbf{T}^t \tag{13}$$

where we have assumed $\boldsymbol{\pi}_t$ is a row vector. A natural question is: what distribution do we end up with as $t \to \infty$? We shall study this issue below.

Let us start with some examples. Consider the 2-state example in Figure 2(left). Suppose $\alpha = \beta = 1$. It is clear that this chain will continually oscillate between states 1 and 2. On average it will spend 50% of its time in each state. Thus $\boldsymbol{\pi}_\infty = (0.5, 0.5)$. Now consider the 3-state example in Figure 2(right). Clearly we will eventually end up in state 3, and once there, we will never leave; this is called an **absorbing state**. Thus $\boldsymbol{\pi}_\infty = (0, 0, 1)$.

We define the **stationary distribution** (also called the **invariant distribution** or **equilibrium distribution**) of a Markov chain to be a probability distribution $\boldsymbol{\pi}$ that satisfies

$$\boldsymbol{\pi} = \boldsymbol{\pi} T, \text{ i.e., } \pi_j = \sum_i \pi_i T_{ij} \tag{14}$$

Once we enter the stationary distribution, we will never leave. (From the above definition, we have $\boldsymbol{\pi} = \boldsymbol{\pi} T^n$ for all $n \geq 0$.)

2.4 Finding the stationary distribution

Since the stationary distribution satisfies $\boldsymbol{\pi} T = \boldsymbol{\pi}$, or, taking transposes, $T^T \boldsymbol{\pi}^T = \boldsymbol{\pi}^T$, we can solve for the stationary distribution by finding the principal **eigenvector** and renormalizing. In Matlab this becomes

```
[evecs,evals]=eig(T');
pi = normalize(evecs(:,1))';
```

Unfortunately this can be numerically unstable, since it does not explicitly take the constraint $\sum_k \pi_k = 1$ into account. An alternative method exploits the fact [Res92, p138] that $\pi 1_{K \times K} = 1_{1 \times K}$, since the rows sum to one. Hence

$$\begin{aligned} 1_{1\times K} &= (\pi - \pi T) + \pi 1_{K \times K} && (15) \\ &= \pi(I - T + 1_{K\times K}) && (16) \end{aligned}$$

We can solve this in Matlab as follows.

```
pi = ones(1,K) / (eye(K)-T+ones(K,K));
```

Another closely related approach is to solve $\boldsymbol{\pi}(I - T) = \mathbf{0}$ using Gaussian elimination (see [GTH85] for details).

For large, but sparse, transition matrices, a more efficient method is to use the **power method** to solve for $\boldsymbol{\pi}$. We simply start with a random vector and pass it through the transition matrix a large number of times. Eventually this will converge to the stationary distribution. If $\mathbf{T}$ is a sparse matrix, this only takes $O(IK)$ time, where K is the number of states and I is the number of iterations, where we assume that each matrix-vector multiply takes $O(K)$ time. For example, we can implement this in one line of Matlab, starting from an arbitrary initial row vector, and using just 20 iterations:

```
pi = rand(1,K);
for i=1:20
  pi = normalize(pi*T);
end
```

This method is used by **Google**'s **Pagerank** algorithm: see Section 3. All of these methods are implemented in the function `mcStatDist`.

Finally, we give a more intuitive method for finding $\boldsymbol{\pi}$. For a distribution to be stationary, we must have that the net probability flow across any cut-set in the state transition diagram is 0. For example, consider a simple two-state chain with transition matrix

$$T = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \tag{17}$$

as in Figure 2(left). The zero net flow condition gives

$$\pi_1 \alpha = \pi_2 \beta \tag{18}$$

Since $\pi_1 + \pi_2 = 1$, we have

$$\pi_1 = \frac{\beta}{\alpha+\beta}, \quad \pi_2 = \frac{\alpha}{\alpha+\beta}, \tag{19}$$

2.5 Conditions under which a stationary distribution exists

(The following section is rather technical, and may be omitted without loss of continuity.)

Consider the 6-state example in Figure 3. If we start in state 5 or 6, we will end up in 6, and never leave. Hence one possible stationary distribution is $\boldsymbol{\pi} = (0, 0, 0, 0, 0, 1)$. However, if we start in state 1 or 2, we will oscillate between them forever. Hence another possible stationary distribution is $\boldsymbol{\pi} = (\frac{1}{3}, \frac{2}{3}, 0, 0, 0, 0)$. If we start in states 3 or 4, we may either end up in state 6, or end up oscillating between states 1 and 2.

