Minicurso de MATLAB - Nível Básico

Paulo Oliveira Lenzi Valente

2016.1

Conteúdo

Introdução **3**

1 Sintaxe do MATLAB e primeiros passos **3**
1.1 Tipos Básicos de Dados e Variáveis 3
1.2 Operações 5
1.3 Estruturas Condicionais 8
1.4 Laços de Repetição 9
1.5 Funções 10
1.5.1 Funções Úteis 12
1.6 Exercícios 13

2 Mínimos Quadrados e o Ajuste Polinomial **15**
2.1 Mínimos Quadrados e a Equação Normal 15
2.1.1 Cálculo dos Mínimos Quadrados através da Equação Normal 16
2.2 Mínimos Quadrados através de *Gradient Descent* 16
2.2.1 Implementação de Mínimos Quadrados com *Gradient Descent* 17
2.3 Apresentação gráfica dos resultados (*Plotting*) 17

Referências **17**

Introdução

Este curso tem como objetivos:

- Apresentar a sintaxe da linguagem MATLAB
- Introduzir o paradigma da *programação vetorizada*

Como motivação, implementaremos o ajuste polinomial de funções através do método dos Mínimos Quadrados.

1 Sintaxe do MATLAB e primeiros passos

Como a maioria das linguagens de programação, MATLAB possui tipos básicos de dados, variáveis, laços de repetição, estruturas condicionais, funções e operações. A seguir, veremos a sintaxe de cada um desses blocos básicos de construção no MATLAB.

1.1 Tipos Básicos de Dados e Variáveis

Variáveis são os elementos aonde podemos guardar valores. Não são declaradas, apenas inicializadas. Além disso, não possuem tipo explicito, o que mais à frente poderá nos ajudar. Os tipos básicos da linguagem podem ser divididos em:

- Tipos numéricos:
  - Inteiros: $int8, int16, int32, int64$
  - Inteiros sem sinal: $uint8, uint16, uint32, uint64$
  - Ponto flutuante: $single, double$
  - Complexos: Definidos por uma parte real X e uma parte imaginária Y: $X + Yi$ ou $X + Yj$;

  Destes, o tipo *double* é o padrão de armazenamento do MATLAB.

- Caracteres e strings: Para o MATLAB, caracteres são strings de comprimento único. Strings são definidas com aspas simples:

```
myString = 'conteudo';
myChar = 'c';
myEmptyString = '';
```

- Arrays, Matrizes e Cells: Arrays são conjuntos de elementos de um mesmo tipo. São definidos como:

```
vetorA = [1 2 3 4 5]
vetorB = 1:5 % o operador : permite que sejam ...
   criados intervalos numericos.

%
% Para acessar o conteudo de um vetor, usamos a ...
   sintaxe nomeDaVariavel(x), onde x e a ...
   posicao desejada.
% nomeDaVariavel(1) e a primeira posicao e ...
   nomeDaVariavel(end) e a ultima
%
% Ambos vetorA e vetorB possuem o mesmo conteudo.
```

Matrizes são vetores multidimensionais, e são declaradas da mesma forma:

```
matrizA = [1 2 3 4 5; 5, 4, 3, 2, 1];
matrizB = [1:5; 5:-1:1];
% Ambas as matrizes matrizA e matrizB possuem o ...
   mesmo conteudo.

% ['string1' 'stringDois'] == 'string1stringDois'
% ['string1'; 'stringDois'] resulta em erro ...
   porque as linhas nao possuem o mesmo tamanho
% ['strings1'; 'string2'] resulta em uma matriz ...
   de duas linhas

% matrizA(1,5) == 5
% matrizB(2,3) == 3
% matrizA(1,3:end) == [3 4 5];
% matrizA(1,end:-1:3) == [5 4 3];
```

Cells são conjuntos heterogêneos, mas funcionam de forma similar a matrizes:

```
myCell = {1, 'dois', 3.0, 3+3i,'cinco'};

% como pode-se observar, uma cell possibilita, ...
   por exemplo, a criacao de
% conjuntos de string de tamanhos diferentes
```

