Modelação e Simulação

2019/20

Trabalho de Laboratório nº4

Detecção de hotspots WiFi

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. Apesar de não haver partes do laboratório que impliquem a preparação prévia de um relatório manuscrito, os alunos devem estudar e preparar previamente as questões a desenvolver durante o laboratório. 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. Esta parte deve conter um cabeçalho com a identificação do trabalho e a identificação dos alunos (número e nome dos alunos, número do grupo e turno de laboratório). As respostas a cada questão deverão ser identificadas pelo seu número. As respostas devem ser concisas.

Nota importante:

Quer neste trabalho, quer nos subsequentes, os relatórios devem ser ***originais*** *e corresponder ao* ***trabalho efetivamente 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 e de regulamentos do Instituto Superior Técnico e da Universidade de Lisboa.*

Será utilizado o software de ***deteção automática de plágio*** *MOSS, disponível em http://moss.stanford.edu.*

1. Descrição do problema

Neste trabalho examina-se uma aplicação de detecção de fontes rádio num espaço confinado (por exemplo, pontos de acesso WiFi localizados num edifício/quarteirão, dispositivos Bluetooth, etc.) baseada em medições de potência do sinal recebido num conjunto de sensores. O sistema funciona de uma forma descentralizada em que os sensores realizam medições de potência e trocam informações entre si para, gradualmente, conseguirem determinar a localização de uma fonte presente na zona. Num possível cenário deste tipo um conjunto de agentes é enviado para "varrer" uma zona sem utilizar uma infraestrutura de telecomunicações pré-existente, pelo que as interacções entre eles para localizar a(s) fonte(s) envolvem essencialmente comunicações ad-hoc entre vizinhos[1].

Especificamente, o modo de funcionamento do sistema examinado neste trabalho é *token-based*: Em cada momento um dos agentes detém o *token*, ao qual está associada uma estimativa de localização da fonte. O agente mede a potência do sinal rádio, refina a estimativa de posição, e reenvia o *token* com toda a informação a um dos vizinhos para que este faça o mesmo. Ao circular repetidamente entre os agentes, o *token* fornece a cada um deles dados progressivamente mais precisos sobre a fonte.

Por simplicidade admite-se que os agentes são estáticos, ocupando posições conhecidas. A figura 1 ilustra a disposição de agentes no plano, bem como o grafo de possíveis comunicações entre eles. Quando reenvia o *token*, um agente escolhe aleatoriamente um dos possíveis destinatários, com uma probabilidade não necessariamente uniforme que reflecte as diferenças de qualidade nas ligações. A circulação do *token* pode assim ser modelada como uma cadeia de Markov.

[1] Este tipo de condições ocorre, por exemplo, em meios submarinos, subterrâneos, em cenários de guerra, ou no âmbito de aplicação das redes de sensores.

As tabelas com as localizações dos agentes e as probabilidades associadas às arestas do grafo de comunicações são fornecidas em ficheiros anexos a este enunciado (*MarkovChain.mat*).

O grafo pode ser representado graficamente numa figura Matlab utilizado o comando fornecido *MarkovChainDraw.m*.