We see from this example that there is not always a unique stationary distribution. However, it is possible to characterize when a unique stationary distribution exists. This will be useful for several applications that we will study later. Unfortunately, precisely characterizing this condition requires a lot of definitions and theorems. We summarize

Term	Meaning
Recurrent/ persistent	Will return to self w.p.1
Transient	Prob. of return is < 1
Null recurrent	Infinite expected time to return to self
Non-null recurrent/ positive recurrent	Finite expected time to return to self
Periodic	Period $d(i) > 1$, where $d = \text{gcd}\{n : T_{ii}(n) > 0\}$
Aperiodic	Period $d(i) = 1$
Ergodic	Recurrent, non-null and aperiodic

Table 1: Summary of kinds of states in a Markov chain.

some of the definitions in Table 1, and state the main theorems below, but we omit the proofs. (See e.g., [GS92] for proofs.) We mostly follow the presentation of [Was04, ch23].

Let us first examine the topology of the state transition graph. We say that state i **reaches** j (or j is **accessible** from i) if $T_{ij}(n) > 0$ for some n, and we write $i \rightarrow j$. If $i \rightarrow j$ and $j \rightarrow i$, then we write $i \leftrightarrow j$, and we say that i and j **communicate**.

Theorem 2.1. *The communication relation is an* **equivalence relation** *and hence satisfies the following properties:*

- *Reflexive:* $i \leftrightarrow i$
- *Symmetric: If* $i \leftrightarrow j$ *then* $j \leftrightarrow i$
- *Transitive: if* $i \leftrightarrow j$ *and* $j \leftrightarrow k$ *then* $i \leftrightarrow k$

Furthermore, the set of states can be written as a disjoint union of **communicating classes**, *where two states* i *and* j *communicate iff they are in the same class. (These correspond to the connected components of the state transition graph.)*

If all states communicate with each other (so the transition graph is a single connected component), then the chain is called **irreducible**. A set of states is **closed** if, once you enter that set of states, you never leave. A closed set consisting of a single state is called an **absorbing state**. In our 6-state example, the communicating classes are $C_1 = \{1, 2\}$, $C_2 = \{3, 4\}$, $C_3 = \{5\}$ and $C_4 = \{6\}$. C_1 and C_4 are irreducible closed sets. State 6 is an absorbing state.

Now we introduce more definitions. State i is **recurrent** or **persistent** if

$$P(X_t = i \text{ for some } t \geq 1 | X_0 = i) = 1 \tag{20}$$

In otherwords, you will definitely return to state i. Otherwise the state is **transient**. In our 6-state example, states 3, 4 and 5 are all transient, because of the path $3 \rightarrow 4 \rightarrow 5 \rightarrow 6$ and because once you enter 6, you cannot return. However, 1,2 and 6 are all recurrent. A chain in which all states are recurrent is called recurrent; a chain in which all states are transient is called transient.

We have the following theorem.

Theorem 2.2. *A state* i *is recurrent iff* $\sum_{t=1}^{\infty} T_{ii}(t) = \infty$.

The intuition behind the proof is that $\sum_{t=1}^{\infty} T_{ii}(t)$ counts the expected number of times you will return to state i. Formally, if $I_t = I(X_t = i)$ indicates whether we are in state i at time t or not, then

$$E[\sum_{t=1}^{\infty} I_t | X_1 = i] = \sum_{t=1}^{\infty} P(X_t = i | X_1 = i) = \sum_{t=1}^{\infty} T_{ii}(t) \tag{21}$$

If the state is recurrent, you will definitely return in n steps, so you will visit the state an infinite number of times.

Theorem 2.3. *Here are some facts about recurrence.*

- *If i is recurrent and $i \rightarrow j$, then j is recurrent.*
- *If i is transient and $i \rightarrow j$, then j is transient.*
- *A finite-state Markov chain must have at least one recurrent state.*
- *The states of a finite-state, irreducible Markov chain are all recurrent.*

We also have the following useful theorem.

Theorem 2.4 (Decomposition theorem)**.** *The state space $\mathcal{X}$ can be partitioned as follows*

$$\mathcal{X} = \mathcal{X}_T \cup \mathcal{X}_1 \cup \mathcal{X}_2 \cdots \tag{22}$$

where $\mathcal{X}_T$ are the transient states and $\mathcal{X}_i$ are closed, irreducible sets of recurrent states.

These facts should all seem intuitively reasonable. In fact, one might wonder how it is possible to only have transient states, since surely you have to return to somewhere? But consider the following example. Suppose we perform a **random walk on the integers**, $\mathcal{X} = \{\dots, -2, -1, 0, 1, 2, \dots\}$. Let $T_{i,i+1} = p$, $T_{i,i-1} = q = 1 - p$. All states communicate, hence either all states are recurrent or all are transient. To see which, suppose we start at $X_1 = 0$. Now

$$T_{00}(2n) = \binom{2n}{n} p^n q^n \tag{23}$$

since the only way to get back to 0 is to take n steps to the right and n steps to the left. We can approximate this using **Stirling's formula**

$$n! \approx n^n \sqrt{n} e^{-n} \sqrt{2\pi} \tag{24}$$

Hence

$$T_{00}(2n) \approx \frac{(4pq)^n}{\sqrt{n\pi}} \tag{25}$$

It is easy to check that $\sum_t T_{00}(t) < \infty$ iff $\sum_t T_{00}(2t) < \infty$. Moreover, $\sum_t T_{00}(2t) = \infty$ iff $p = q = \frac{1}{2}$. By Theorem 2.2, the chain is recurrent if $p = \frac{1}{2}$, otherwise it is transient. This should be intuitively clear: if $p > \frac{1}{2}$, the system will wander off to $+\infty$; if $p < \frac{1}{2}$, it will wander off to $-\infty$.