1.2 Operações

O MATLAB é uma linguagem voltada para facilitar cálculos matemáticos. Com isso temos alguns tipos de operações que veremos a seguir:

```
%% Operacoes Numericas
x = 1.5 + 2.5; % Soma
y = 3 - 1;     % Subtracao
z = x * y;     % Multiplicacao
i = z / 8;     % Divisao --- IMPORTANTE: i e a variavel ...
   complexa. Assim, ao realizarmos esse calculo, estamos ...
   sobreescrevendo esta variavel ate executarmos o ...
   comando para limpar a memoria do programa.

k = 2 \ 4; % Executa a divisao do termo da direita pelo ...
   da esquerda
2 == k     % Comparacao. Retorna 1, o valor logico para ...
   verdadeiro
1024 = k^10; % exponenciacao
1000 = 1e3; % a notacao AeB e utilizada para notacao ...
   cientifica. E o mesmo que multiplicar A por 10 ...
   elevado a B
%% Operacoes matriciais/vetoriais
% Valem as regras matematicas de multiplicacao vetorial

A = [1:3; 2:4; 3:5];
B = -1*A; % multiplicacao por escalar
C = A + B; % soma matricial
D = A - B; % subtracao matricial
L = [1 2 3];
L = L' % transposicao

moduloQuadrado = L * L'; % produto interno entre dois ...
   vetores
```

```
E = A * L; % Multiplicacao vetorial/matricial
F = E/2 ; % Divisao por escalar

% Operacoes especiais com matrizes
A = [1 2 3];
A = [A; A; A];

B = [1:3; -1 -2 -3; 0 0 0];

inversaB = inv(B); % funcao para inverter uma matriz. ...
   Mais sobre funcoes abaixo

C = A .* B; % produto elemento a elemento
C2 = A ./ B; % divisao elemento a elemento
D = A / B; % mesma coisa que A * inv(B);
E = A \ B; % o mesmo que inv(A) * B;

% Caso de uso: gerar todas as potencias de um numero com ...
   expoentes de 0 ate n
n = 10;
numero = 2;
potencias = numero.^(1:n);

% (1:n).^2 eleva todos os numeros do vetor [1:n] ao ...
   quadrado, enquanto 2.^(1:n) eleva 2 a cada um dos ...
   elementos do vetor [1:n] e resulta em um vetor de ...
   mesmo tamanho


%% Operacoes logicas
A seguir, vamos gerar as tabelas-verdade das operacoes ...
   logicas basicas


A = floor( (0:3)/2 ); % [0 0 1 1]
B = mod( [0:3] , 2 ); % [0 1 0 1]

%Operacoes bit a bit (bitwise)
% not A
~A == [1 1 0 0];
% A AND B
  A & B == [0 0 0 1];
% A OR B
    A | B == [0 1 1 1];
```

```
%Comparacoes e operacoes entre expressoes logicas:

C = [1:11];
C == 5; % [0 0 0 0 1 0 0 0 0 0];
C > 5; % [0 0 0 0 0 1 1 1 1 1];
C < 5; % [1 1 1 1 0 0 0 0 0 0];
C >= 5; % [0 0 0 0 1 1 1 1 1 1];
C <= 5; % [1 1 1 1 1 0 0 0 0 0];
D = -1*C(end:-1:1);

any( ~(C + D) ) && any(C == -D) % AND entre duas ...
   expressoes logicas
all(C + D < 10) || ( C + D >= -10) % OR entre duas ...
   expressoes logicas

%Logical Indexing
% Pode-se utilizar um array logico para extrair ...
   informacoes de outro array

A = [1:10];

% Extrair os elementos pares e impares de A:
pares = A( ~mod(A,2) );
impares = A ( logical(mod(A,2)) );

% Zerar os elementos pares e impares de A:
zera_impares = A .* ( ~mod(A,2) );
zera_pares = A .* ( mod(A,2) );
```

1.3 Estruturas Condicionais

Os blocos condicionais do MATLAB não são muito diferentes do que estamos acostumados em linguagens como C. Existem dois tipos de estrutura condicional, como veremos a seguir.

```
% IF/ELSEIF/ELSE
  A = input('Insira um numero'); % a funcao input aceita ...
     entradas do usuario

  if (A < 0)
    disp('Voce inseriu um numero negativo');
  elseif (A > 0)
    disp('Voce inseriu um numero negativo');
  else
    disp('Voce inseriu 0');
  end

% SWITCH
numero = input('Insira um numero: ');
try
    numero = mod(numero,2);
    switch numero
        case 0
            disp('Voce inseriu um numero par!');
        case 1
            disp('Voce inseriu um numero impar!');
        otherwise
            disp('Voce nao inseriu um numero!');
    end
catch
    disp('Voce nao inseriu um numero!');
end


% A estrutura try-catch funciona para tratamento de ...
   erros. E equivalente ao try-except da linguagem Python.
```

