CCI-22 Trabalho 5
Ajuste de Curvas

April 16, 2018

1 Implementação

Implementar as seguintes funções em MATLAB (cada uma em um arquivo .m separado):

1.1 [2.5pt] c = MinimosQuadrados(Phi, x, y)

realiza ajuste de curva usando o Método dos Mínimos Quadrados (MMQ). Os vetores coluna $\mathbf{x} = [x_1, x_2, ..., x_m]^T$ e $\mathbf{y} = [y_1, y_2, ..., y_m]^T$ representam os m pontos $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ para os

quais se deseja ajustar uma função $f^*(x)$ com modelo:

$$f^*(x) = c_0\Phi_0(x) + c_1\Phi_1(x)c1\Phi1(x) + \cdots + c_n\Phi_n(x) = \mathbf{c}^T\mathbf{\Phi}(x)$$

em que $\mathbf{\Phi}(x) = [\Phi_0(x), \Phi_1(x), \ldots, \Phi_n(x)]^T$ e $\mathbf{c} = [c_0, c_1, ..., c_n]^T$. No MMQ, determina-se $\mathbf{c}$ de modo a minimizar a seguinte função de custo:

$$R(\mathbf{c}) = \sum_{i=1}^{m} \left(f^*(x) - y_i\right)^2 = \sum_{i=1}^{m} \left(\mathbf{c}^T\mathbf{\Phi}(x) - y_i\right)^2$$

1.2 [1pt] [c, r] = RegressaoLinear(x, y)

realiza Regressão Linear de modo a obter a melhor reta que aproxima os m pontos $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$ no

sentido de mínimos quadrados. Assim, tem-se um caso particular de MMQ em que o modelo para $f^*(x)$ é:

$$f^*(x) = c_0 + c_1 x = \mathbf{c}^T \begin{bmatrix} 1 \\ x \end{bmatrix}$$

Além disso, a função calcula o coeficiente de correlação r como estimativa da qualidade do ajuste:

$$r = \sqrt{\frac{R_M - R}{R_M}}$$

onde

$$R = \sum_{i=1}^{m} \left(f^*(x) - y_i\right)^2 = \sum_{i=1}^{m} \left(c_0 + c_1 x_i - y_i\right)^2$$

$$R_M = \sum_{i=1}^{m} \left(\frac{\sum_{i-1}^{m} y_i}{m} - y_i\right)^2$$

1.3 [1pt] c = RegressaoLinear2D(x1, x2, y)

realiza Regressão Linear com duas variáveis independentes x_1 e x_2 . Os vetores coluna $\mathbf{x_1} = [x_{11}, x_{12}, ..., x_{1m}]^T$, $\mathbf{x_2} = [x_{21}, x_{22}, ..., x_{2m}]^T$ e $\mathbf{y} = [y_1, y_2, ..., y_m]^T$ representam os m pontos $\{(x_{1i}, x_{2i}, y_i)\}$, $i = 1, \ldots, m$ para os quais se deseja ajustar uma função de duas variáveis $f^*(x_1, x_2)$ que define um plano no $\mathbb{R}^3$:

$$f^*(x_1, x_2) = c_0 + c_1 x_1 + c_2 x_2$$

Assim, a função determina $\mathbf{c} = [c_0, c_1, c_2]^T$ que minimiza a seguinte função de custo:

$$R(\mathbf{c}) = \sum_{i=1}^{m} \left(f^*(x) - y_i\right)^2 = \sum_{i=1}^{m} \left(c_0 + c_1 x_{1i} + c_2 x_{2i} - y_i\right)^2$$

2 Análise: Complexidade de uma função desconhecida

Encontre experimentalmente a complexidade para o tempo de execução da função fmagicexp, que foi fornecida apenas como arquivo .p. Submeta o código utilizado.

O cabeçalho da função é:
`fmagicexp(n)`
onde

- n é o tamanho da entrada
- A complexidade é O(βn^{α}), e queremos encontrar os 2 parâmetros
- α não é inteiro.
- O tempo de execução foi perturbado artificialmente com um erro extra além de variações normais no tempo de execução.

1. [0.5pt] Depois de gerar as medidas, apresente um gráfico com as medidas que foram consideradas.
2. [1.5pt] utilize mínimos quadrados para encontrar o expoente α e a constante multiplicativa β, do modelo $f(x) = \beta x^{\alpha}$, que utilizaremos para aproximar os dados no sentido de mínimos quadrados, onde x é o tamanho da entrada n.
3. [0.5pt] Depois de encontrar α, plote, no eixo x, o tamanho da entrada n, e no eixo y, o $tempo^{1/\alpha}$. Deve ser uma reta. Use `RegressãoLinear` para encontrar a equação desta reta e o coeficiente de correlação. responda na forma $y = ax + b$, além do valor do coeficiente de correlação.