It should be obvious that a transient chain will not have a stationary distribution. Does that mean that a recurrent chain will have a stationary distribution? For example, consider Figure 2(left): this is irreducible (so long as $\alpha > 0$ and $\beta > 0$), and hence, by Theorem 2.3, is recurrent. It also has a stationary distribution of $\boldsymbol{\pi} = (0.5, 0.5)$,as we saw above. Also, the random walk on the integers we considered above is irreducible and recurrent if $p = \frac{1}{2}$. But does it have a stationary distribution? In fact it does not. The intuitive reason is that the distribution keeps spreading out over a larger and larger set of the integers, and never converges.

This motivates the following definitions. Let

$$f_{ij}(n) = P(X_1 \neq j, X_2 \neq j, \dots, X_{n-1} \neq j, X_n = j | X_0 = i) \tag{26}$$

be the probability that the first visit to state j, starting from i, takes place at the n'th step. Define

$$f_{ij} = \sum_{n=1}^{\infty} f_{ij}(n) \tag{27}$$

to be the probability that the chain ever visits j, starting from i. Obviously j is recurrent iff $f_{jj} = 1$. Define the **mean recurrence time** μ_i of a state i as

$$\mu_i = \sum_n n f_{ii}(n) \tag{28}$$

if i is recurrent; we define $\mu_i = \infty$ if i is transient. Finally, define a recurrent state i as **null** if $\mu_i = \infty$, and as **non-null** or **positive** if $\mu_i < \infty$.

We have the following important theorem.

Theorem 2.5. *An irreducible chain has a stationary distribution $\boldsymbol{\pi}$ iff all the states are non-null recurrent. In this case, $\pi_i = 1/\mu_i$, where μ_i is the mean recurrence time.*

It can be shown (e.g., [GS92, p143]) that for the random walk on the integers, $\mu_i = \infty$ if $p = \frac{1}{2}$. (Intuitively, it takes infinitely long, on average, to return to where you started.) Hence if $p = \frac{1}{2}$, all the states are recurrent, but null. (If $p \neq \frac{1}{2}$, all states are transient.) Thus this Markov chain does not have a stationary distribution. However, one can show that for a finite-state Markov chain, all recurrent states are non-null. By Theorem 2.3, all states of a finite-state irreducible Markov chain are recurrent; hence we have

Corollary 2.1. *Every finite-state irreducible Markov chain has a unique stationary distribution.*

Now consider the example of Figure 2(left) again, where $\alpha = \beta = 1$. If we start at $t = 1$ in state 1, then on every odd time step (1,3,5,...) we will be in state 1; but if at $t = 1$ we start in state 2, then on every odd time step we will be in state 2. Thus although the chain has a unique stationary distribution $\boldsymbol{\pi} = (0.5, 0.5)$, it does not "forget" about the initial starting state. This motivates the following definition.

Let us say that a chain has a **limiting distribution** if

$$T^n \rightarrow \begin{pmatrix} \boldsymbol{\pi} \\ \vdots \\ \boldsymbol{\pi} \end{pmatrix} \tag{29}$$

for some $\boldsymbol{\pi}$. That is, $\pi_j = \lim_{n\rightarrow\infty} T^n_{ij}$ exists and is independent of i. If this holds, then the long-run distribution over states will be independent of the starting state:

$$P(X_t = j) = \sum_i P(X_0 = i) T_{ij}(t) \rightarrow \pi_j \text{ as } t \rightarrow \infty \tag{30}$$

Let us now characterize when a limiting distribution exists. Define the **period** of state i to be

$$d(i) = \text{gcd}\{t : T_{ii}(t) > 0\} \tag{31}$$

where gcd stands for **greatest common divisor**, i.e., the largest integer that divides all the members of the set. (For example, gcd(6,8,10) = 2, and gcd(6,7,8)=1.) In our 2-state chain with $\alpha = \beta = 1$, each state has period 2. We say a state i is **aperiodic** if $d(i) = 1$. (A sufficient condition to ensure this is if state i has a self-loop, but this is not a necessary condition.) Finally, define a state as **ergodic** if it is recurrent, non-null and aperiodic. Define a chain to be ergodic if all its states are ergodic. We can now state our main theorem.