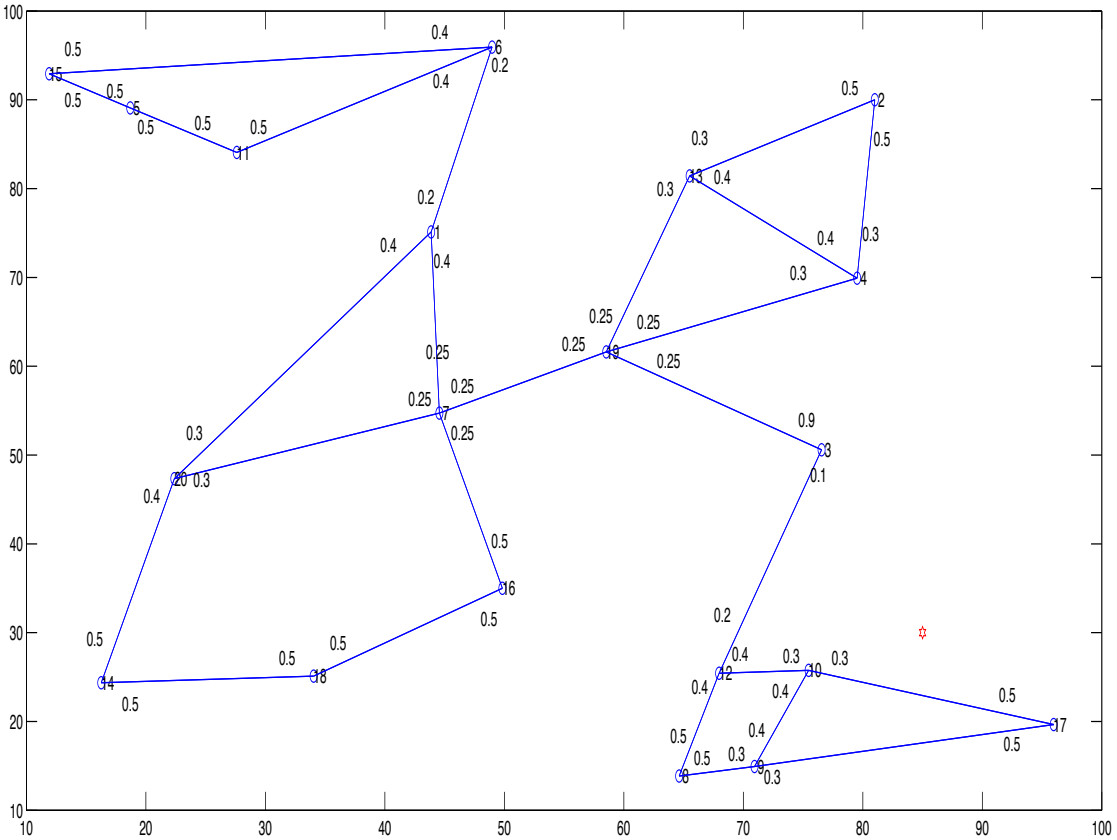

Fig. 1. Grafo de comunicações entre os agentes.

Algoritmo de localização

Muitos algoritmos de localização processam observações que permitem inferir a distância da fonte aos agentes (estes pontos de referência designam-se por *âncoras* neste contexto), determinando a posição da fonte por intersecção aproximada de superfícies de ambiguidade (círculos/esferas) conforme esquematizado na figura

