Markov models

Kevin P. Murphy

Last updated October 24, 2006

* Denotes advanced sections that may be omitted on a first reading

1 Definitions

A **stochastic process** is an indexed collection of random variables, $\{X_i\}$, $i \in \mathcal{I}$, $X_i \in \mathcal{X}$. If the index set $\mathcal{I}$ is discrete, we will often write $i \in \{1, 2, \ldots\}$, to represent discrete time steps. For a finite number of variables, we will assume $i \in 1 : D$ as usual. If the state space $\mathcal{X}$ is discrete and finite, we will write $X_i \in \{1, 2, \ldots, K\}$, where K is the number of states. Recall that for any set of random variables $X_1, \ldots, X_D$, we can write the joint density using the chain rule as

$$p(X_1, \ldots, X_D) = p(X_1) \prod_{i=2}^{D} p(X_i | X_{1:i-1}) \tag{1}$$

A (first order) **Markov chain** is a stochastic process in which X_i only depends on X_{i-1}, not the whole past. Hence the joint becomes

$$p(X_{1:D}) = p(X_1)p(X_2|X_1)p(X_3|X_2) \ldots = p(X_1) \prod_{i=2}^{D} p(X_i|X_{i-1}) \tag{2}$$

The resulting process is called a **(discrete-time, discrete-state) Markov chain**. $p(X_1 = j) = \pi_j$ is the **initial state distribution** and $p(X_i = k|X_{i-1} = j) = T^{(i)}_{jk}$ is the **state transition matrix** at time i. If we assume the transition matrix $T^{(i)}$ is independent of time, then the chain is called **homogeneous/ stationary/ time-invariant**. We will usually make this assumption. Each row of the matrix sums to one, $\sum_k T(j, k) = 1$, so this is called a **stochastic matrix**.

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, Figure 1 shows a **left-to-right** transition matrix commonly used in speech recognition. Another example is a **simple random walk** on the integers $\{1, \ldots, K\}$, so $T(x \to x') = p$ iff $x' = x + 1$, $T(x \to x') = q$ if $x' = x - 1$, and $T(x \to x') = 0$ otherwise. If we make states 1 and K be absorbing states, we get (for $K = 5$)

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ q & 0 & p & 0 & 0 \\ 0 & q & 0 & p & 0 \\ 0 & 0 & q & 0 & p \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

The n**-step transition matrix** $T(n)$ is defined as

$$T_{jk}(n) = p(X_{i+n} = k|X_i = j) \tag{4}$$

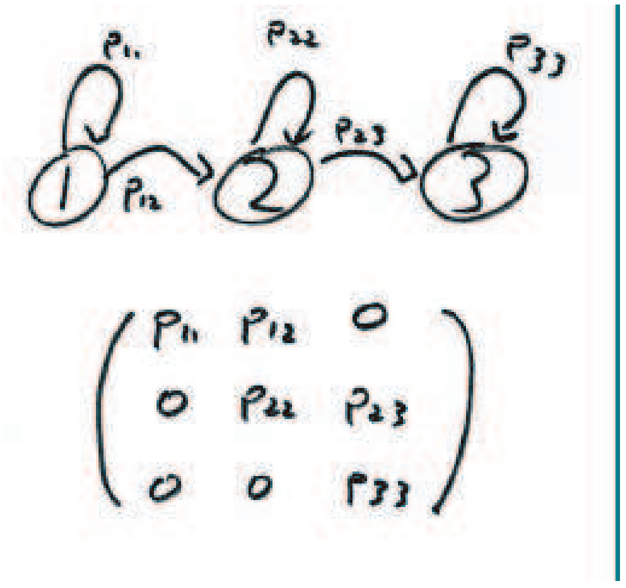

Figure 1: State transition diagram for a 3-state left-to-right Markov chain, with the corresponding transition matrix shown below. Zeros in the matrix correspond to absent edges in the graph.

