Modelação e Simulação

2013/14

Trabalho de Laboratório nº4

Seriação de páginas web

Objectivo

Após realizar este trabalho, o aluno deverá ser capaz de

- Construir um modelo do tipo cadeia de Markov
- Simular a evolução no tempo das probabilidades dos estados da cadeia de Markov considerada.
- Simular um sistema estocástico de Markov discreto usando o método de Monte Carlo.

Bibliografia

- Acetatos do módulo 8 de *Modelação e Simulação*.
- D. G. Luenberger (1979). *Introduction to Dynamic Systems – Theory, Models, and Applications*. John Wiley and Sons. Cap. 7.

Elementos a entregar

Cada grupo deverá entregar por email um relatório sucinto respondendo às questões do enunciado. As respostas às questões de preparação prévia, identificadas no enunciado como "Em casa", deverão ser manuscritas e entregues em papel. A parte correspondente às questões de simulação deverá ser gerada automaticamente através da função "Publish" do MATLAB,

e entregue por via electrónica conjuntamente com os ficheiros MATLAB/SIMULINK utilizados. Ambas as partes deverão conter um cabeçalho com a identificação do trabalho e a identificação dos alunos (número e nome). As respostas a cada questão deverão ser identificadas pelo seu número. As respostas devem ser concisas.

Nota importante:

Neste trabalho, tal como nos anteriores, o relatório deve ser original e corresponder ao trabalho efectivamente realizado pelo grupo que o subscreve. Relatórios não originais ou correspondentes a software ou outros elementos copiados terão nota zero, sem prejuízo de procedimentos disciplinares previstos pela Lei Portuguesa ou pelos regulamentos do IST.

1. Descrição do problema

Ao processarem o pedido de um utilizador os motores de busca actuais encontram todas as páginas web armazenadas contendo os termos pretendidos, o que resulta habitualmente num volume enorme de dados. A utilidade de toda esta informação aumenta substancialmente se as páginas encontradas forem apresentadas por ordem (decrescente) de relevância. Com uma boa seriação as primeiras entradas da lista são de facto as mais importantes para o utilizador, sendo esta a chave do enorme sucesso do motor de pesquisa Google. À data da sua estreia, no final da década de 90 do século XX, o serviço Google cilindrou a concorrência, apresentado resultados de pesquisa extraordinariamente precisos e alcançando uma vantagem que perdura até hoje. A chave do sucesso do Google é um algoritmo simples mas muito eficaz para seriação (*ranking*) de páginas web, que resultou da investigação realizada por Larry Page e Sergei Brin enquanto alunos de Doutoramento na universidade de Stanford. Este algoritmo *PageRank*, que fornece o contexto para o presente trabalho, baseia-se na ideia que uma página é tanto mais relevante quanto mais vezes for referenciada em hiperligações (*hyperlinks*) a partir de outras páginas. Este modelo de avaliação da relevância a partir da conectividade das páginas é muito diferente de anteriores abordagens em que o motor de pesquisa procurava (com sucesso questionável) inferir a relevância das páginas a partir do seu conteúdo[1].

Atribuindo índices a todas as páginas web guardadas na base de dados do motor de busca, seja então π_j a relevância da página *j*, que dá uma medida de quantos *links* inversos apontam para ela. Para evitar manipulações abusivas o algoritmo pondera todos os n_i *links* contidos numa página *i* por um factor $1/n_i$, pelo que cada página apenas dispõe de *um voto* a distribuir por todas aquelas para as quais aponta. É também natural valorizar mais uma

[1] Tal como todos os outros, o critério de seriação usado pela Google não é imune a manipulações, frequentemente designadas por *Google spoofs*. Algumas destas, como a manipulação de referências ao CV do presidente George W. Bush durante a guerra do Iraque, adquiriram grande notoriedade.

dada ligação para efeitos de cálculo do *ranking* da página de destino se a página de partida da ligação tem ela própria *ranking* elevado. É portanto razoável admitir que o *ranking* π_j é dado por

$$\pi_j = \sum_{i \sim j} \frac{\pi_i}{n_i} \tag{1}$$

onde a notação no índice do somatório $i \sim j$ designa o conjunto dos nós com ligações que apontam para o nó *j*. A expressão acima não deve ser encarada como uma fórmula para o cálculo do *ranking* da página *j*, pois este depende dos *rankings* das páginas que para ela apontam e que são, eles próprios, desconhecidos. Na verdade trata-se de uma equação linear relacionando *rankings*, e juntando as equações referentes a todos os nós obtém-se um sistema de equações cuja solução é o conjunto de *todos* os *rankings* das páginas. O sistema de equações é escrito como

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{P} \tag{2}$$

onde $\boldsymbol{\pi}$ é um vector linha contendo os *rankings*. Reconhece-se que se trata do problema de determinação do vector próprio da matriz $\mathbf{P}^T$ associado ao valor próprio 1. A principal dificuldade deste problema é a sua dimensão; para toda a web esta matriz teria um número de linhas e colunas da ordem de 10^{12}. Neste trabalho considera-se uma versão miniatura deste problema, adoptando-se uma rede com menos de 20 nós representada na Fig. 1.

