2 *Fundamentos de Regressão Linear*

Dada a importância do tema para a apresentação das propostas desenvolvidas nesta tese, neste capítulo são apresentados conceitos básicos sobre o problema de regressão linear simples e múltipla, bemo como sobre o problema de estimação de parâmetros através do método dos mínimos quadrados.

2.1 O Problema de Regressão Linear

Em muitos problemas práticos, há duas ou mais variáveis numéricas que parecem estar intrisicamente relacionadas, sendo necessário analisar a natureza matemática dessa relação de maneira mais formal a fim de entender melhor o problema. A análise de regressão é uma técnica estatística cujo objetivo principal reside justamente na investigação das relações entre duas ou mais variáveis e na consequente modelagem matemática do problema. A análise de regressão pode ser usada, por exemplo, na construção de um modelo que expresse o resultado de uma variável como função de uma ou mais variáveis. Esse modelo pode, então, ser usado para predizer o resultado de uma variável em função da outra (HINES et al., 2006).

Na descrição que se segue, assume-se que exista uma única variável dependente, ou de resposta, $y \in \mathbb{R}$, relacionada com p variáveis independentes, ou *regressoras*, $x_1, x_2, \ldots, x_p$, $x_j \in \mathbb{R}$. A variável de resposta y é uma variável aleatória, enquanto que as variáveis regressoras $x_1, x_2, \cdots, x_p$ são medidas com erro desprezível e são frequentemente controladas pelo experimentador (usuário). Isto posto, a relação entre y e as p variáveis regressoras é comumente escrita da seguinte forma:

$$y = f(x_1, x_2, \ldots, x_p|\beta) + \varepsilon, \tag{2.1}$$

$$= f(\mathbf{x}|\beta) + \varepsilon, \tag{2.2}$$

em que $f(\cdot|\cdot)$ é denominada de função de regressão, $\mathbf{x} \in \mathbb{R}^p$ é o vetor de variáveis regressoras, $\beta \in \mathbb{R}^p$ é o vetor de parâmetros da função regressora, e ε denota o erro (ruído) aleatório, de

média zero e variância σ_ε^2, presente na medição de y. Assume-se também que ε é uma variável aleatória independente, ou seja, amostras de ε são independentes entre si.

Note que o modelo descrito na Equação (2.1) é um modelo teórico, uma vez que, em geral, nem a função de regressão $f(\cdot|\cdot)$, nem a componente aleatória ε são conhecidas. A escolha da forma funcional da equação de regressão $f(\cdot)$ é feita com base em informação *a priori*, fruto de conhecimento prévio acerca do problema; ou então, através de experimentação com diferentes formas funcionais. A escolha da forma funcional mais adequada a um dado problema é feita (ou pelo menos deveria ser) através de rigoroso processo de análise dos resultados das predições para cada forma funcional escolhida.

Qualquer que seja a forma funcional da equação de regressão, o seu vetor de parâmetros β deve ser estimado. Para isso, faz-se necessário medir um conjunto de n valores de y e de suas variáveis regressoras $\{x_1, x_2, \ldots, x_p\}$:

$$(y_i, x_{i1}, x_{i2}, \cdots, x_{ip}), \qquad i = 1, \ldots, n, \tag{2.3}$$

ou, em forma condensada, faz-se necessário coletar n pares entrada-saída $(y_i, \mathbf{x}_i)$, $i = 1, \ldots n$.

A estimativa do vetor β é simbolizada como $\hat{\beta}$, sendo ela utilizada na seguinte equação para predizer novos valores da variável de resposta:

$$\begin{aligned} \hat{y} &= \hat{f}(x_1, x_2, \ldots, x_p|\hat{\beta}), & (2.4)\\ &= \hat{f}(\mathbf{x}|\hat{\beta}), & (2.5) \end{aligned}$$

em que $\hat{f}(\cdot|\cdot)$ denota uma aproximação da função de regressão do modelo teórico.

