Projeto de Tópicos de Álgebra Linear: Resolução do problema de quadrados mínimos discrtos lineares utilizando a decomposição QR

Prof. Abel Soares Siqueira

Entrega: 30 de Janeiro de 2015

Projeto

1. Seja $A \in \mathbb{R}^{m \times n}$, Assuma que

   - A matriz sempre terá posto-coluna completo;
   - $m \geq n$.

   Implemente o método de Gram-Schmidt aplicado à matriz A, retornando as matrizes $Q \in \mathbb{R}^{m \times n}$ e $R \in \mathbb{R}^{n \times n}$, tais que $A = QR$, Q é uma matriz ortonormal e R é triangular superior.

2. Considere um conjunto de dados $\{(x_i, y_i) : i = 1, \ldots, m\}$, com x_i distintos, e o conjunto de funções $\{\phi_1, \ldots, \phi_n\}$, definidas e contínuas em todo x_i, de modo que a matriz $F = [F_{ij}]$ dada por $F_{ij} = \phi_j(x_i)$ tem posto coluna completo. Procuramos uma função real f, combinação linear dessas funções, que melhor aproxime os valores y_i no sentido de ser solução de

   $$\min_{\alpha} \sum_{i=1}^{m} \left[\sum_{j=1}^{n} \alpha_j \varphi_j(x_i) - y_i \right]^2 = \|F\alpha - Y\|^2 .$$

   (a) Mostre que a solução desse problema é única se, e somente se, a matriz F tem posto-coluna completo.

   (b) Encontre a solução explícita do problema utilizando a decomposição QR.

   (c) Implemente um programa que resolve o problema acima, utilizando a decomposição QR. Seu programa deve, obrigatoriamente seguir os padrões dispostos na seção seguinte.

3. Utilize o seu programa para encontrar a curva na forma $\alpha_1 + \beta_1 x$ que melhor se aproxima dos pontos

x_i	-2.3	-1.5	-0.6	0	1.1	2.3	3.9	4.4	5.9	6.0
y_i	1.0	2.3	3.4	4.1	5.5	6.5	8.9	9.1	11.3	11.2

   Faça um gráfico com os dados e a curva resultante.

Padrões para o código e código auxiliar

Octave/Matlab

Você deve implementar ao menos os três seguintes arquivos do octave/matlab:

- **tri_sup.m**: Implementa a função `tri_sup`, com chamada

  ```
  x = tri_sup(U,b);
  ```

  onde `U` é uma matriz triangular superior n por n, `b` é um vetor de n elementos e `x` é a solução do sistema $Ux = b$.

- **gram_sch.m**: Implementa a função `gram_sch`, com chamada

  ```
  [Q,R] = gram_sch(A);
  ```

  onde `A` é uma matriz m por n com $m \geq n$, `Q` é uma matriz ortonormal m por n e `R` é uma matriz triangular superior n por n tais que $A = QR$ é a decomposição QR de A. Essa decomposição deve ser obtida através do método de Gram-Schmidt.

- **quad_min.m**: Implementa a função `quad_min` que resolve o problema de quadrados mínimos, com chamada

  ```
  [alpha, r] = quad_min(x, y, Funcoes);
  ```

  onde `x` e `y` são vetores coluna de tamanho m com os dados (x_i, y_i), `Funcoes` é uma célula de n elementos, onde cada elemento é um function handler, `alpha` é o vetor de tamanho n com os coeficientes da função minimizadora, e `r` é o vetor de tamanho m com o resíduo.

Julia

Você deve implementar ao menos os três seguintes arquivos do Julia:

- **tri_sup.jl**: Implementa a função `tri_sup`, com chamada

  ```
  x = tri_sup(U,b);
  ```

  onde `U` é uma matriz triangular superior n por n, `b` é um vetor de n elementos e `x` é a solução do sistema $Ux = b$.

- **gram_sch.jl**: Implementa a função `gram_sch`, com chamada

  ```
  (Q,R) = gram_sch(A);
  ```

  onde `A` é uma matriz m por n com $m \geq n$, `Q` é uma matriz ortonormal m por n e `R` é uma matriz triangular superior n por n tais que $A = QR$ é a decomposição QR de A. Essa decomposição deve ser obtida através do método de Gram-Schmidt.

- **quad_min.jl**: Implementa a função `quad_min` que resolve o problema de quadrados mínimos, com chamada

  ```
  (alpha, r) = quad_min(x, y, Funcoes);
  ```

  onde `x` e `y` são vetores coluna de tamanho m com os dados (x_i, y_i), `Funcoes` é um vetor de n elementos, onde cada elemento é uma função (Function), `alpha` é o vetor de tamanho n com os coeficientes da função minimizadora, e `r` é o vetor de tamanho m com o resíduo.