A normalização pelo factor $1/n_i$ aplicada aos *links* da página *i* (arbitrária) significa que todas as linhas da matriz **P** têm soma unitária, dizendo-se que esta matriz é estocástica por linhas. Daí resulta que o maior valor próprio da matriz **P** é 1, pelo que o sistema de equações escrito acima tem solução. Uma matriz **P** deste tipo é designada por matriz de Markov, podendo-se associar uma máquina de estados à rede correspondente. Tem-se assim uma forma apelativa de reinterpretar a fórmula de cálculo dos *rankings*: Imagine-se que o estado da máquina evolui de acordo com a estrutura probabilística do grafo (rede), transitando de um estado *i* para o seguinte de acordo com as probabilidades de transição $1/n_i$ contidas na linha *i* da matriz **P**. Então o elemento π_j é a probabilidade de ocorrência do estado *j* em regime de equilíbrio. As páginas com *ranking* (<1) elevado são aquelas que

são frequentemente visitadas no modelo de máquina de estados. Como se verá neste trabalho, a interpretação em termos de máquinas de estados permite alargar o leque de técnicas utilizáveis no cálculo do *ranking* de páginas, constituindo uma vantagem importante para lidar com problemas de dimensão muito elevada.

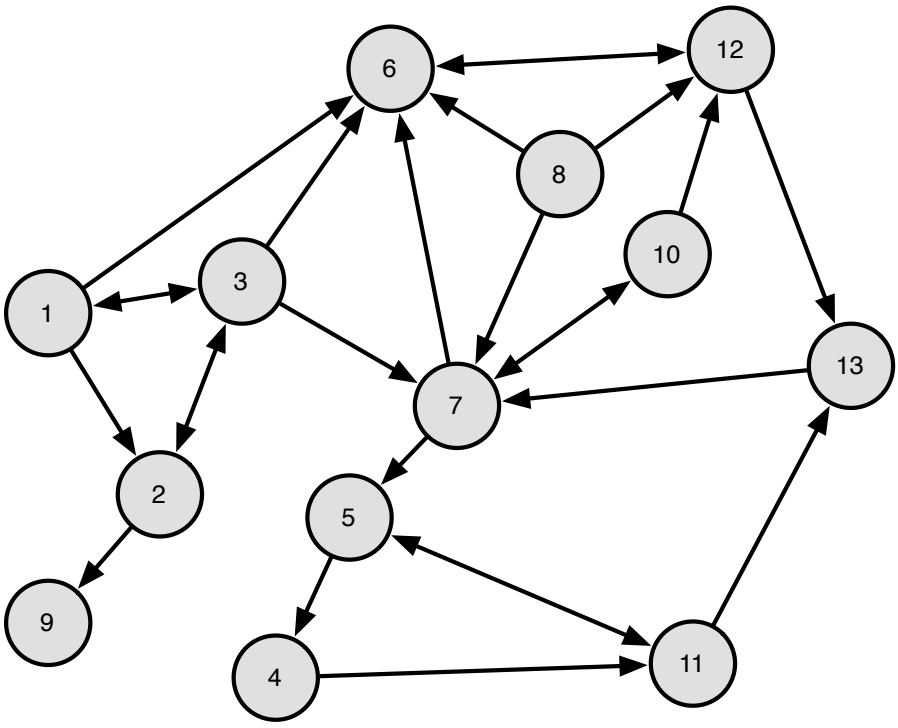

Fig. 1. Um modelo em miniatura da web.