A análise de regressão é dita linear quando se assume que a relação matemática entre as variáveis de interesse é uma função linear de seus parâmetros. Neste caso, o modelo de regressão teórico passa a ser chamado modelo de *regressão linear múltipla*, sendo escrito como

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon & (2.6)\\ &= \beta^T \mathbf{x} + \varepsilon & (2.7) \end{aligned}$$

em que o vetor de parâmetros $\beta \in \mathbb{R}^{p+1}$ contém $p+1$ componentes, $\beta_j, j = 0, 1, \cdots, p$, chamadas genericamente de coeficientes de regressão. Como conseqüência, a primeira componente do vetor $\mathbf{x} \in \mathbf{R}^{p+1}$ é igual a 1, sendo as restantes as próprias variáveis regressoras $\{x_1, x_2, \ldots, x_p\}$.

Os modelos de regressão linear múltipla são usados, em geral, como funções aproximadoras, e a equação de regressão é ajustada ao conjunto de pares entrada saída $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$. Lembrando que a verdadeira relação funcional entre y e $x_1, x_2, \cdots, x_p$ é geralmente descon-

hecida, mas em muitos casos práticos o modelo de regressão linear apresenta-se como uma aproximação adequada (HINES et al., 2006). Nestes casos, a equação de predição passa então a ser escrita como

$$\begin{aligned} \hat{y} = \quad E[y|\mathbf{x},\hat{\beta}] &= \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \ldots + \hat{\beta}_p x_p, \\ &= \hat{\beta}^T \mathbf{x}, \end{aligned} \tag{2.8}$$

em que $E[y|\mathbf{x},\hat{\beta}]$ denota o valor esperado da variável de resposta y condicionado ao vetor de variáveis regressoras $\mathbf{x}$ e à estimativa do vetor de parâmetros $\hat{\beta}$. A Equação (2.8) define um hiperplano no espaço p-dimensional das variáveis regressoras x_j.

Além de sua simplicidade, uma vantagem do modelo linear reside na interpretação direta que pode ser dada ao parâmetro β_j, como representando a mudança esperada na resposta y por unidade de mudança em x_j quando todas as demais variáveis independentes $x_i (i \neq j)$ são mantidas constantes, ou seja, $\beta_j = \frac{dy}{dx_j}$. Isto permite identificar diretamente quais variáveis regressoras são mais relevantes para a variável de saída ou, dito de outra forma, quais variáveis regressoras influenciam mais a variável de resposta.

Quando o problema de regressão lienar envolve apenas uma única variável regressora, x, tem-se uma regressão linear simples. Neste caso, a relação matemática entre uma única variável de entrada x e uma variável de saída y é definida por uma reta, ou seja,

$$y = \beta_0 + \beta_1 x_1 + \varepsilon, \tag{2.9}$$

em que β_0 é o intercepto e β_1, a inclinação da reta. Consequentemente, a equação de predição para o problema de regressão linear simples é dada por

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1, \tag{2.10}$$

em que $\hat{\beta}_0$ e $\hat{\beta}_1$ são, respectivamente, as estimativas do intercepto e do coeficiente angular da reta de regressão.

Como o modelo de regressão linear múltipla é mais geral que o de regressão simples, todo o desenvolvimento teórico que se segue será feito com base nesta formulação do problema de regressão.

2.2 Estimador de Mínimos Quadrados Ordinário

Como já mencionado na seção anterior, os dados a serem usados para estimar o vetor de parâmetros β consistem de n observações do par entrada-saída $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$. Em palavras,

a i-ésima observação inclui uma resposta escalar y_i e o vetor de variáveis regressoras correspondente $\mathbf{x}_i = [x_{i1} \;\; x_{i2} \;\; \cdots \;\; x_{ip}]^T$, em que x_{ij} denota a i-ésima observação da j-ésima variável regressora.