where $T(1) = T$. The **Chapman-Kolmogorov** equations state that

$$T_{jk}(m+n) = \sum_l T_{jl}(m) T_{lk}(n) \tag{5}$$

So $T(m+n) = T(m)T(n)$ and $T(n) = T^n$. Hence we can simulate multiple steps of a Markov chain by powering up the matrix. For example, the probability distribution over states at time t is given by

$$\begin{aligned} \pi_t(k) &= p(X_t = k) = \sum_j p(X_t = k | X_0 = j) p(X_0 = j) && (6) \\ &= \sum_j \pi_0(j) T_{jk}(t) && (7) \\ \pi_t &= \pi_0 T^t && (8) \end{aligned}$$

where we have assumed π_t is a row vector.

2 Parameter estimation

Since $T_{jk} = p(X_i = k | X_{i-1} = j)$ is just a conditional multinomial distribution, it is easy to estimate its parameters from an observed sequence of states. Let $\pi_k^1 = p(X_1 = k)$ be the initial state distribution. Then the likelihood of a single sequence $x_{1:D}$ is

$$\begin{aligned}
p(x_{1:D}|T, \pi^1) &= \pi^1(x_1) \prod_{i=2}^{D} p(x_i|x_{i-1}) & (9)\\
&= \pi^1(x_1) \prod_i \prod_{jk} T_{jk}^{I(x_i=k, x_{i-1}=j)} & (10)\\
&= \pi^1(x_1) \prod_{jk} T_{jk}^{N_{jk}} & (11)\\
&= \prod_k (\pi^1_k)^{N^1_k} \prod_{jk} T_{jk}^{N_{jk}} & (12)
\end{aligned}$$

where N_{jk} is the number of $j \to k$ transitions and N^1_k is the number of times we start in state k:

$$\begin{aligned}
N_{jk} &\overset{\text{def}}{=} \sum_{i=2}^{D} I(X_i = k, X_{i-1} = j) & (13)\\
N^1_k &\overset{\text{def}}{=} I(X_1 = k) & (14)
\end{aligned}$$

If we have N iid sequences, the likelihood of all the data $\mathcal{D}$ is given by

$$\begin{aligned}
p(\mathcal{D}|\theta) &= \prod_{n=1}^{N} \prod_k (\pi^1_k)^{I(X_{n1}=k)} \prod_{i=2}^{D} \prod_j \prod_k T_{jk}^{I(X_{ni}=k, X_{n,i-1}=j)} & (15)\\
&= \prod_k (\pi^1_k)^{N^1_k} \prod_j \prod_k T_{jk}^{N_{jk}} & (16)\\
N_{jk} &\overset{\text{def}}{=} \sum_n \sum_{i=2}^{D} I(X_{n,i} = k, X_{n,i-1} = j) & (17)\\
N^1_k &\overset{\text{def}}{=} \sum_n I(X_{n1} = k) & (18)
\end{aligned}$$

We can optimize the likelihood wrt the multinomial parameters π^1 and $T_{j,\cdot}$ for each state j separately, to yield

$$\begin{aligned}
\hat{T}_{jk} &= \frac{N_{jk}}{N_j} & (19)\\
N_j &= \sum_k N_{jk} & (20)\\
\hat{\pi}^1_k &= \frac{N^1_k}{N} & (21)
\end{aligned}$$

Note that N_j is the number of transitions leaving state j at any timestep after 1, whereas N^1_k is the number of times we start in state k. An alternative to estimating π^1 is to set it to be the stationary distribution of T. This is actually a harder optimization problem, however, since the parameters become coupled.