Funções auxiliares

Junto com o projeto, também é disponibilizado as funções que serão utilizadas na avaliação. Com os arquivos acima implementados, você pode rodar cada uma delas e verificar se sua solução está correta. **Você não deve enviar esses arquivos com sua submissão, pois eles não serão considerados. Isso também quer dizer que modificações neles serão ignoradas.**

Para rodar esses arquivos no Octave/Matlab, simplesmente digite o nome do arquivo sem a extensão e dê enter. Para rodar esses arquivos no Julia, use o comando `\include("nome_do_arquivo.jl")`

Minha recomendação é testar cada arquivo na ordem a seguir e só passar para o próximo quando conseguir fazer aquele funcionar. Os arquivos são

1. `teste_tri_sup`
2. `teste_qr`
3. `teste_basico`
4. `teste_quad_min`
5. `teste_quad_min_exato`

A submissão do projeto deve constar de duas partes:

- Um relatório em pdf, escrito em latex, descrevendo o que foi feito, incluindo os testes computacionais. Não precisa enviar o .tex;
- Um arquivo .zip, .rar, ou .tar.gz que contenha os códigos requeridos acima, e outros que sejam necessários para seu programa rodar.

Dicas de LaTeX

- Para inserir seu código no programa, procure o pacote `listings`.
- Para escrever uma matriz, use o código

```
\begin{equation*}
\left[ \begin{array}{cccc}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots &        & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{array} \right].
\end{equation*}
```

que gera

$$\left[\begin{array}{cccc} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{array} \right].$$

O número de c's em `\begin{array}{cccc}` é o número de colunas da matriz.

Dicas de Octave

- O comando `A(:,j)` pega a coluna `j` da matriz `A`.
- Colocando um `.` (ponto) antes de uma operação com matrizes transmite a operação para cada elemento. Então `A^2` corresponde a A^2, mas `A.^2` corresponde à matriz B com $B_{ij} = A_{ij}^2$.
- O comando `f = @(x) x.^2` define uma função f que recebe uma variável x e calcula x^2. Note que, com o `.` antes da operação, podemos passar um vetor ou matriz, e a operação será feita para cada elemento. O nome dessa estrutura é function handler.
- Uma célula é uma estrutura que parece uma matriz, mas cada elemento não precisa ser um número. Podemos ter uma célula de matrizes, ou uma célula de function handlers.

Dicas do Julia

- O comando `A[:,j]` pega a coluna `j` da matriz `A`.
- Colocando um `.` (ponto) antes de uma operação com matrizes transmite a operação para cada elemento. Então `A^2` corresponde a A^2, mas `A.^2` corresponde à matriz B com $B_{ij} = A_{ij}^2$.
- O comando `f = x->x.^2` define uma função f que recebe uma variável x e calcula x^2. Note que, com o `.` antes da operação, podemos passar um vetor ou matriz, e a operação será feita para cada elemento. O nome dessa estrutura é Function.

Quero mais - teoria

- Considere o problema contínuo linear, onde $f(x) = \alpha_1\varphi_1(x) + \cdots + \alpha_n\varphi_n(x)$ é procurada para melhor aproximar uma função $g(x)$ no domínio $D \subset \mathbb{R}$ de todas essas funções. Nesse problema, o somatório não faz sentido, mas podemos utilizar o produto interno

$$\langle f, g\rangle = \int_D f(x)g(x)w(x)\mathrm{d}x.$$

  onde w é uma função positiva no domínio D. O problema então é

$$\min \left\| \sum_{i=1}^{n} \alpha_i\varphi - g \right\|^2.$$

  Qual o procedimento de solução nesse caso? Quais as diferenças principais do caso contínuo para o discreto? Quais as principais dificuldades desse problema?

Quero mais - computacional

- Quando trabalhamos com código, precisamos de alguma maneira eficiente de gerenciar as versões do trabalho. A maneira mais conhecida, e menos eficaz, e guardar cópias do trabalho (`tese-v1.zip`, `tese-v2.zip`, `tese-final.zip`, `tese-final-corrigida.zip`,

`tese-agora-vai.zip`). Uma maneira melhor é utilizar algum software de versionamento. Um dos mais usados atualmente é o `git`. Com ele você pode atualizar as mudanças do código, sem ter que guardar vários arquivos, e ainda pode fazer um backup em alguns dos sites especializados, como o `GitHub`.

- Suponha que você fez um programa que resolve problemas de quadrados mínimos. Daí você resolve atualizar o programa para resolver sistemas sub-determinados. Antes de implementar, naturalmente, você escolhe alguns testes de problemas sub-determinados. Depois de algum tempo você consegue implementar essa modificação, testa os problema sub-determinados e o seu método funciona. Você sobe isso para o GitHub, onde será de utilidade de outras pessoas. Um mês depois, você recebe uma mensagem dizendo que seu programa está quebrado. Ao examinar você descobre que fez uma condição que faz falhar o programa para problemas de quadrados mínimos, que você não testou depois de modificar seu programa.

  Para evitar esse e outros problemas, existem ferramentas de teste automatizados. Uma delas é o Travis CI, que é gratuito para uso com o GitHub.