Assim, para o modelo de regressão linear múltipla, a variável resposta é uma função linear das variáveis regressoras:

$$\begin{aligned} y_i &= \beta^T \mathbf{x}_i + \varepsilon_i, && (2.11) \\ &= \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i && (2.12) \end{aligned}$$

em que ε_i corresponde à i-ésima observação do erro aleatório. Para a formulação que se segue, assume-se que as seguintes suposições são verdadeiras:

1. Existem (muito) mais observações que incógnitas (i.e. $n \gg p$).
2. O erro ou ruído no modelo (ε) tem média 0 e variância σ_ε^2.
3. As observações $\{\varepsilon_i\}$ são não-correlacionadas.

Em forma expandida, o modelo de regressão mostrado na Equação (2.11) corresponde a um sistema de equações com n equações e $p+1$ incógnitas, ou seja

$$\begin{aligned} y_1 &= \beta_0 + \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_p x_{1p} + \varepsilon_1 \\ y_2 &= \beta_0 + \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_p x_{2p} + \varepsilon_2 \\ &\vdots \qquad\quad \vdots \qquad\quad \vdots \\ y_n &= \beta_0 + \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_p x_{np} + \varepsilon_n \end{aligned} \qquad (2.13)$$

O sistema de equações mostrado na Equação (2.13) pode ser escrito em notação matricial como

$$\mathbf{y} = \mathbf{X}\beta + \varepsilon, \qquad (2.14)$$

em que os vetores $\mathbf{y} \in \mathbb{R}^n$ e $\varepsilon \in \mathbb{R}^n$, assim como a matriz de regressão $\mathbf{X} \in \mathbf{R}^n \times \mathbf{R}^{p+1}$, são definidos como

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n\times 1}, \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}_{n\times(p+1)},$$

com o vetor de parâmetros $\beta \in \mathbb{R}^{p+1}$ e o vetor aleatório $\varepsilon \in \mathbb{R}^n$ sendo definidos por

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}_{(p+1)\times 1} \quad \text{e} \quad \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}_{n\times 1}.$$

Usar a técnica de mínimos quadrados para encontrar estimativas para os coeficientes de regressão β_j, $j = 1, \ldots, n$, corresponde a minimizar a seguinte função-custo:

$$J(\beta_1, \beta_2, \ldots, \beta_p) = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2. \tag{2.15}$$

Dessa forma, minimizar a função-custo $J(\beta_1, \beta_2, \ldots, \beta_p)$ equivale a fazer com que a soma dos quadrados dos desvios ε_i entre os valores observados de y_i e o hiperplano de regressão seja mínima. Em forma vetorial, a função-custo $J(\beta_1, \beta_2, \ldots, \beta_p)$ pode ser escrita como

$$J(\beta) = \|\varepsilon\|^2 = \varepsilon^T \varepsilon = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta), \tag{2.16}$$

em que $\|\cdot\|$ denota a norma euclidiana de um vetor

A forma vetorial permite também adicionar uma outra interpretação ao problema de minimizar a função-custo $J(\beta)$: minimizar esta função-custo corresponde a encontrar uma estimativa do vetor de parâmetros β que produza o vetor de erros aleatórios com menor norma quadrática.

As expressões para cálculo das estimativas de β, denotadas como $\hat{\beta}$, podem ser obtidas a partir da minização da função-custo dos erros quadráticos na forma escalar (Equação 2.15) ou na forma vetorial (Equação 2.16). Ambos os casos são apresentados a seguir.

2.2.1 Equações Normais dos Mínimos Quadrados (Forma Escalar)

A função $J(\beta_1, \beta_2, \ldots, \beta_p)$ deve ser minimizada individualmente em relação a cada um dos parâmetros $\beta_0, \beta_1, \cdots, \beta_p$, ou seja, parâmetro a parâmetro. Para isso, a derivada parcial de $J(\beta_1, \beta_2, \ldots, \beta_p)$ deve ser tomada em relação a cada parâmetro β_j, $j = 1, \ldots, p$ e igualada a zero, ou seja

$$\frac{\partial J}{\partial \beta_0} = -2 \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j x_{ij} \right) = 0 \tag{2.17}$$

e

$$\frac{\partial J}{\partial \beta_j} = -2\sum_{i=1}^{n} x_{ij}\left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_{ij}\right) = 0, \qquad j = 1,2,\ldots,p. \tag{2.18}$$