2.1 Dicas

- Defina um número de pontos e valores coerentes.
- Na primeira chamada de uma função, o MATLAB precisa carregar na memória as funções necessárias, alocar variáveis, etc. Devido ao tempo de carregamento, a primeira medida pode ser errada, por isso podemos fazer uma medida falsa e a descartar.

- (IMPORTANTE) O tempo de execução não é confiável para valores de n muito pequenos. Pois para n pequeno os tempos constantes para excutar a função, alocar variáveis, etc, podem ser dominantes, e a função que deveria ser exponencial pode ser constante ou ter variações aleatórias para n muito pequenos. Como sabemos que é uma exponencial, devemos considerar apenas a partir do momento em que a exponencial é visível no gráfico. Para definir o ponto correto, gere e inspecione um gráfico preliminar que não precisa incluir na resposta.
- Realize uma média de no mínimo 3 medidas com o mesmo valor de n para cada ponto para minimizar o erro.
- Não sobrecarregue a sua CPU ao medir tempo de execução. Feche aplicativos pesados, não assista filmes, etc.

3 Análise: Trilateração

Definição e Teoria

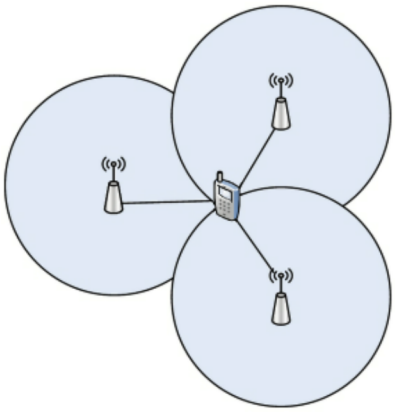

Figure 1: Trilateração: encontrar a posição do sensor dadas medidas de distâncias até beacons.

Queremos determinar a posição (u_x, u_y) de um sensor que pode medir distâncias até n pontos de referência, ou beacons, no plano[1]. Representando a posição do beacon i por (x_i, y_i), a medida correspondente de distância como r_i, então cada medida fornece uma equação na forma:

[1] note que utilizar 2D é apenas uma simplificação. GPS calcula posições em 3 dimensões.

$$(x_i - u_x)^2 + (y_i - u_y)^2 = r_i^2$$

cuja igualdade será exata apenas se não houver erro. Normalmente, a presença de erros de medida resulta em um *resíduo*. Podemos representar estas n equações em forma de matriz:

$$\begin{bmatrix} (x_1 - u_x)^2 + (y_1 - u_y)^2 \\ (x_2 - u_x)^2 + (y_2 - u_y)^2 \\ \vdots \\ (x_n - u_x)^2 + (y_n - u_y)^2 \end{bmatrix}_{n\times 1} = \begin{bmatrix} r_1^2 \\ r_2^2 \\ \vdots \\ r_n^2 \end{bmatrix}$$

Mas estas equações não são lineares, portanto não temos ainda um sistema linear! Podemos linearizar estas equações subtraindo a última linha e realizando alguma manipulação:

$$\mathbf{Au} = \mathbf{b}$$

$$-2 \begin{bmatrix} (x_1 - x_n) & (y_1 - y_n) \\ (x_2 - x_n) & (y_2 - y_n) \\ \vdots & \vdots \\ (x_{n-1} - x_n) & (y_{n-1} - y_n) \end{bmatrix}_{n-1\times 2} \begin{bmatrix} u_x \\ u_y \end{bmatrix}_{2\times 1} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix}_{n-1\times 2}$$

Onde $b_i = r_i^2 - r_n^2 - x_i^2 + x_n^2 - y_i^2 + y_n^2$. Agora temos um sistema linear sobredeterminado, que chamamos de *equações de resíduo linearizadas*, com as seguintes considerações:

- o número de equações foi reduzido de n para $n-1$, mas isto *não deve ser relevante pois o sistema deve ser bastante sobredeterminado.*
- Uma das equações (a n-ésima no exemplo) foi escolhida para ser subtraída das demais. Portanto, erros na posição (x_n, y_n) afetarão todas as equações e portanto terão muito maior efeito no resultado do que erros nas posições dos outros beacons! Mas *o problema supõe que as posições dos beacons são conhecidas com precisão muito maior do que os erros das medidas de distância*!
- Agora temos um problema de mínimos quadrados bem definido e linear! Podemos derivar as equações de resíduo linearizadas, igualá-las a zero, e encontrar o ponto (u_x, u_y), ou aplicar a equação $\mathbf{u} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$.
  - Não esqueça da constante -2!