Alternatively, we can adopt a Bayesian approach. If we we assume the prior on each row of the transition matrix is independent (the **local parameter independence** assumption) and Dirichlet (the conjugate assumption), then the prior is

$$p(T) = \prod_{j=1}^{K} Dir(T_{j,1:K}|\alpha_{j1}, \ldots, \alpha_{jK}) \qquad (22)$$

where $\alpha_{j,\cdot}$ are the hyperparameters for row j. A factored prior times a factored likelihood induces a factored posterior:

$$p(T|D) = \prod_{j=1}^{K} Dir(T_{j,1:K}|N_{j1} + \alpha_{j1}, \ldots, N_{jK} + \alpha_{jK}) \tag{23}$$

and the posterior mean is

$$T_{jk} = p(X_{i+1} = k|X_i = j) = \frac{N_{jk} + \alpha_{jk}}{N_j + \sum_{k'} \alpha_{jk'}} \tag{24}$$

We can estimate $p(\pi^1|D)$ similarly.

3 Computing the likelihood of a sequence

At test time, we can plug in either the posterior mean or the MLE to compute the predictive density

$$p(x_{1:D}|T, \pi^1) \quad = \quad \pi^1(x_1) \prod_{i=2}^{D} \prod_{jk} T_{jk}^{I(x_i=k, x_{i-1}=j)} \tag{25}$$

Of course, this is likely to underflow, so for numerical reasons we would usually compute the log-likelihood as

$$\log p(x_{1:D}|T, \pi^1) \quad = \quad \log \pi^1(x_1) + \sum_{jk} N_{jk} \log T_{jk} \tag{26}$$

Note that if we use the MLE, it may happen that $\hat{T}_{jk}^{MLE} = 0$, but this can only occur if $N_{jk}^{train} = 0$. Since we define $0 \log 0 = 0$, we will not encounter any numerical difficulties on the training set, but if $N_{jk}^{test} > 0$, we will get a "log of zero" error. Equivalently, the model is saying the probability of this test sentence is 0. We can avoid such problems by using pseudocounts.

4 Application: Language models

If the state space of the Markov chain is all the words in English (or some other natural language), then 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 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 a **trigram model**. And so on. In general these are called **language models**. For example, Figure 2 shows 1-gram and 2-grams counts for the *letters* $\{a, \ldots, z, -\}$ (where - represents space) estimated from *The FAQ Manual for Linux*.

We can sample from the resulting model to produce pseudo-English sentences. With enough training data, this can produce surprisingly good results: see Figure 3 for an example. Another application is to use the Markov model as a class-conditional density in a Bayesian classifier. This can be used to perform offline classification of variable-length sequences.

We can compute maximum likelihood parameter estimates of the transition matrix T_{jk} by counting how many times word j is followed by word k. The above examples used letters, but for *words*, since the state space is large ($K \sim 50,000$), estimating bigram probabilities using maximum likelihood suffers from two problems. The first is that unseen words will be given probability zero. This is often overcome by replacing all novel words (at test time) with the symbol **unk**, which stands for unknown, and assigning a probability to that. Using a uniform Dirichlet prior can avoid 0 counts, but this is not a very accurate prior, since we do not believe that all possible words are equally likely.

The second problem is that our estimate of the probability of word occurrences may have high variance if the sample size is small. One approach is to use **deleted interpretation** [CG96]. This defines the transition matrix as a convex combination of the bigram counts $f_{jk} = N_{jk}/N_j$ and the unigram counts $f_k = N_k/N$:

$$T_{jk} = (1 - \lambda) f_{jk} + \lambda f_k \tag{27}$$

i	a_i	p_i
1	a	0.0575
2	b	0.0128
3	c	0.0263
4	d	0.0285
5	e	0.0913
6	f	0.0173
7	g	0.0133
8	h	0.0313
9	i	0.0599
10	j	0.0006
11	k	0.0084
12	l	0.0335
13	m	0.0235
14	n	0.0596
15	o	0.0689
16	p	0.0192
17	q	0.0008
18	r	0.0508
19	s	0.0567
20	t	0.0706
21	u	0.0334
22	v	0.0069
23	w	0.0119
24	x	0.0073
25	y	0.0164
26	z	0.0007
27	–	0.1928