Resolvendo as Equações (2.17) e (2.18), obtemos um sistema de equações conhecido como *equações normais de mínimos quadrados*, em sua forma escalar:

$$n\hat{\beta}_0 + \hat{\beta}_1\sum_{i=1}^{n} x_{i1} + \hat{\beta}_2\sum_{i=1}^{n} x_{i2} + \cdots + \hat{\beta}_p\sum_{i=1}^{n} x_{ip} = \sum_{i=1}^{n} y_i, \tag{2.19}$$

$$n\hat{\beta}_0\sum_{i=1}^{n} x_{i1} + \hat{\beta}_1\sum_{i=1}^{n} x_{i1}^2 + \hat{\beta}_2\sum_{i=1}^{n} x_{i1}x_{i2} + \cdots + \hat{\beta}_p\sum_{i=1}^{n} x_{i1}x_{ip} = \sum_{i=1}^{n} x_{i1}y_i, \tag{2.20}$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$n\hat{\beta}_0\sum_{i=1}^{n} x_{ip} + \hat{\beta}_1\sum_{i=1}^{n} x_{ip}x_{i1} + \hat{\beta}_2\sum_{i=1}^{n} x_{ip}x_{i2} + \ldots + \hat{\beta}_p\sum_{i=1}^{n} x_{ip}^2 = \sum_{i=1}^{n} x_{ip}y_i. \tag{2.21}$$

Note que existem $p+1$ equações normais, uma para cada coeficiente de regressão; logo, o sistema acima é quadrado. A solução das equações normais produz as estimativas de mínimos quadrados dos coeficientes de regressão $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_p$ (HINES et al., 2006).

2.2.2 Equações Normais dos Mínimos Quadrados (Forma Vetorial)

Para o caso vetorial, a função-custo $J(\beta)$ mostrada na Equação (2.16) precisa primeiro ser decomposta da seguinte forma:

$$\begin{aligned} J(\beta) &= = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta), && (2.22)\\ &= \mathbf{y}^T\mathbf{y} - \beta^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta + \beta^T\mathbf{X}^T\mathbf{X}\beta, && (2.23)\\ &= \mathbf{y}^T\mathbf{y} - 2\beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta, && (2.24) \end{aligned}$$

em que se fez uso do fato de o termo $\beta^T\mathbf{X}^T\mathbf{y}$ ser um escalar e, portanto, seu transposto $\beta^T\mathbf{X}^T\mathbf{y}^T = \mathbf{y}^T\mathbf{X}\beta$ resulta no mesmo escalar.

Portanto, para minimizar o funcional $J(\beta)$, deve-se tomar a sua derivada parcial em relação ao vetor de parâmetros β e igualá-la ao vetor nulo $\mathbf{0} \in \mathbb{R}^{p+1}$, ou seja

$$\frac{\partial J}{\partial \beta} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\hat{\beta} = \mathbf{0}, \tag{2.25}$$

que, simplificando, resulta em

$$\mathbf{X}^T\mathbf{X}\hat{\beta} = \mathbf{X}^T\mathbf{y}, \tag{2.26}$$

que corresponde à versão vetorial das equações normais dos mínimos quadrados mostradas nas

Equações (2.19) a (2.21).

Para encontrar a solução das equações normais, multiplica-se ambos os lados da Equação 2.26 pela inversa de $\mathbf{X}^T\mathbf{X}$. Assim, a estimativa de mínimos quadrados ordinário (MQO) do vetor de parâmetros β é dada por

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \tag{2.27}$$

que também é conhecida como solução pelo método da pseudoinversa de Moore-Penrose (GOLUB; VAN LOAN, 1996).

O modelo de predição linear baseado na estimativa MQO do vetor de parâmetros é dado por

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta}, \tag{2.28}$$

em que $\hat{\mathbf{y}} \in \mathbb{R}^n$ é o vetor de predições da variável resposta. O vetor de erros de predição é então dado por $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$, sendo que a sua norma quadrática é a menor possível $\|\mathbf{e}\|^2$, segundo o critério dos mínimos quadrados.