Theorem 2.6. *An irreducible, ergodic Markov chain has a limiting distribution, which is equal to $\boldsymbol{\pi}$, its unique stationary distribution.*

2.6 Time reversibility

We say that a Markov chain T is **time reversible**, or satisfies **detailed balance**, if there exists a distribution $\boldsymbol{\pi}$ such that

$$\pi_i T_{ij} = \pi_j T_{ji} \tag{32}$$

Detailed balance guarantees that $\boldsymbol{\pi}$ is a stationary distribution. To see this, note that

$$\sum_i \pi_i T_{ij} = \sum_i \pi_j T_{ji} = \pi_j \sum_i T_{ji} = \pi_j \tag{33}$$

and hence $\boldsymbol{\pi} T = \boldsymbol{\pi}$. Hence if we can construct a chain with detailed balance, we can ensure that by simulating the chain, we can draw samples from $\boldsymbol{\pi}$. This is the basis of **Monte Carlo Markov chain (MCMC)** methods: see Chapter **??**.

3 Application: Google's PageRank algorithm

We will now see how the concept of stationary distribution is used in **Google**'s **PageRank** algorithm to determine the importance of web pages. We follow the presentation of [Mol04, ch2]. In particular, in this subsection, we assume T is a stochastic matrix in which *columns* sum to one, and in which $\boldsymbol{\pi}$ is a *column* vector. This simplifies the Matlab implementation.

Consider the 6 web pages linked together as shown in Figure 4. We can represent this as a sparse adjacency matrix, where $G_{ij} = 1$ iff there is a link from j to i. In Matlab we can type

```
i = [ 2 6 3 4 4 5 6 1 1];
j = [ 1 1 2 2 3 3 3 4 6];
n = 6;
G  = sparse(i,j,1,n,n);
```

which creates a sparse $n \times n$ matrix with 1's in specified positions, where n is the number of nodes (web pages). (In May 2002, the number of web pages that could be reached by following a series of hyperlinks starting at Google was about 2.7 billion.)

Imagine performing a random walk on this graph, where at every time step, with probability $p = 0.85$ you follow one of the outlinks uniformly at random, and with probability $1 - p$ you jump to a random node, again chosen uniformly at random. If there are no outlinks, you just jump to a random page. (These random jumps ensure the chain is irreducible, i.e., you can get from every node to every other node. This is sufficient to ensure there is a unique stationary distribution, by Corollary 2.1.) This defines the following transition matrix:

$$T_{ij} = \begin{cases} pG_{ij}/c_j + \delta & \text{if } c_j \neq 0 \\ 1/n & \text{if } c_j = 0 \end{cases} \tag{34}$$

where $\delta = (1-p)/n$ is the probability of jumping from one page to another without following a link and $c_j = \sum_i G_{ij}$ represent the out-degree of page j. We can write this more compactly as follows. Define the diagonal matrix D with entries

$$d_{jj} = \begin{cases} 1/c_j & \text{if } c_j \neq 0 \\ 0 & \text{if } c_j = 0 \end{cases} \tag{35}$$

Define the vector $\mathbf{z}$ with components

$$z_j = \begin{cases} \delta & \text{if } c_j \neq 0 \\ 1/n & \text{if } c_j = 0 \end{cases} \tag{36}$$

Then we can rewrite Equation 34 as follows:

$$T = pGD + \mathbf{1}\mathbf{z}^T \tag{37}$$

The matrix T is not sparse, but it is a rank one modification of a sparse matrix. Most of the elements of T are equal to the small constant δ. We can use any of the methods in Section 2.4 to find the stationary distribution of this chain. The resulting entries will be the PageRank scores of each web page.

Let us first find $\boldsymbol{\pi}$ by solving the linear system $\boldsymbol{\pi}(I - T + 1_{n\times n}) = 1_{n\times 1}$:

```
c = sum(G,1);
k = find(c~=0); % non zero outdegree
D = sparse(k,k,1./c(k),n,n);
e = ones(n,1);
I = speye(n,n);
p = 0.85;
z = ((1-p)*(c~=0) + (c==0))/n; % row vector
T = p*G*D + e*z;
pi = (I-T+ones(n,n))\e;
```

We find

$$\boldsymbol{\pi} = (0.3209, 0.1706, 0.1065, 0.1368, 0.0643, 0.2008) \tag{38}$$

So a random surfer will visit site 1 about 32% of the time. We see that node 1 has a higher PageRank than nodes 4 or 6, even though they all have the same number of in-links.

Let us now use the **power method**. We simply iterate

$$\boldsymbol{\pi} = T\boldsymbol{\pi} = pGD\boldsymbol{\pi} + \mathbf{e}\mathbf{z}^T\boldsymbol{\pi} \tag{39}$$

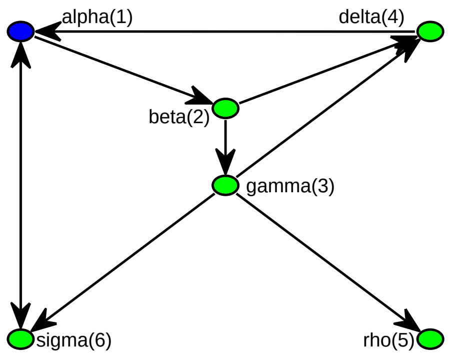

Figure 4: A very small world wide web. Produced by `pagerankDemo`, written by Tim Davis. Based on the example in Section 2.11 of [Mol04].