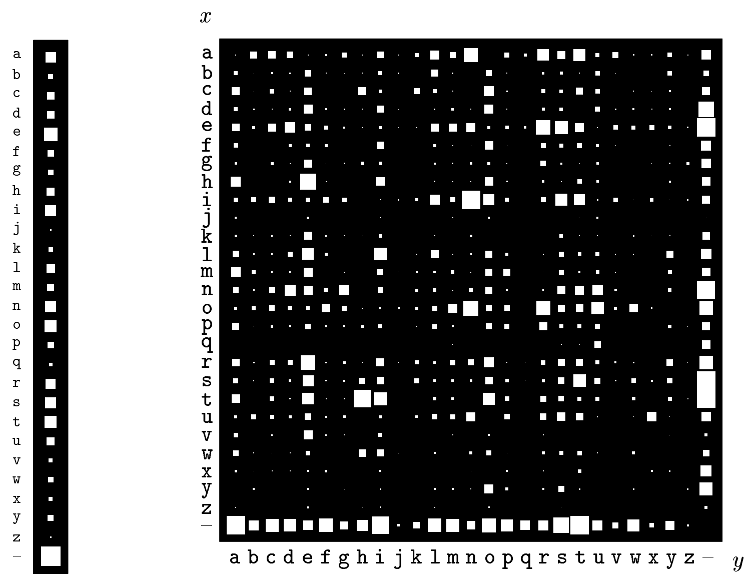

Figure 2: Unigram and bigram counts, from [Mac03, p22].

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 3: Example output from an n-gram letter model. Top: input text (declaration of independence for the USA). Bottom: output sample from a 5-gram model. Source: `http://jonathanwellons.com/n-grams/index.cgi`

where λ is set by cross validation. (There is also a closely related technique called **backoff smoothing**.) See [MP95] for a Bayesian version.

5 Stationary distribution of a Markov chain

Besides their use in language modeling, Markov chains have many other applications as models of stochastic processes. Often we are interested in the long run behavior of such systems. We study this below.

Consider a random walk on the integers 0..20. Imagine starting the chain in state 10, so $p^{(0)}(x) = [0, \ldots, 1, 0, \ldots 0]$, and then iterating $\pi_t(x') = \sum_x \pi_{t-1}(x) T(x \to x')$. The distribution we end up with as $t \to \infty$ is called the **stationary (invariant) distribution** of the chain. Such a distribution π satisfies

$$\pi(x) = \sum_{x'} T(x' \to x) \pi(x') \tag{28}$$

or, equivalently,

$$\pi \quad = \quad \pi T \tag{29}$$

Hence π is a left eigenvector of T with eigenvalue 1 (since $\sum_x \pi(x) = 1$). We can compute π by the **power method**: simply start with an arbitrary distribution π_0 and then iteratively compute

$$\pi_t = T \pi_{t-1} \tag{30}$$

This will converge to the principal eigenvector. See Figure 4 for an example.

5.1 Example

Consider a simple two-state chain with transition matrix

$$T = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix} \tag{31}$$

The stationary distribution π can be found by computing the eigenvector. More intuitively, we must have that the net probability flow across any cut-set in the state transition diagram is 0. Applying this to Figure 5 we have

$$\pi_1 \alpha = \pi_2 \beta \tag{32}$$

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

$$\pi_1 = \frac{\beta}{\alpha + \beta}, \quad \pi_2 = \frac{\alpha}{\alpha + \beta}, \tag{33}$$

5.2 Conditions under which a stationary distribution exists *

Not all Markov chains have stationary distributions, or they may have more than one (eg if $T = I$). We say that a chain has a **limiting distribution** if there exists a π such that $\pi_k = \lim_{n \to \infty} T^n_{jk}$ exists and is independent of j. The precise theorem that specifies when this exists is as follows:

Theorem 1 *A Markov chain has a unique stationary distribution π iff it is irreducible and ergodic. In this case the limiting distribution exists and is equal to π.*

Defining all these terms precisely is beyond the scope of this chapter (see [Was04] for an excellent treatment). But intuitively, they mean the following. An **irreducible chain** is one in which the state transition diagram is a singly connected component. An **ergodic chain** is one all of whose states are ergodic. A state is ergodic if it is recurrent, non-null and aperiodic. A state is **recurrent (persisent)** if you will return to it with

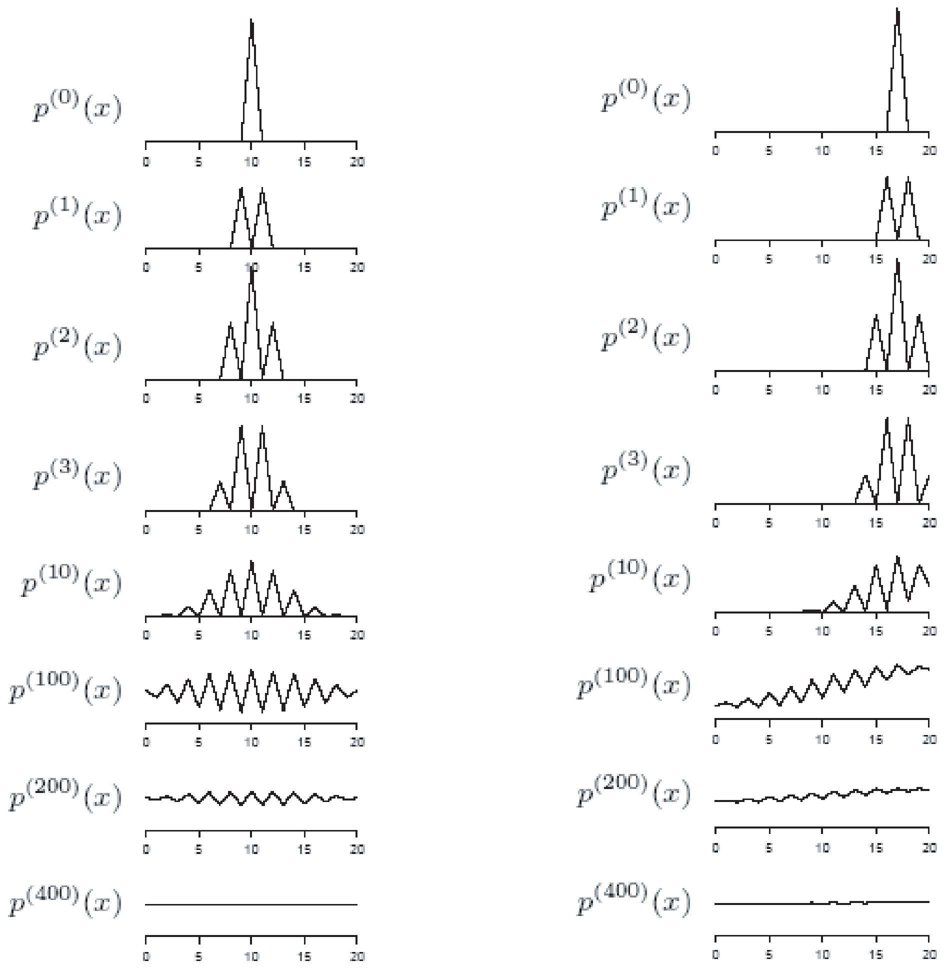

Figure 4: Starting with two different initial distributions and iterating the chain results in the same (uniform) stationary distribution. We assume the transition matrix is a symmetric random walk on the integers 1:20, with reflecting boundary conditions. Source: [Mac03].