Na notação escalar, a equação de predição é dada por

$$\hat{y}_i = \hat{\beta}^T\mathbf{x}_i = \hat{\beta}_0 + \sum_{j=1}^{p}\hat{\beta}_j x_{ij}, \tag{2.29}$$

para $i = 1, 2, \ldots, n$, e o erro correspondente é dado por $e_i = y_i - \hat{y}_i$. A soma dos erros quadráticos (SEQ), $\sum_{i=1}^{n} e_i^2$, que nada mais é do que a norma quadrática $\|\mathbf{e}\|^2$, é comumente usada como critério de avaliação da qualidade da regressão.

2.3 Estimador de Mínimos Quadrados Regularizado

Muitas vezes, a matriz $\mathbf{X}^T\mathbf{X}$ é singular (i.e. não é de posto completo) ou muito próxima da singularidade. Neste caso, tem-se que

$$det(\mathbf{X}^T\mathbf{X}) \approx 0, \tag{2.30}$$

fato este que pode comprometer toda a validade do processo de inferência da regressão linear múltipla, pois é fonte de instabilidades numéricas durante o cálculo da estimativa MQO do vetor de parâmetros mostrada na Equação (2.27). Isto ocorre geralmente quando as variáveis de entrada são intercorrelacionadas. Quando essa intercorrelação é grande, dizemos que existe *multicolinearidade*, ou seja, as linhas da matriz $\mathbf{X}^T\mathbf{X}$ não são linearmente independentes.

A fim de evitar problemas numéricos, faz-se necessário utilizar estratégias que permitam estimar β de modo confiável. Para este fim, um dos métodos mais conhecidos é o *método de regularização de Thikonov* (HOERL; KENNARD, 1970). Utilizando regularização de Thikonov, o estimador de mínimos quadrados de β é dado por

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.31}$$

em que

- $0 \leq \lambda \ll 1$ é uma constante de valor bem pequeno, e
- $\mathbf{I}$ é uma matriz identidade de dimensão $(p+1) \times (p+1)$.

A regressão que utiliza esse tipo de regularização é chamada de regressão de cumeeira (*ridge regression*) e a função-custo associada é dada por

$$J(\beta) = \|\varepsilon\|^2 + \lambda\|\beta\|^2, \tag{2.32}$$

que permite interpretar a Equação (2.31) como aquela que produz uma estimativa do vetor de parâmetros $\hat{\beta}$ que tenta satisfazer dois critérios de otimalidade:

- Um que procura minimizar a norma do erro quadrático, ou seja, $\|\mathbf{e}\|^2$,
- E um outro que procura minimizar a norma do vetor de parâmetros, ou seja, $\|\beta\|^2$,

tal que a importância do segundo critério frente ao primeiro é regulada pelo valor de λ.

2.4 Regressão Linear no Matlab

A versão vetorial da solução dos mínimos quadrados (Equação (2.27)), assim como a sua versão regularizada mostrada na Equação (2.31), são úteis não apenas pela notação matemática compacta, mas principalmente porque sua implementação em ambientes de computação científica, tais como Matlab©, Octave e Scilab, é possível sem maior esforço de programação.

Como estes ambientes são amplamente utilizados em Engenharia e Ciências, existem diversos comandos e funções que geram, em princípio, os mesmos resultados numéricos para um dado problema de regressão linear simples ou múltipla. Contudo, alguns comandos são mais eficientes, seja porque consomem menos tempo, seja porque são menos susceptíveis a erros numéricos. Isto posto, vale a pena fazer alguns comentários sobre este tema e sugerir

formas eficientes de implementação das equações de estimação de parâmetros pelo método dos mínimos quadrados, ordinário ou regularizado.

Assumindo que o vetor de observações da variável de resposta $\mathbf{y}$ e a matriz de variáveis regresssoras $\mathbf{X}$ são denotadas como `y` e `X` no Matlab, então a Equação (2.27) pode ser implementada da forma que se lê, ou seja,

```
» B = inv(X'*X)*X'y;
```

em que `B` denota a estimativa MQO do vetor de parâmetros β. Contudo, esta forma de se estimar β não é a mais recomendada para problemas de maior escala por duas razões: (*i*) Possui elevado custo computacional, e (*ii*) é muito susceptível a erros numéricos.