In this example, we will start from the uniform distribution, but in general, we can start from the distribution computed using last month's web graph. (See [LM06] for details.)

```
pi = e/n;
for i=1:10
  pi = normalize((p*G*D)*pi + e*(z*pi));
end
```

This rapidly converges to the solution (within 10 or so iterations). See `pagerankDemoKPM` for the script file. It is also possible to implement the power method without using any matrix multiplication, by simply sampling from the transition matrix and counting how often you visit each state: see `pagerankpow`, by Cleve Moler, and `pagerankDemo`, by Tim Davis, for illustrations.

Note that PageRank is only one of several factors that Google uses to determine relevancy of a web page. Clearly the most important one is that it contains the **query terms** thay you specified. It is easy to find all pages that contain a given word using a data structure called an **inverted index**.

4 Application: language modeling

Another important application of Markov models is to make statistical **language models**: we simply define the state space to be all the words in English (or some other language), and we then get a probability distribution over word sequences of any length, $p(x_{1:t}|\boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\boldsymbol{\pi}, T)$.

The marginal probabilities $p(X_i = k)$ are called **unigram statistics**. If we use a first-order Markov model, then $p(X_i = k|X_{i-1} = j)$ is called a **bigram model**. If we use a **second-order Markov model**, then $p(X_i = k|X_{i-1} = j, X_{i-2} = l)$ is called a **trigram model**. And so on. In general these are called **n-gram models**.

We can reduce the size of the vocabulary (state-space) somewhat by mapping rare words to the special symbol **unk**, which stands for "unknown". This is particularly helpful when the test set contains words (e.g., proper nouns) that have never been seen before.

Language models can be used for several things, such as

- **Sentence completion**. A language model can predict the next word given the previous words in a sentence by evaluating $p(x_t|x_{1:t-1}) = p(x_t|x_{t-1})$. This can be used to reduce the amount of typing required, which is particularly important for disabled users (see e.g., David Mackay's **Dasher** system[1]).

- **Data compression**. Any density model can be used to define an encoding scheme (see Chapter **??**). The more accurate the predictive model, the fewer the number of bits it requires to store the data.

- **Text classification**. Any density model can be used as a class-conditional density and hence turned into a (generative) classifier: see Exercise **??** for an example. Note that using a 0-gram class-conditional density (i.e., only unigram statistics) would be equivalent to a naive Bayes classifier (see Chapter **??**).

[1]`http://www.inference.phy.cam.ac.uk/dasher/`

```
When in the course of human Events, it becomes necessary for one
People to dissolve the Political Bands which have connected them with
another, and to assume among the Powers of the Earth, the separate and
equal Station to which the Laws of Nature and of Natures God entitle
them, a decent Respect to the Opinions of Mankind requires that they
should declare the causes which impel them to the Separation....
```

```
our Emigrations hitherefore, That the Life, Liberty, all other
Legislature Separalleled totally unwarrantablishing Warfare, acquiesce
in Warfare, transporting his Government. The History of our Laws for
absolved for these right their Safety and Usurpation. He has marked by
these Oppresent ought a ...
```

Figure 5: Example output from an n-gram letter model. Top: input (training) text (Declaration of Independence for the USA, 4 July, 1776). Bottom: output sample from a 5-gram model. Source: `http://jonathanwellons.com/n-grams/index.cgi`

- **Automatic essay writing**. One can sample from $p(x_{1:t})$ to generate artificial text. This is one way of assessing the quality of the model. In Figure 5, we give an example of text generated from a 5-gram model, trained on a relatively short piece of historical text.
- **Speech recognition**. Often the acoustic signal is ambiguous, so it can be combined with a prior, $p(x_{1:t})$, over possible sentences to come up with plausible interpretations of the signal. In this case the sequence of words is unobserved, so we need to use a **hidden Markov model** (HMM): see Section **??**.
- **Spelling correction**. We often make typing mistakes. We can make an HMM in which the typed characters are treated as noisy observations of the true characters which you "meant" to type. We can then infer the most probable spelling of a word.

4.1 Perplexity

The quality of a language model is often measured using **perplexity**. For a fixed probability distribution p, perplexity is defined as

$$\text{perplexity}(p) = 2^{H(p)} = 2^{-\sum_x p(x)\log_2 p(x)} \tag{40}$$

where $H(p)$ is the **entropy**. If $p = (1/K, \ldots, 1/K)$ is the uniform distribution, then

$$\text{perplexity}(p) = 2^{-\sum_{x=1}^K \frac{1}{K}\log_2 \frac{1}{K}} = 2^{-\log_2 \frac{1}{K}} = K \tag{41}$$

Thus a perplexity of K means we are equally confused between K possible choices.