1.4 Laços de Repetição

MATLAB possui dois tipos de estruturas de repetição: FOR e WHILE. A seguir, veremos como calcular o fatorial de um numero usando cada uma das estruturas, e a forma vetorizada de realizar este calculo. É importante notar que a forma vetorizada tende a ser muito mais rápida do que o código escrito com loops em MATLAB. Assim, sempre devemos buscar vetorizar o código ao máximo, utilizando operações matriciais e funções prontas da linguagem. Casos em que a vetorização não costuma ser possível são os casos em que uma iteração do loop utiliza resultados da iteração anterior, ou quando no número de iterações é inerente ao algoritmo.

```
numero = input('Insira um numero para calcular seu ...
    fatorial: ');

% FOR
  fatorial = 1;
  for n=1:numero
    fatorial = fatorial * n;
  end
  disp(fatorial);
% WHILE
  fatorial = 1;
  iterador = numero;
  while (iterador > 0)
    fatorial = fatorial * iterador;
    iterador = iterador - 1;
  end
  disp(fatorial);

% FORMA VETORIZADA - UTILIZA A FUNCAO prod()
  fatorial = prod( 1:numero );
  disp(fatorial);

% existem tambem os comandos break e continue, que ...
    permitem que voce saia do loop(break) ou termine a ...
    iteracao atual(continue);
```

1.5 Funções

Já utilizamos algumas funções em exemplos anteriores, mas vamos olhar um pouco mais a fundo como construir uma função. Após isso, veremos uma lista de funções muito úteis para a vetorização de códigos.

```
function [output_args] = nomeDaFuncao([input_args])
  %manual da funcao
  %manual da funcao
  %manual da funcao

  corpo da funcao
end
```

Uma função é um bloco de código que pode ser invocado sem a necessidade de saber sua implementação. Ou seja, podemos utilizar uma função sabendo *o que* ela faz, sem saber *como*. Funções possuem argumentos de saida (output_args) e argumentos de entrada (input_args) que são por onde recebemos o resultado, caso haja algum, e por onde inserimos valores na função, respectivamente. Assim como em Python, se colocarmos comentários no topo do corpo da função, esses comentários serão o texto mostrado quando pedirmos seu manual `help nome_da_funcao`.

Agora, vamos ver como fica a aproximação da função Seno por sua série de Taylor, no Matlab:

$$sen(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + ...$$

```
% Funcao para calcular o seno de um numero atraves da ...
   serie de taylor
function [output] = senoTaylor(numero, numero_termos)
  calcular = 1;
  termo = 0;
  output = 0;

  while (calcular)
      termoAtual = numero^(2*termo+1) * (-1)^(termo) / ...
          factorial(2*termo+1);
      termo = termo + 1;
      if (termo == numero_termos)
          calcular = 0;
      end
      output = output + termoAtual;
  end
end

% versao vetorizada da funcao
% o termo ((2)*mod( (1:numero_termos), 2) - 1) gera um ...
   vetor alternado de [1, -1, 1, -1 ...]
% o termo factorial(1:2:2*numero_termos) gera um vetor ...
   de mesmo tamanho do termo anterior, que contem o ...
   fatorial dos "numero_termos" primeiros numeros impares
% Ao efetuar ./, criamos um vetor com tamanho ...
   numero_termos que carrega os coeficientes de taylor ...
   da nossa funcao
% Por fim, efetuamos o produto interno deste vetor com o ...
   vetor de "numero" elevao aos "numero_termos" ...
   primeiros numeros impares. Isso efetua a ...
   multiplicacao de cada potencia com seu coeficiente e ...
   executa a soma dos resultados
function [saida] = senoTaylor_vect(numero, numero_termos)
  saida = ((2)*mod( (1:numero_termos), 2) - 1) ./ ...
   factorial(1:2:2*numero_termos) * ...
   (numero.^(1:2:2*numero_termos))';
end
```