Para problemas maiores recomenda-se usar o operador *barra invertida* (\), ou seja

```
» B = X\y;
```

Os cálculos realizados pelo uso do operador *barra invertida* (\) baseiam-se em grande parte no método de ortogonalização conhecido como fatoração QR.

Uma segunda maneira de estimar o vetor β é através do comando `PINV`:

```
» B = pinv(X)*y;
```

Embora a obtenção da solução via comando `pinv` seja computacionalmente mais custosa que a obtida pelo uso do operador *barra invertida*, visto que é baseada na técnica conhecida como SVD (*singular value decomposition*), ela é preferível em problemas em que a matriz de variáveis regressoras $\mathbf{X}$ possui deficiência de posto (i.e. tem posto incompleto).

Por fim, uma terceira maneira de se estimar o vetor β no Matlab é por meio do comando `REGRESS`:

```
» B = regress(y,X);
```

Para a versão regularizada do estimador MQO, também é possível estimar $\hat{\beta}$ através da escrita direta da Equação (2.31) no prompt do Matlab:

```
» l = 0.01;
» I=ones(size(X'*X));
» B = inv(X'*X + l*I)*X'y;
```

Porém, pelas mesmas razões apontadas anteriormente, recomenda-se também usar o operador *barra invertida* (\). Neste caso, a seqüência de comandos passa ser a seguinte:

```
» l = 0.01;
» I=eye(size(X'*X));
» A=X'*X + l*I;
» r=X'*y;
» B = A\r;
```

Um exemplo de aplicação dos comandos acima em um problema real será apresentado logo a seguir. Este exemplo, além de ser útil do ponto de vista didático, será retomado no próximo capítulo, quando abordaremos o problema de regressão robusta.

2.4.1 Exemplo Numérico: Regressão Linear Simples

A título de ilustração, vamos aplicar os conceitos de regressão linear apresentados neste capítulo ao conjunto de dados mostrado na Tabela 2.1. Este conjunto está disponível em Freedman et al. (2007) e envolve duas variáveis: uma é variável regressora $x \in \mathbb{R}$ (consumo per capita de cigarros em um dado país em 1930), enquanto a variável resposta $y \in \mathbb{R}$ corresponde ao número de mortes (por milhão de pessoas) por câncer de pulmão naquele país em 1950.

Índice	País	Cigarro per capita	Mortes por milhão de pessoas
1	Austrália	480	180
2	Canadá	500	150
3	Dinamarca	380	170
4	Finlândia	1100	350
5	Grã Bretanha	1100	460
6	Islândia	230	60
7	Holanda	490	240
8	Noruega	250	90
9	Suécia	300	110
10	Suíça	510	250

Tabela 2.1: Consumo per capita de cigarros em vários países em 1930 e as taxas de morte por câncer de pulmão em 1950.

Como o conjunto de dados envolve apenas um variável regressora é possível visualizar os pares (x_i, y_i), $i = 1, \ldots, 10$ em um diagrama de dispersão (*scatterplot*), marcando um círculo em cada coordenada, conforme ilustrado na Figura 2.1.

Pode-se perceber pelo diagrama de dispersão que há uma tendência linear nos dados, ou seja, há uma tendência de os pontos se organizarem ao longo de uma reta hipotética. Esta reta é, na verdade, a reta de regressão, cujos parâmetros podem ser calculados facilmente usando os