We can obviously achieve a perplexity of 1 using a degenerate distribution with 0 entropy. So we define the perplexity of a probability model $p_{model}(x)$ relative to a true or empirical distribution p_{emp} as follows:

$$\text{perplexity}(p_{model}) \quad = \quad 2^{H(p_{emp}, p_{model})} \tag{42}$$

where $H(p, q)$ is the **cross entropy** (see Section **??**):

$$H(p, q) = -\sum_x p(x) \log_2 q(x), \tag{43}$$

Thus log-perplexity (relative to the empirical distribution) is equivalent to negative log likelihood (nll). We usually compute this on a per-word basis. Suppose the nll for 1000 words was 7950. Then we would say the model has a perplexity of $2^{7.95} = 247$. In other words, the model is as confused on test data as if it had to choose uniformly and independently among 247 possibilities for each word.

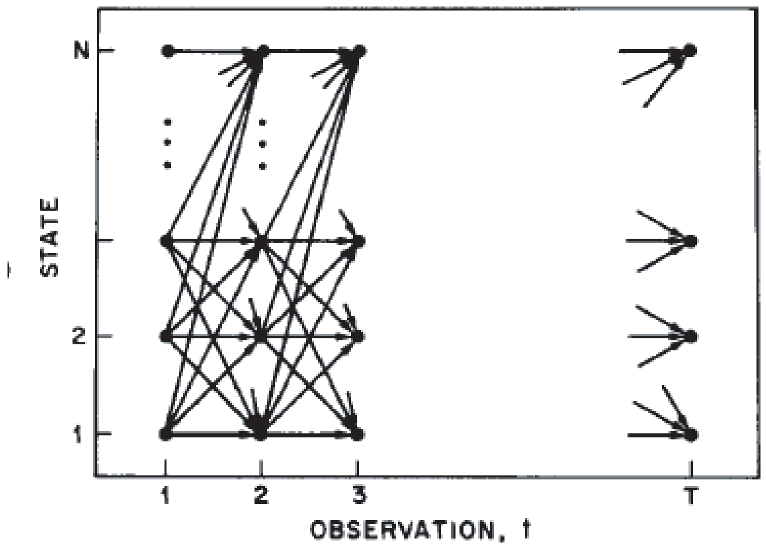

Figure 6: The trellis of states vs time for a Markov chain. Source: [Rab89].

4.2 Maximum likelihood estimation

The probability of any particular sequence is given by

$$p(x_{1:d}|T,\pi) = \pi(x_1)T(x_1,x_2)\dots T(x_{d-1},x_d) \quad (44)$$

$$= \prod_{j=1}^{K} \pi_j^{I(x_1=j)} \prod_{t=2}^{d}\prod_{j=1}^{K}\prod_{k=1}^{K} T_{jk}^{I(x_t=k,x_{t-1}=j)} \quad (45)$$

This is often visualized in terms of paths through a **trellis diagram**, which illustrates the set of legal trajectories through state space, i.e., the set of possible joint assignments $x_{1:d}$. This is shown in Figure 6, where each row corresponds to a state, and each column to a time step.

Of course, the probability of any particular path is likely to be very small, so for numerical reasons we would usually compute the log-likelihood:

$$\log p(x_{1:d}|T,\pi) = \sum_j I(x_1=j)\log \pi_j + \sum_{jk} N_{jk}\log T_{jk} \quad (46)$$

where $N_{jk} = \sum_{t=2}^{d} I(x_{t-1}=j, x_t=k)$ is the number of $j \to k$ transitions. Hence the log-likelihood of a set of sequences $D = (\mathbf{x}_1,\dots,\mathbf{x}_n)$, where $\mathbf{x}_i = (x_{i1},\dots,x_{i,d_i})$ for a sequence of length d_i, is given by

$$\log p(D|\boldsymbol{\pi},T) = \sum_{i=1}^{n} \log p(\mathbf{x}_i|\boldsymbol{\pi},T) = \sum_j N_j^1 \log \pi_j + \sum_j\sum_k N_{jk}\log T_{jk} \quad (47)$$

$$N_j^1 = \sum_{i=1}^{n} I(x_{i1}=j) \quad (48)$$

$$N_{jk} = \sum_{i=1}^{n}\sum_{t=1}^{d_i-1} I(x_{i,t}=j, x_{i,t+1}=k) \quad (49)$$

We now discuss how to optimize this wrt the parameters $\boldsymbol{\pi}$ and T.

Since $X_1 \sim \text{Mu}(\boldsymbol{\pi}, 1)$, we can immediately write down its mle as

$$\hat{\pi}_j = N_j^1/n \tag{50}$$

which is just the fraction of times we started in state j. However, we often set $\boldsymbol{\pi}$ to be uniform or to be the steady state distribution of T, since we typically are not able to predict the first word of a sentence very reliably. Indeed, we will often work with conditional density models of the form $p(x_{2:t}|x_1)$, which does not require that we specify $\boldsymbol{\pi}$.