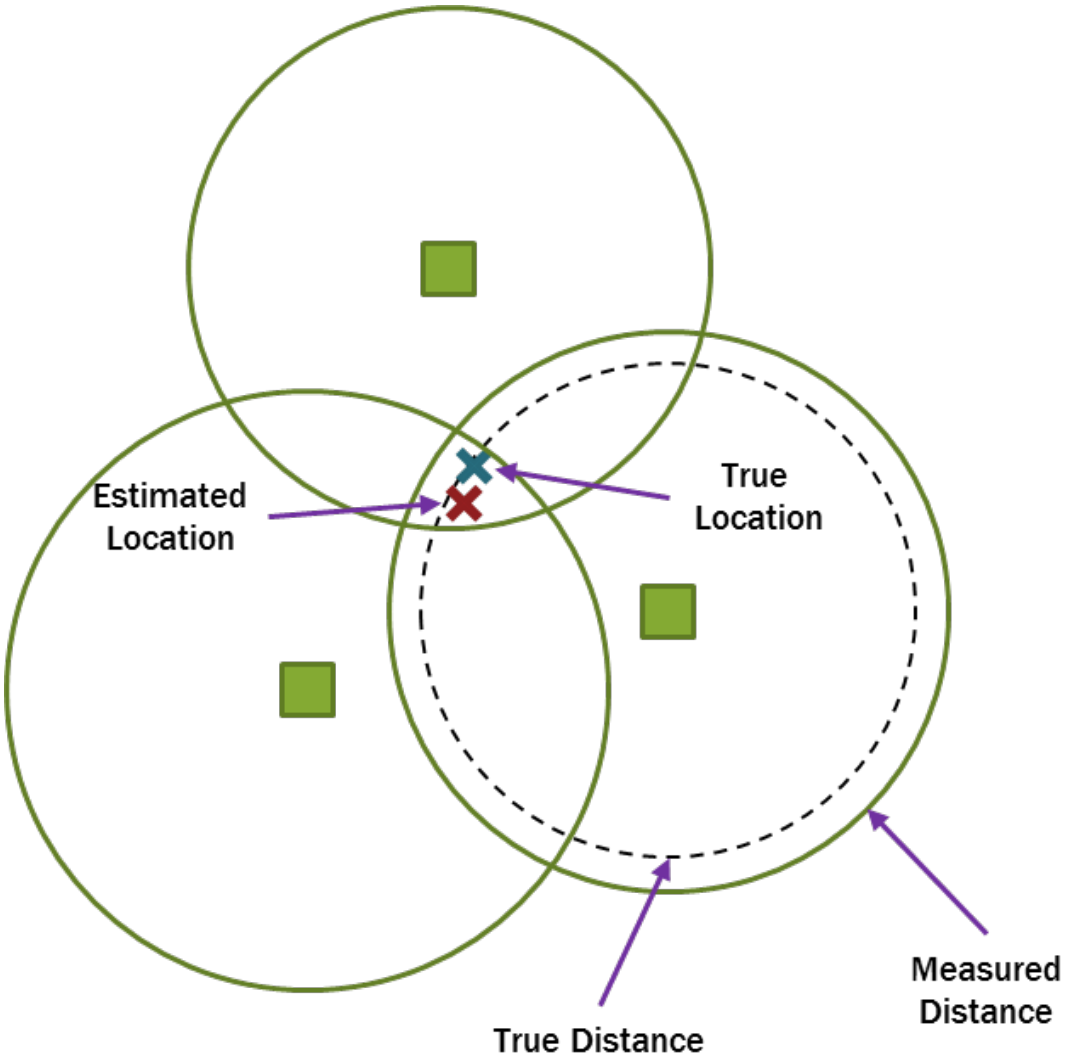

Fig. 2. Localização de uma fonte por intersecção aproximada de círculos/esferas centrados nas âncoras, com raio dado pelas distâncias medidas. As posições das âncoras (agentes) são conhecidas.

Neste trabalho as âncoras medem a potência do sinal recebido (RSS – *Received Signal Strength*).
Na âncora *i* esta potência, P_i, relaciona-se com a distância fonte-âncora através do modelo

$$P_i = \frac{P_0}{\|x - a_i\|^2} e^{n_i}, \tag{1}$$

onde *x* é a posição da fonte e P_0 a sua potência (ambas desconhecidas), a_i é a posição da âncora, e e^{n_i} é um termo de ruído multiplicativo com distribuição log-normal (n_i é uma variável Gaussiana de média nula). Tem-se portanto

$$P_i\|x - a_i\|^2 = P_i(\|x\|^2 - 2a_i^T x + \|a_i\|^2) = P_0 e^{n_i}.$$

Para ruído suficientemente fraco pode fazer-se a aproximação $e^{n_i} \approx 1 + n_i$, de onde resulta

$$P_i(\|x\|^2 - 2a_i^T x + \|a_i\|^2) = P_0 + v_i,$$

onde $v_i = P_0 n_i$ é um termo de ruído Gaussiano de média nula. Agrupando as equações correspondentes a *M* observações, pode formular-se um problema de mínimos quadráticos para determinação das incógnitas *x*, ||*x*|| e P_0, cuja função de custo é dada por