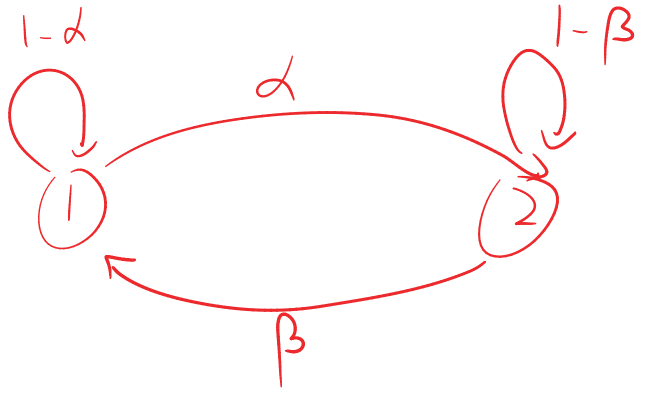

Figure 5: A simple two state Markov chain.

probability 1; otherwise it is called **transient**. A recurrent (persistent) state is **non-null (positive)** if the expected time to return is finite; otherwise it is called null. A state is **aperiodic** if it is possible to return to it in one step; otherwise it is called periodic.

For example, consider a random walk on the integers, where we move left with probability p and right with probability $q = 1 - p$. Clearly the chain is irreducible, since we can reach any states from any other state. If $p \neq q$, then all states are transient, and the system shoots off to $+\infty$ or $-\infty$; hence the chain does not have a stationary distribution. If $p = q = 0.5$, then all states are null[1] recurrent (persistent) with period 2; hence the chain still does not have a stationary distribution.

However, in a finite state Markov chain, one can show that all recurrent (persistent) states are positive (non-null). It is easy to make all states recurrent, by ensuring the transition diagram is a single connected component. And it is easy to ensure all states are aperiodic, by adding self loops. Hence for finite state spaces, we can always ensure a stationary distribution will exist (such a chain will be ergodic).

Sometimes we want to design a markov chain to have a desired stationary distribution. We can do this using the following concept. π satisfies **detailed balance** for T (i.e., the chain is **reversible**) if

$$\pi(x)T(x \rightarrow x') = \pi(x')T(x' \rightarrow x) \tag{34}$$

Theorem 2 *If π satisfies detailed balance for T, then π is a stationary distribution of T.*
Proof:

$$\sum_{x'} \pi(x')T(x' \rightarrow x) = \sum_{x'} \pi(x)T(x \rightarrow x') = \pi(x) \sum_{x'} \pi(x'|x) = \pi(x) \tag{35}$$

Hence if we can construct a chain with detailed balance, we can ensure that by simulating the chain, we can draw samples from π. This is the basis of **Monte Carlo Markov chain (MCMC)** methods.

5.3 Finding the stationary distribution

Since the stationary distribution satisfies

$$\pi = \pi T \tag{36}$$

or, assuming (as is more conventional) that π is a column vector,

$$T'\pi' = \pi' \tag{37}$$

we can solve for the stationary distribution in Matlab by finding the principal eigenvector and renormalizing:

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

$$\pi(I - T + 1_{K\times K}) = 1_{1\times K} \tag{38}$$

We can solve this in Matlab as follows.

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

Alternatively, since we have K constraints from $\pi(I - T) = \vec{0}$, and 1 constraint from $\pi 1_{K\times 1} = 0$, but only K unknowns, we can replace any column (e.g., the last) of $I - T$ with 1, and set the corresponding (last) element of $\vec{0}$ to 1:

[1]It is intuitively obvious that all states are recurrent (persistent), since we can always "walk backwards" to undo any trajectory, but why does it take an infinite amount of time to return on average? One can prove that a persistent state is null iff $p_{ii}(n) \rightarrow 0$ as $n \rightarrow \infty$. So when we power up the matrix and look at the diaognal entries, we see that they tend to 0 and hence all states are null.