1.5.1 Funções Úteis

```
help      % comando usado para conseguir o manual de uma ...
   funcao ou comando

input     % aceita uma entrada e calcula seu valor. Pode ...
   receber a entrada como string, caso especificado
disp      % Mostra uma string ou valor de variavel no ...
   console
sprintf   % Formata dados em uma string. A formatacao e ...
   igual a formatacao em C

% Para as funcoes abaixo, pode-se especificar em qual ...
   direcao se deseja executar a funcao. O padrao e ao ...
   longo de cada coluna
sum       % soma dos elementos de um vetor ou das ...
   colunas de uma matriz
prod      % produto dos elementos de um vetor ou das ...
   colunas de uma matriz
mean      % media dos elementos de um vetor ou das ...
   colunas de uma matriz
diff      % calcula as derivadas de ordem N de um vetor ...
   ou das colunas de uma matriz

% Funcoes relacionadas a matrizes
diag      % gera uma matriz diagonal com as dimensoes ...
   especificadas
ones      % gera uma matriz de 1's
zeros     % gera uma matriz de 0's
eye       % gera a matriz identidade
repmat    % cria uma matriz gerada atraves de repeticoes ...
   de uma matriz ou vetor de entrada, de modo a atingir ...
   as dimensoes especificadas
bsxfun    % recebe uma matriz A, um vetor B e executa ...
   uma funcao especificada, repetindo B ao longo de A. ...
   Ha tambem o equivalente cellfun
reshape   % muda o formato de uma matriz, mantendo seus ...
   elementos em ordem. Os elementos sao escolhidos ...
   coluna a coluna

% Funcoes relacionadas a graficos
hold      % toggle no estado que diz se a figura atual ...
   vai ser sobreescrita ou mantida no proximo grafico. ...
   Aceita a extensao hold on ou hold off.
```

```
figure    % cria uma nova figura para graficos. Aceita ...
   um parametro numerico
plot      % gera graficos bidimensionais
ezsurf    % forma facil de gerar graficos 3D. Pode-se ...
   especificar os simbolos utilizados com o comando syms ...
   "simbolo"
```

1.6 Exercícios

Para os exercicios a seguir, evitar usar loops sempre que possivel

1. Gerar uma lista dos N primeiros numeros primos usando a função isprime
2. Gerar um vetor-linha com números aleatórios entre 0 e N utilizando a função randi
3. Gerar um vetor-linha equivalente ao anterior através da função rand
4. Contar a quantidade de números pares e ímpares no vetor aleatório gerado anteriormente.
5. *(Extra)* Problema das 8 Rainhas - Solução não-recursiva

   Passo 1: Verificar se um dado tabuleiro de xadrez representado por uma matriz lógica 8x8 é solução do problema das 8 Rainhas.

   Passo 2: Caso seja um tabuleiro válido, guardá-lo em um cell.

   Dicas:

   - Para gerar os possíveis tabuleiros, pode-se usar a função perms(1:8), onde cada uma das permutações resultantes pode determinar a linha e a coluna nas quais deve-se posicionar uma rainha. Exemplo: o vetor [1 3 2 4 5 7 8 6] gera a matriz de zeros com 1's nas posições: (1,1), (2,3), (3,2), (4,4), (5, 5), (6, 7), (7, 8), (8, 6)
   - Pode-se eliminar uma grande quantidade de tabuleiros gerados segundo o método proposto encontrando números adjacentes, que simbolizam rainhas imediatamente diagonais. A função diff() é de grande ajuda aqui.

- O método proposto gera apenas tabuleiros com uma rainha por linha e por coluna. Só é necessário verificar as diagonais.
- Utilizar as funções ones() e diag() pode ajudar na verificação das diagonais. Outra possibilidade é utilizar os indices lineares da matriz para verificar as diagonais. Para as diagonais ortogonais à diagonal principal, basta inverter as colunas da matriz e checar as diagonais novamente.
- Só é necessário utilizar loops para percorrer a lista de permutações e para verificar as diagonais. Cada diagonal pode ser verificada em uma iteração, juntamente com sua diagonal complementar (a diagonal equivalente na matriz com colunas invertidas descrita no item anterior).