$$\sum_{i=1}^{M} (P_i(\|x\|^2 - 2a_i^T x + \|a_i\|^2) - P_0)^2, \tag{2}$$

ou

$$\left\| A \begin{bmatrix} x \\ P_0 \\ y \end{bmatrix} - b \right\|^2, \tag{3}$$

com

$$A = \begin{bmatrix} -2P_1 a_1^T & -1 & P_1 \\ \vdots & \vdots & \vdots \\ -2P_M a_M^T & -1 & P_M \end{bmatrix}, b = \begin{bmatrix} -P_1\|a_1\|^2 \\ \vdots \\ -P_M\|a_M\|^2 \end{bmatrix}, y = \|x\|^2. \tag{4}$$

Uma forma simples de obter uma estimativa da posição da fonte consiste em ignorar a restrição $y = \|x\|^2$, resolvendo o problema de mínimos quadráticos (convencional) acima como se *x* e *y* fossem variáveis independentes. Na presença de ruído esta simplificação provoca alguma degradação na estimativa, que é aceitável no contexto deste trabalho.

À medida que o *token* circula na rede, cada agente **acrescenta uma linha** à matriz *A* e vector *b* na equação (4) correspondentes à sua observação mais recente, e resolve o problema de mínimos quadráticos aumentado para refinar a estimativa da localização da fonte.

A resolução do problema de mínimos quadrados atrás descrito aumenta de complexidade à medida que o número de observações aumentam. Em alternativa, pode usar-se o algoritmo RLS[2] (*Recursive Least-Squares*) para que tal seja feito de forma eficiente, com custo computacional constante ao longo do tempo apesar do crescimento de *A* e *b*, e transmitindo não estas quantidades, mas apenas variáveis internas com dimensão fixa.

Não sendo o estudo deste algoritmo recursivo de estimação um objectivo do trabalho, fornece-se código que o implementa de uma forma numericamente robusta (*qrrls.m*) e exemplos de aplicação (*rssiloc.m*).

[2] http://en.wikipedia.org/wiki/Recursive_least_squares_filter

2. Modelo de Markov

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

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

Nesta expressão $\boldsymbol{\pi}$ é o vector com a probabilidade de cada um dos estados, e P é a matriz de probabilidades de transição entre estados.

Neste trabalho, cada estado corresponde ao *token* estar na posse de um determinado agente. Nesta secção do enunciado estuda-se o comportamento da cadeia de Markov como forma de compreender se os agentes têm uma frequência de acesso ao *token* adequada. Aplicando recursivamente no tempo a igualdade acima obtém-se a relação entre o vector de probabilidades no instante inicial e as probabilidades num instante posterior

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

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[3], esta expressão convergir para um limite único e bem definido,

$$\boldsymbol{\pi}_\infty = \lim_{t\to\infty} \boldsymbol{\pi}(t).$$

2.a) Usando a função *eig* do MATLAB **decomponha** a matriz $\mathbf{P}^T$ em valores e vectores próprios. **Obtenha** a distribuição de equilíbrio da cadeia de Markov por normalização de um dos vectores próprios e ilustre a mesma usando o comando *bar*. **Qual** o estado mais provável e qual o estado menos provável?

[3] 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/)

2.b) Gere observações de RSS de acordo com o modelo (1) para a localização da fonte fornecida, com $P_0 = 100$ e $n_i \sim N(0, \sigma^2)$, $\sigma^2 = 10^{-2}$.

Atendendo à distribuição de equilíbrio que determinou, **resolva** uma versão ponderada do problema de mínimos quadráticos (2)-(3) para inferir qual seria a posição da fonte estimada, assumindo uma **troca de *tokens* suficientemente longa** entre os agentes.

2.c) Simule a evolução das 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.

2.d) Através da análise do grafo de comunicações entre agentes e da distribuição de equilíbrio determinada na alínea 2.a), **identifique** subconjuntos de estados da cadeia (agentes) onde o *token* possa circular durante períodos relativamente longos.

Relacione a existência destes subconjuntos com o ritmo de convergência que observou no método da alínea anterior.