Na web é frequente existirem páginas sem *links* ($n_i = 0$), para as quais não é possível normalizar por $1/n_i$. Nesses casos, o algoritmo *PageRank* substitui toda a linha *i* nula por elementos constantes 1/*N*, onde *N* é o número de nós da rede. Nesta máquina de estados modificada, cuja matriz de transição se designará por **Q**, não existem "becos sem saída". Além disso na web real podem existir grupos de páginas que não podem ser acedidos a partir de certos outros grupos de páginas, o que inviabiliza a hipótese de existência de um regime de equilíbrio para a máquina de estados. Por esta razão, o algoritmo *PageRank* adopta a matriz de Markov modificada

$$\bar{\mathbf{P}} = \alpha\mathbf{Q} + (1-\alpha)\frac{1}{N}\mathbf{E} \tag{3}$$

onde $0 < \alpha < 1$, e **E** é uma matriz com entradas iguais a 1. Criam-se assim ligações fictícias de fraca intensidade (no caso de interesse $\alpha \approx 1$) de um nó para todos os restantes, o que assegura que é sempre possível transitar de qualquer nó para qualquer outro, e que a cadeia de Markov atinge o regime de equilíbrio.

O Google corre o algoritmo *PageRank* regularmente sobre todas as páginas guardadas (cerca de uma vez por mês, crê-se), usando métodos iterativos para obter o vector próprio associado ao valor próprio 1 (método das potências). Os *rankings* resultantes são armazenados numa base de dados e associados às páginas correspondentes, sendo utilizados estaticamente para seriação dos resultados das pesquisas que daí em diante forem apresentadas ao sistema, até nova actualização.

2. Modelo de Markov

Entre dois instantes de tempo consecutivos a evolução das probabilidades dos estados de uma **cadeia de Markov** é dada por

$$\boldsymbol{\pi}(t+1) = \boldsymbol{\pi}(t)\mathbf{P} \tag{4}$$

Aplicando recursivamente no tempo esta igualdade obtém-se a relação entre o vector de probabilidades no instante inicial e as probabilidades num instante posterior

$$\boldsymbol{\pi}(t+1) = \boldsymbol{\pi}(0)\mathbf{P}^t \tag{5}$$

O *método das potências* para determinação da distribuição de equilíbrio dos estados de uma cadeia de Markov baseia-se na propriedade de, sob certas condições[2] (exploradas parcialmente adiante), esta expressão convergir para um limite único e bem definido, $\boldsymbol{\pi} = \lim_{t\to\infty} \boldsymbol{\pi}(t)$.

2.a) (Em casa) Escreva a matriz de probabilidades de transição **P** para a cadeia de Markov associada ao grafo da Fig. 1. Quando uma página não tem *links* admita que a cadeia permanecerá indefinidamente no estado

[2] Este resultado está formalizado no teorema de Perron-Frobenius. Ver, por exemplo, o cap. 8 de Carl D. Meyer, "Matrix Analysis and Applied Linear Algebra" SIAM, 2001 (http://www.matrixanalysis.com/)

correspondente. Verifique que a soma de cada uma das linhas desta matriz é 1 (porquê?) Apresente também a matriz **P** na forma canónica redutível por permutação de linhas/colunas.

2.b) Usando a função *eig* do MATLAB decomponha a matriz $\mathbf{P}^T$ em valores e vectores próprios. Atendendo a que a distribuição de equilíbrio procurada deveria ser dada pelo vector próprio associado ao valor próprio 1, devidamente normalizado[3] para ter soma unitária, interprete os resultados que obteve.

Determine as probabilidades de absorção em cada uma das classes ergódicas partindo de uma distribuição uniforme em cada uma das classes transientes. Determine também o número esperado de passos até à absorção partindo de cada um dos estados transientes. Interprete os resultados.

2.c) Escreva um programa em MATLAB para calcular as probabilidades dos diversos estados da cadeia de Markov ao longo do tempo para diferentes condições iniciais $\boldsymbol{\pi}(0)$.

Use a função *plot3* do MATLAB para visualizar a evolução das probabilidades dos diferentes estados ao longo do tempo e para cada estado. Verifique que, para cada *t*, a soma das entradas de $\boldsymbol{\pi}(t)$ é 1. Interprete os comportamentos que obteve para *t* suficientemente elevado e compare com os resultados de decomposição em valores e vectores próprios da alínea anterior.

2.d) Modifique agora a matriz de transição de probabilidades conforme preconizado em (3) para o algoritmo *PageRank* (**P**→**Q**→$\overline{\mathbf{P}}$ tomando $\alpha = 0.95$) e repita as alíneas 2.b-c, comentando as diferenças face ao caso anterior. O *ranking* de páginas que obteve faz sentido?

[3] Em particular, deveria existir apenas um vector deste tipo, com componentes não negativas.

2.e) Considere agora versões modificadas do diagrama da Fig. 1 em que se elimina sistematicamente uma das ligações[4]. Recalcule os *rankings* seguindo o procedimento da alínea 2.d e determine o caso que mais se afasta do resultado obtido nessa alínea. Interprete o resultado.