For the transition matrix, recall that each row defines a distribution over next states given the current state. Hence we have $X_t|X_{t-1} = j \sim \text{Mu}(T_{j,\cdot}, 1)$, so the mle of the transition matrix is given by

$$\hat{T}_{jk} = N_{jk}/N_j \tag{51}$$

$$N_j = \sum_k N_{jk} = \sum_{i=1}^{n} \sum_{t=1}^{d_i-1} I(x_{it} = j) \tag{52}$$

where N_j is the number of times we made a transition out of state j.

4.3 Hierarchical Bayesian model

If we have $K \sim 50,000$ words in our vocabulary, then a bi-gram model will have about 2.5 billion free parameters, corresponding to all possible word pairs. In general, an n-gram models has $O(K^n)$ parameters. It is hard to reliably estimate the probability of so many n-tuples from reasonable amounts of training data. For example, in a bigram model, many of the N_{ij} counts might be zero, even though it is possible that word i can be followed by word j. This is just another example of the **sparse data problem** that we first encountered in Section **??**. Even if the counts are not exactly zero, our estimates of their probabilties will have high variance if the sample size is small. We shall now investigate a hierarchical Bayesian solution to this problem.

A common **heuristic** used to fix this problem is called **deleted interpretation** [CG96]. This defines the transition matrix as a convex combination of the bigram counts $f_{ij} = N_{ij}/N_i$ and the unigram counts $f_j = N_j/N$:

$$T_{ij} = (1-\lambda)f_{ij} + \lambda f_j \tag{53}$$

The term λ is usually set by **cross validation**. There is also a closely related technique called **backoff smoothing**; the idea is that if f_{ij} is too small, we "back off" to a more reliable estimate, namely f_j.

We will now show that the deleted interpolation heuristic is an approximation to a simple hierarchical Bayesian model [MP95]. First, let us use an independent Dirichlet prior on each row of the transition matrix:

$$\mathbf{T}_i \sim \text{Dir}(\alpha m_1, \ldots, \alpha m_K) = \text{Dir}(\alpha \mathbf{m}) \tag{54}$$

where $\mathbf{T}_i$ is row i of the transition matrix, $\mathbf{m}$ is the prior mean (satisfying $\sum_k m_k = 1$) and α is the prior strength. We will use the same prior for each row: see Figure 7. The posterior is given by $\mathbf{T}_i \sim \text{Dir}(\alpha\mathbf{m} + \mathbf{N}_i)$, where $\mathbf{N}_i = (N_{i1}, \ldots, N_{iK})$ is the vector which records the number of times we have transitioned out of state i to each of the other states. From Equation **??**, the posterior mean parameters for the transition matrix are given by

$$\overline{T}_{ij} = \frac{N_{ij} + \alpha m_j}{N_i + \alpha} = \frac{f_{ij}N_i + \alpha m_j}{N_i + \alpha} = (1-\lambda_i)f_{ij} + \lambda_i m_j \tag{55}$$

where

$$\lambda_i = \frac{\alpha}{N_i + \alpha} \tag{56}$$

And from Equation **??**, the posterior predictive density is

$$p(X_{t+1} = j|X_t = i, D) = \overline{T}_{ij} = (1-\lambda_i)f_{ij} + \lambda_i m_j \tag{57}$$

Equation 57 is similar to Equation 53 but not identical. We use a context-dependent weight λ_i to combine m_j with the empirical frequency f_{ij}. This is like *adaptive* deleted interpolation. Furthermore, rather than backing off to the empirical marginal frequencies f_j, we back off to the (learned) model parameter m_j.

To learn the α and $\mathbf{m}$ parameters, we can optimize the marginal likelihood using numerical methods (see below); since these are hyper-parameters, this technique is called **empirical Bayes** or **type II maximum likelihood**.

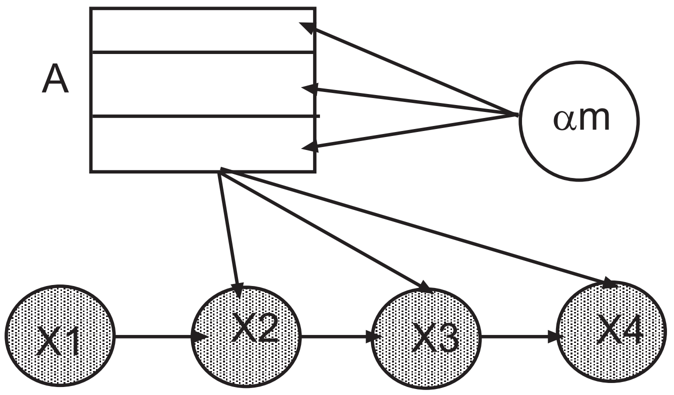

Figure 7: A Markov chain in which we put a different Dirichlet prior on every row of the transition matrix T, but the hyperparameters of the Dirichlet are shared.