2 Mínimos Quadrados e o Ajuste Polinomial

O Método dos Mínimos Quadrados tem como objetivo encontrar a solução que melhor satisfaz a um sistema de equações lineares. Este método pode ser usado para se encontrar uma função que sirva de aproximação para um dado conjunto de pontos. Comumente, essa função é linear do tipo:

$$f(x) = ax + b$$

Para tal aproximação, criamos um sistema de equações baseado em um conjunto de pontos do tipo (x_n, y_n) e na função polinomial f(x) de grau n. Então, representamos o sistema de equações através da seguinte equação:

$$X\theta = y \tag{2.0.0}$$

2.1 Mínimos Quadrados e a Equação Normal

Na equação (2.0.0), a matriz $X_{M\times N}$ carrega em cada uma de suas colunas uma potência de x_n, enquanto a matriz θ carrega os coeficientes θ_n de cada grau e a matriz y, os y_n correspondentes a cada linha:

$$\begin{array}{cc} \text{Grau:} & \begin{array}{ccccc} 0 & 1 & 2 & ... & n \end{array} \\ & \begin{bmatrix} 1 & x_{12} & x_{13} & ... & x_{1n} \\ 1 & x_{22} & x_{23} & ... & x_{2n} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ 1 & x_{m2} & x_{m3} & ... & x_{mn} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ . \\ . \\ . \\ y_n \end{bmatrix} \end{array}$$

Agora, vamos modificar a equação para chegarmos à solução de Mínimos Quadrados:

$$X^T X\theta = X^T y$$

Multiplicando por $(X^T X)^{-1}$:

$$\theta = (X^T X)^{-1} X^T y \tag{2.1.1}$$

A equação (2.1.1) é a Equação Normal, cuja solução é a solução de mínimos quadrados do sistema. Nela, a matriz θ contém os coeficientes da função polinomial que melhor se ajusta aos pontos (x_n, y_n) dados.

É importante salientar que esta não é a forma mais eficiente de se chegar à solução de Mínimos Quadrados. Contudo, ela é de fácil implementação e é dela que partiremos para a primeira tarefa:

2.1.1 Cálculo dos Mínimos Quadrados através da Equação Normal

Crie uma função com o protótipo abaixo, que encontre a matriz θ que soluciona a equação $X\theta = Y$ através da equação normal (2.1.1).

```
function [theta] = leastSquares_normalEq(X, y)
```

Agora que vimos como método funciona, partiremos para a implementação mais eficiente, implementada através de *Gradient Descent*, do Inglês, Descida de Gradiente.

2.2 Mínimos Quadrados através de *Gradient Descent*

Gradient Descent é um método que faz uso do gradiente da função $f(x)$ que se deseja modelar para encontrar a solução de Mínimos Quadrados, que se traduz em um mínimo local da função. Para encontrar esse ponto de mínimo, associamos ao algoritmo uma função de custo:

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(f(x^i) - y^i)^2 \tag{2.2.1}$$

$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + ... + \theta_n x^n$

A função $J(\theta)$ é a função de custo que desejamos minimizar (encontrar um mínimo local seu).

A minimização de (2.2.1) se dará pela seguinte regra de atualização:

$$\theta_j = \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}(f(x^i) - y^i)x_j^i \tag{2.2.2}$$

θ_j é o elemento j da matriz-coluna θ

Sobre a equação (2.2.2), é importante salientar que a atualização de todos os θ_j deve ser feita simultaneamente.

2.2.1 Implementação de Mínimos Quadrados com *Gradient Descent*

Implemente a função com o protótipo abaixo, que utilize o método de *Gradient Descent*, através da equação (2.2.2)

```
function [theta] = leastSquares(X, y)

% Dica: inicialize theta com um valor aleatorio. ...
   Utilize a funcao size() para descobrir o tamanho ...
   que theta deve ter.
```

2.3 Apresentação gráfica dos resultados (*Plotting*)

Agora, vamos criar o gráfico dos pontos no plano cartesiano, o chamado *scatter plot*. Depois, vamos criar, na mesma figura, o gráfico da função que encontramos através do Ajuste Polinomial.

— *Seção incompleta* —

Referências

[1] Página de Sistemas Lineares da Mathworks:
`http://www.mathworks.com/help/control/examples/plotting-system-responses.html`

[2] Página sobre a linguagem e funcionalidades do Matlab:
`http://www.mathworks.com/help/matlab/index.html`

[3] Curso sobre Matlab (*Coursera*):
`https://www.coursera.org/course/matlab`

[4] Curso sobre Aprendizado de Máquina (*Coursera*):
`https://www.coursera.org/learn/machine-learning`