3. Simulação de Monte Carlo

A interpretação do problema de determinação do *ranking* de páginas como a obtenção da distribuição de equilíbrio dos estados de uma cadeia de Markov sugere uma estratégia alternativa, baseada em simulação, que evita o cálculo dos produtos matriciais (potencialmente de grandes dimensões) de acordo com (4). A grande esparsidade de **P** permite reduzir drasticamente a complexidade computacional mas, ainda assim, o cálculo de (4) pode ser inconveniente em situações realistas. A metodologia preconizada para simulação consiste em obter múltiplas realizações aleatórias de sequências da máquina de estados, e estimar as probabilidades dos estados directamente a partir do histograma dos estados percorridos. Estes são chamados métodos de **Monte Carlo**[5] devido à analogia com os métodos utilizados nos Casinos.

Na aplicação do método de Monte Carlo ao problema de interesse iremos simular o avanço de uma *marca* que parte de um estado inicial e avança aleatoriamente pelo diagrama de estados durante um certo número de instantes de tempo, anotando-se o número de vezes que a marca ocorre em cada estado. O procedimento total descrito acima, habitualmente designado por *run* de Monte Carlo, é repetido um número suficiente de vezes,

[4] Exclua deste estudo as ligações cuja eliminação torna algum estado inacessível ou transforma-o num "beco sem saída".

[5] Embora tenha raízes mais antigas, que vão até aos trabalhos de Rayleigh e a Buffon, que o utilizou em meados do século XIX para determinar uma estimativa do número π, as origens do nome (que evoca o Casino de Monte Carlo) e da aplicação sistemática do método remontam a 1944, quando foi utilizado no Projecto Manhattan (em que se construiu a primeira bomba atómica) para modelar a difusão de neutrões em bombas atómicas de fissão nuclear. O Método de Monte Carlo é aplicável a uma grande variedade de problemas matemáticos, quer tenham ou não uma essência probabilística. O cálculo numérico do valor de integrais ligados à Estimação Bayesiana é uma das principais aplicações em Engenharia, estando ligado a métodos de grande importância para a tecnologia do séc. XXI.

habitualmente estabelecido à partida para assegurar que os resultados têm significado estatístico. No final, é calculada a frequência relativa com que a marca cai em cada estado (número total de vezes que caiu num estado, a dividir pelo número total de estados percorridos em todos os *runs*).

Implementação Computacional

A estrutura do programa de simulação é muito simples, consistindo num ciclo exterior que percorre *runs* de Monte Carlo, e um ciclo interior que percorre a sequência de estados aleatórios em cada *run*. Por fim, faz-se o cálculo das estatísticas com base nos histogramas acumulados. Pretende-se poder simular algumas variantes do diagrama de estados, pelo que este deve ser codificado numa estrutura de dados apropriada, e não especificado de forma estática no código do programa.

Quando, num estado, pretender seguir um de *n links* equiprováveis, será necessário gerar aleatoriamente um inteiro entre 1 e *n*. Isto pode ser conseguido em MATLAB com a função *randi(n)*, quando disponível, ou com *ceil(n*rand)*. De forma mais geral, se pretender gerar uma variável aleatória com valores entre 1 e *n* em conformidade com um conjunto de probabilidades contidas num vector **p**, pode usar *find(cumsum([p]) > rand,1,'first')*.

A simulação de um número de *runs* elevado pode levar um tempo apreciável. Para ter uma ideia do tempo que falta pode criar uma “barra de espera”. Para isso use o comando *hh=waitbar(espera)* em que *espera* é um número entre 0 e 1 que traduz o comprimento da barra e *hh* é um *handle* da janela onde é criada a barra. No fim, pode fechar a janela com o comando *close(hh)*.

Trabalho a realizar

3.a) Realize simulações de Monte Carlo com inicialização aleatória do estado em cada *run* e um número fixo de passos da máquina de estados. Utilize o diagrama de estados subjacente à matriz de transição modificada (3). Seleccione um período de *burn-in* apropriado atendendo aos resultados de convergência da cadeia de Markov que observou na alínea 2.c.
Supondo que as passagens da máquina por um dado estado são acontecimentos estatisticamente independentes (o que é obviamente falso, mas admitido por simplicidade) discuta a precisão esperada das estimativas de probabilidade dos vários estados. Discuta também de que forma a convergência das estimativas difere dos resultados que obteve na secção 2.

3.b) Tome $\alpha = 1$ e modifique o grafo da Fig. 1 para induzir comportamento periódico na cadeia de Markov. Confirme o resultado esperado por simulação de Monte Carlo e por decomposição em valores/vectores próprios a partir da matriz de transição.