4.4 Fitting the hierarchical Bayesian model

In order to derive the expression for m_j above, we proceed as follows. In the empirical Bayes approach, we maximize the marginal likelihood (evidence)

$$\mathbf{u} = \arg\max_{\mathbf{u}} p(D|\mathbf{u}) = \arg\max_{\mathbf{u}} \prod_i \frac{\prod_j \Gamma(N_{ij} + \alpha m_j)}{\Gamma(N_i + \alpha)} \frac{\Gamma(\alpha)}{\prod_j \Gamma(\alpha m_j)} \tag{58}$$

where $\mathbf{u} = \alpha \mathbf{m}$. First we compute derivatives of the log evidence:

$$\frac{\partial}{\partial u_j} \log p(D|\mathbf{u}) = \sum_i \Psi(N_{ij} + u_j) - \Psi(N_j + \sum_{i'} u_{i'}) + \Psi(\sum_{i'} u_{i'}) - \Psi(u_j) \tag{59}$$

where we define the **digamma function** as

$$\Psi(x) = \frac{\partial}{\partial x} \log \Gamma(x) \tag{60}$$

We can now use e.g., conjugate gradients to find a local optimum. Various approximations to these expressions yield Equation **??**: see [MP95] for details.

5 Testing for Markovianity

We can test whether a sequence of data is better modeled by a Markov chain or as an iid sequence by using Bayesian hypothesis testing.[2] Let M_0 be the independence model, where $p(X_t = i) = \theta_i$ and $p(\theta_{1:K}) = Dir(\beta_{1:K})$. Then, using Equation **??**, the marginal likelihood of a sequence $X_{1:N}$ under M_0 is

$$p(X|M_0) = p(x_1) \frac{\Gamma(\sum_j \beta_j)}{\Gamma(N - 1 + \sum_j \beta_j)} \prod_j \frac{\Gamma(N_j + \beta_j)}{\Gamma(\beta_j)} \tag{61}$$

The marginal likelihood under a Markov chain is the product of multinomial evidences, one per state:

$$p(X|M_1) = p(x_1) \prod_i \frac{\Gamma(\sum_j \alpha_{ij})}{\Gamma(N_{i\cdot} - 1 + \sum_j \alpha_{ij})} \prod_j \frac{\Gamma(N_{ij} + \alpha_{ij})}{\Gamma(\alpha_{ij})} \tag{62}$$

where $N_{i\cdot}$ is the number of transitions leaving state i. Hence if we use uniform priors, $\alpha_{ij} = \beta_j = 1$, the Bayes factor for the independence model is

$$\frac{p(X|M_0)}{p(X|M_1)} = \frac{\prod_i \Gamma(N_{i\cdot} + K)}{\Gamma(K)^{K-1} \Gamma(N - 1 + K)} \prod_j \frac{\Gamma(N_j + 1)}{\prod_i \Gamma(N_{ij} + 1)} \tag{63}$$

[2]This section is based on [Min03], although we use slightly different notation (we reverse the subscripts).

For example, suppose $X = (1, 2, 1, 2, \ldots, 1, 2)$. The counts are

$$N_{ij} = \begin{pmatrix} 0 & N/2 - 1 \\ N/2 & 0 \end{pmatrix}, \quad N_j = \begin{pmatrix} N/2 - 1 \\ N/2 \end{pmatrix} \tag{64}$$

so the evidence ratio is

$$\frac{\Gamma(N/2 + K)\Gamma(N/2 - 1 + K)}{\Gamma(K)^{K-1}\Gamma(N - 1 + K)} \tag{65}$$

which rapidly becomes very small as K and N increase, implying the variables must be dependent.

References

[CG96] S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proc. 34th ACL*, pages 310–318, 1996.

[GS92] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford, 1992.

[GTH85] W. Grassmann, M. Taksar, and D. Heyman. Regenerative analysis and steady state distributions for markov chains. *Operations Research*, 33(5):1107–1116, 1985.

[LM06] A. Langville and C. Meyer. Updating Markov chains with an eye on Google's PageRank. *SIAM J. on Matrix Analysis and Applications*, 27(4):968–987, 2006.

[Min03] Tom Minka. Bayesian inference, entropy and the multinomial distribution. Technical report, CMU, 2003.

[Mol04] Cleve Moler. *Numerical Computing with MATLAB*. SIAM, 2004.

[MP95] David McKay and Linda C. Bauman Peto. A hierarchical dirichlet language model. *Natural Language Engineering*, 1(3):289–307, 1995.

[Rab89] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[Res92] Sidney I. Resnick. *Adventures in Stochastic Processes*. Birkhauser, 1992.

[Was04] L. Wasserman. *All of statistics. A concise course in statistical inference*. Springer, 2004.