Dado o problema e demo de trilateração:

3.1 [0.5 pt] Derive as equações normais

Deduza a fórmula fechada das equações normais para o problema da trilateração e escreva as equações no seu relatório na forma de um sistema linear cuja solução seja diretamente a solução do problema. Note que acima, apenas linearizamos o sistema de equações de resíduo original, obtendo um sistema linearizado que consiste de $n-1$ equações onde n é o número de medidas. A resposta aqui é uma fórmula fechada que é um sistema linear com matriz de coeficientes quadrada com tamanho correspondente ao número de incógnitas, ou, para este problema específico, de tamanho 2, onde a solução corresponde diretamente às coordenadas (u_x, u_y) da posição do objeto-sensor a ser localizado. Ou seja, a resposta é um sistema na forma:

$$\hat{\mathbf{A}}_{2\text{x}2} \cdot \mathbf{u}_{2\text{x}1} = \hat{\mathbf{b}}_{2x1}$$

que chamamos de equações normais.

3.2 [1 pt] Implemente as equações normais.

Derive e implemente as equações normais na forma da função

```
function [A,B] = calcAandBforTrilateration(x,y,b)
```

onde `A \ B` é diretamente a solução do problema, e faça passar o teste correspondente.

Não vale utilizar apenas a equação

$$\mathbf{u} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} \tag{1}$$

, ou utilizar `inv()`. É preciso implementar a fórmula fechada das equações normais.

3.3 Comparação dos métodos de solução.

Os métodos para solução de mínimos quadrados são:

1. método dos slides (ou o caminho do cálculo): tomar as equações de resíduo linearizadas, calcular as suas derivadas parciais, igualar as derivadas parciais a zero, e encontrar um sistema linear com tamanho igual ao numero de incógnitas, e obter o resultado diretamente como a solução desse sistema. Para diferenciar este sistema do sistema linearizado de equações de resíduo, denotemos aqui as matrizes com um "^", e teremos a equação $\widehat{\mathbf{A}}\mathbf{x} = \widehat{\mathbf{b}}$.

2. $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$, usando $\mathbf{A}$ e $\mathbf{b}$ do sistema linearizado, o que é uma outra forma para as mesmas equações normais, que vocês aprenderam na disciplina de álgebra.

Não precisa comparar com o método do matlab, embora ele também seja chamado no demo. Sabemos que os métodos 1 e 2 fornecem resultados iguais para este problema, e correspondem à caminhos ligeiramente diferentes para chegar à solução. Se em termos de custo computacional não há muita diferença, então vale escolher o caminho mais fácil ou computacionalmente robusto para o problema específico a ser resolvido.

A desvantagem do segundo método é numérica, ele necessita do cálculo da inversa: mas isso não é importante se o tamanho da matriz é 2!

O primeiro é analiticamente mais complexo pois precisamos derivar as equações do sistema linearizado, igualá-las a zero, etc. Mas chegamos diretamente a um sistema linear que pode ser resolvido sem necessitar de cálculo de inversa: isso seria mais importante se a matriz a ser invertida fosse maior, e/ou se sabemos que ela pode ser singular ou malcondicionada.

3.3.1 [0.25pt] A complexidade dos 2 métodos é a mesma em relação ao número de incógnitas?

Exemplo: Em um sistema GPS real, deveriamos estimar não só 2 incógnitas, mas também a 3a coordenada z. Além disso, como as medidas de distância são derivadas de medidas de tempo de propagação de sinal entre satélites e o receptor GPS, os chamados 'pseudo-ranges', deve-se estimar parâmetros de sincronização dos relógios e intercorrelação de fases de sinal - ou seja, há mais incógnitas.

3.3.2 [0.25pt] A complexidade dos 2 métodos é a mesma em relação ao número de beacons?

Compare a complexidade dos dois métodos em relação ao número de beacons. Para este problema específico, realmente não faz muito sentido pensar em milhares de satélites ou de beacons de navegação, mas de qualquer forma, existem dezenas de satélites considerando os diferentes sistemas GPS, e podemos questionar como os métodos escalam para problemas em geral.