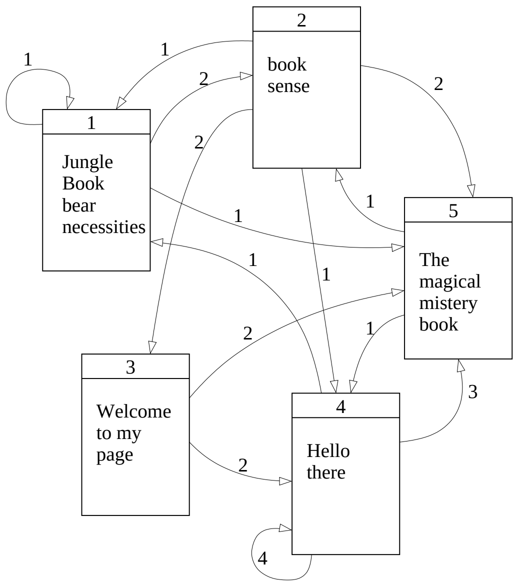

Figure 6: A very small world wide web. The boxes, which are states in the Markov chain, represent web pages that contain various words.

```
tmp = eye(K)-T;
tmp(:,K) = 1;
pi = [zeros(K-1,1); 0] / tmp;
```

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

Finally, we can use the **power method** to solve for π. 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 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. For example, in Matlab with $K = 4$ and $I = 20$, we have

```
x=[.5 .3 .1 .1 0];  % arbitrary
pi = x*(T^20)
```

5.4 Application: Google's PageRank algorithm

Consider the 5 web pages linked together as shown in Figure 6.[2] The weight on edge $j \rightarrow k$ represents the number of hyperlinks from page j to page k; the words in each node represent the words that appear on that page. We can convert this graph into a transition matrix, where $T(j,k)$ represnets the probability of jumping from page j to k, simply by normalizing the counts: $T(j,k) = \frac{N(j,k)}{\sum_{k'} N(j,k')}$. (Note that each row of T sums to one, so T is called a stochastic matrix.) We get

```
T = [1/4 2/4 0 0 1/4
     1/6 0 2/6 1/6 2/6
     0 0 0 2/4 2/4
     1/8 0 0 4/8 3/8
     0 1/2 0 1/2 0];
```

[2]Thanks to Nando de Freitas for this example.

It is easy to find all pages that contain a given word[3]; the problem is to determine the order to return them to the user. Google uses the Page Rank algorithm, which (roughly speaking) determines the "authority" of a page by the number of times it is linked to (directly or indirectly). More precisely, Google treats the web as a Markov chain, and solves for its stationary distribution π (the principal eigenvector of T) using the power method. The authority of page i is then π_i; Google returns answers to queries in order of decreasing probability under the distribution π.

Since the web is a finite state Markov chain, we can guarantee the existence of a unique stationary distribution by ensuring that the graph is irreducible; this will ensure that all states are (non-null) recurrent. We can do this by connecting each node to every other node with a small probability ϵ. (This simulates jumping to a random web page.) Finally we need to ensure all states are aperiodic; we do this by adding a self-loop with small probability ϵ. The remaining probability mass is divided evenly amongst all the original outgoing hyperlinks.

In the example in Figure 6, we find that the stationary distribution is

$$\pi = (0.1053, 0.1842, 0.0614, 0.3860, 0.2632) \tag{39}$$

so the pages are ranked by authority as follows:

$$\pi_4 > \pi_5 > \pi_2 > \pi_1 > \pi_3 \tag{40}$$

Hence when we search for "book", we return the following pages in order: 5 ("The magical mystery book"), 2 ("book sense"), and 1 ("Jungle book: bear necessities"). We do not return pages 3 or 4 since they do not contain the word "book".

Note that, when the web changes its structure slightly, from T to T', we can efficiently compute π' from the old values rather than starting from scratch: see [LM06] for details.

References

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

[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.

[Mac03] D. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003.

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

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

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

[3] To do this efficiently, one should use a data structure called an **inverted index**.