Sugira e teste:

- Do ponto de vista da equipa, **alterações no peso de algumas ligações** (probabilidades de transição), ou mesmo reorganização de ligações estratégicas, para melhorar a eficácia de circulação global do *token.*

- Do ponto de vista de um elemento hostil, **alterações no peso de algumas ligações** (por exemplo, usando *jamming* selectivo do canal de comunicações) para dificultar a circulação do *token.*

Que impacto poderá ter a fluidez de circulação do *token* na precisão de localização da fonte? (não necessariamente na posição indicada)

3. Simulação de Monte Carlo

A abordagem utilizada anteriormente enfatiza a determinação de equilíbrios e médias estatísticas para caracterizar o comportamento da cadeia de Markov. Nesta secção explora-se uma alternativa baseada em simulação que consiste em obter múltiplas realizações aleatórias de sequências da máquina de estados, e estimar probabilidades dos estados, ritmos de convergência, precisões de localização, ou outras quantidades, directamente a partir do historial assim obtido. Estes são chamados métodos de **Monte Carlo**[4] devido à analogia com os métodos utilizados nos Casinos.

A aplicação do método de Monte Carlo ao problema de interesse é imediata, simulando-se o avanço do *token* que parte de um estado inicial e avança aleatoriamente pelo diagrama de estados durante um certo número de instantes de tempo, anotando-se o historial (ou apenas certas contagens de eventos, se forem suficientes para determinação das estatísticas de interesse). O procedimento total descrito acima, habitualmente designado por *run* de Monte Carlo, é repetido um número suficiente de vezes, habitualmente estabelecido à partida para assegurar que os resultados têm significado estatístico.

[4] 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.

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 no historial acumulado. O diagrama de estados não deve ser especificado de forma estática no código do programa, mas sim codificado numa estrutura de dados apropriada.

Quando, num estado, pretender seguir uma de *n* transições, com probabilidades contidas num vector **p** (com *n* componentes), pode usar o comando

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 (uniforme) do estado em cada *run* e um número fixo de passos da máquina de estados. **Confirme** que a distribuição dos estados é próxima da que determinou na secção 2.

De forma semelhante à alínea 2c, **represente** a evolução no tempo desta distribuição empírica e **compare** os respectivos ritmos de convergência. **Nota diferenças** na concordância com os resultados de 2c entre os estados mais frequentes e os menos frequentes?

Represente também os ritmos de convergência das variantes do grafo da alínea 2d e **comente** os resultados.

3.b) Represente o erro de estimativa da posição da fonte ao longo do tempo para o conjunto das simulações efectuadas.

Represente igualmente o erro ao longo do tempo calculado apenas nas simulações cujo estado inicial pertence ao(s) subconjunto(s) de estados que tende(m) a reter o *token*. **Nota diferenças** na evolução destes erros? As variantes do grafo da alínea 2d influenciam a evolução do erro?

3.c) Simule agora uma movimentação **lenta**, à escolha, da fonte em simultâneo com a circulação do *token.* Neste caso, num dado instante, não é apropriado resolver o problema de mínimos quadráticos (2)-(3) para localização com base em todas as observações disponíveis, pois entre as mais antigas e as mais recentes existirão inconsistências na posição da fonte. Uma forma de lidar com este problema consiste em introduzir um factor de esquecimento[5] $0 < \lambda \leq 1$ no algoritmo RLS, para que na função de custo (2) este pondere os termos da última (mais recente) observação com peso

[5] A implementação fornecida do RLS já incorpora um factor de esquecimento, bastando especificar o seu valor num dos campos da estrutura de parâmetros do algoritmo.

$\lambda^0 = 1$, da penúltima com peso λ, da antepenúltima com λ^2, e assim sucessivamente, de tal forma que o passado é gradualmente descartado.

Ilustre a precisão no seguimento da trajectória da fonte para $\lambda = 1$ e outros 2 valores à escolha do factor de esquecimento.