Modelo de resposta conjunta para 3.3.1 e 3.3.2

	$diferenciação\,das\,equações\,de\,resíduo$	$aplicação\,da\,equação\,1\,\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$
complexidade (função O(n,m))		

Justifique cada resposta descrevendo os passos do algoritmo e a complexidade de cada passo.

3.3.3 [1 pt] Mostre que os métodos são analiticamente equivalentes, pelo menos para este problema.

Esta pergunta não é computacional, é um problema de cálculo/álgebra. Mostre que $\widehat{\mathbf{A}} = \mathbf{A}^T\mathbf{A}$, e $\widehat{\mathbf{b}} = \mathbf{A}^T\mathbf{b}$, ou seja:

$$\widehat{\mathbf{A}}\mathbf{x} = \widehat{\mathbf{b}} \Leftrightarrow \mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \Leftrightarrow \mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

o que implica que realmente os dois métodos são equivalentes.

Não precisa de uma prova geral para o MMQ em geral, ou considerar uma classe de problemas, apenas considere as equações linearizadas deste problema e prove para este problema. Derive as equações de resíduo, calcule analiticamente $\widehat{\mathbf{A}}$ e $\widehat{\mathbf{b}}$, e mostre que $\widehat{\mathbf{A}} = \mathbf{A}^T\mathbf{A}$ e $\widehat{\mathbf{b}} = \mathbf{A}^T\mathbf{b}$.

Dicas

- inv() no matlab tem complexidade $O(n^3)$ onde n é o tamanho da matriz a ser invertida. Primeiro a matriz é decomposta com uma decomposição chamada LDL que aproveita o fato de ser simétrico, depois é resolvido um sistema linear, e ambos os passos são $O(n^3)$. Usaria decomposição LU se a matriz não fosse simétrica. Note que é necessário resolver 2 problemas $O(n^3)$, a decomposição e o sistema linear, e não apenas um, ou seja, se mantivermos a constante para indicar que são 2 problemas, a complexidade seria $O(2n^3)$. Para matrizes 2x2 ou 3x3 são utilizadas formulas fechadas diretas, mas para n maiores precisamos considerar $O(n^3)$.

- cuidado com a multiplicação de matrizes. Em geral: para multiplicar $\mathbf{A}_{n\times k}$ por $\mathbf{B}_{k\times m}$ o resultado terá tamanho $n \times m$, e cada elemento do resultado requer multiplicar e somar linhas/colunas de tamanho k. Portanto a complexidade é $O(nmk)$, o que é simplificado para $O(n^3)$ ***apenas*** se ambas as matrizes forem quadradas de tamanho n. Prestem atenção em quem é n,m e k.

- A forma mais geral de responder ao mesmo tempo as duas perguntas sobre complexidade, é responder com na notação bigO usando tanto o número de incógnitas quanto o número de medidas. Note que para desconsiderar termos é preciso haver outro termo dominante com a mesma variável. Exemplo: $O(n^3+n)$ é simplificado para apenas $O(n^3)$,

desconsiderando o termo menor. Mas $O(n^3 + m + n)$ não é $O(n^3)$, é $O(n^3 + m)$.

- Vale fazer substituições de varáveis para simpificar a escrita. Não precisa carregar termos como $(y_i - y_m)$ até o fim, ou continuar considerando o tamanho da matriz como $m - 1$ até o fim. Os termos do vetor $\mathbf{b}$ também se repetem, e podemos usar b_i para simplificar.

4 Instruções

- Não é necessário reimplementar métodos de laboratórios anteriores. Assim, funções implementadas em laboratórios anteriores podem ser utilizadas. Caso prefira, também é permitido utilizar funções equivalentes do MATLAB.
- Não é necessário se preocupar com verificação dos dados de entrada: assuma que x e y tem mesma dimensão etc.
- Os arquivos .m implementados devem ser entregues juntamente com um relatório. No relatório, não é necessário demonstrar que as funções implementadas funcionam corretamente. Basta incluir resultados e conclusões relativos à Análise.

5 Dicas

- A função logaritmo neperiano no MATLAB é `log`.
- Não precisa usar a função geral MinimosQuadrados para implementar Regressão Linear, dado que já há fórmulas fechadas para Regressão Linear na aula.
- Utilize cell array para definir o vetor de funções `Phi`: `Phi{1} = @(x) x, Phi{2} = @(x) x^2, Phi{3} = @(x) cos(x)` etc.