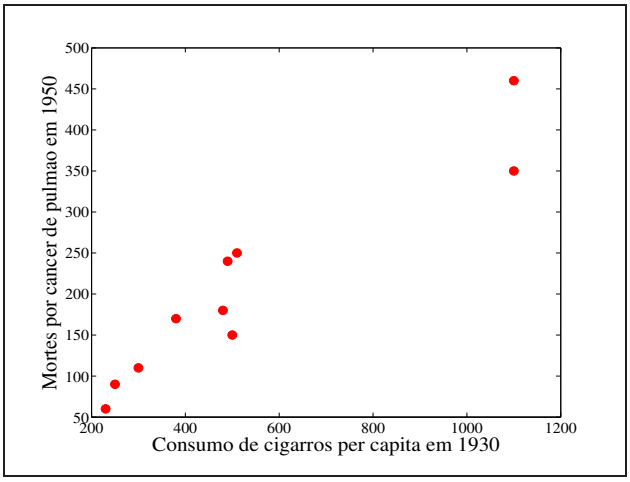


Figura 2.1: Diagrama de dispersão para o conjunto de dados da Tabela 2.1.

comandos do Matlab discutidos na seção anterior. Para isso, podemos usar a seguinte seqüência de comandos:

```
» x=[480; 500; 380; 1100; 1100; 230; 490; 250; 300; 510];
» y=[180; 150; 170; 350; 460; 60; 240; 90; 110; 250];
» n=length(x);
» X=[ones(n,1) x];
» B=X\y
B=
9.1393
0.3687
```

Assim, tem-se que $\hat{\beta}_0 = 9{,}14$ e $\hat{\beta}_1 = 0{,}37$, com a equação da reta de regressão sendo dada por $\hat{y}_i = 9,14 + 0,37x$. O gráfico da reta de regressão superposta aos pontos do conjunto de dados é mostrado na Figura 2.2.

No próximo capítulo, este mesmo conjunto de dados, adicionado de mais um par de pontos (x_i, y_i) relativo ao consumo de cigarros nos Estados Unidos, será usado para introduzir conceitos de regressão robusta, ou seja, na presença de pontos discrepantes (*outliers*).

2.5 Resumo do Capítulo

Esse capítulo apresentou os conceitos fundamentais sobre o problema de regressão linear, principalmente o de regressão linear múltipla. A análise de regressão trata da modelagem e investigação das relações entre duas ou mais variáveis, ou seja, da relação entre uma variável

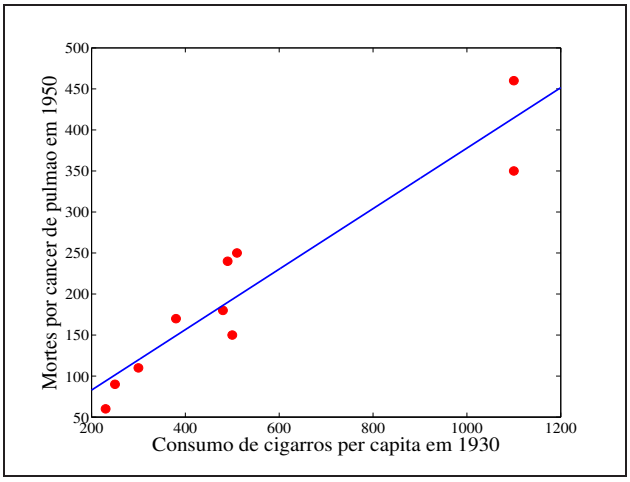


Figura 2.2: Gráfico da reta de regressão ajustada aos dados da Tabela 2.1.

resposta (ou de saída), e uma ou mais variáveis regressoras (independentes). A regressão linear simples se dá quando existe apenas uma variável regressora, e é definida por uma reta, enquanto que a regressão linear múltipla envolve mais de uma variável regressora, sendo definida por um plano.

Para estimar os parâmetros do modelo de regressão linear apresentado neste capítulo foi utilizado o método dos mínimos quadrados ordinário, bem como uma variante regularizada deste método. Além disso, foram apresentadas e discutidas diferentes meios para se implementar o método dos mínimos quadrados, ordinário e regularizado, em ambientes de computação científica, tais como Matlab©e Octave, tendo em vista questões relacionadas a problemas de natureza numérica.

Por fim, um conjunto de dados real foi utilizado com o propósito de ilustrar os conceitos introduzidos neste capítulo, principalmente para mostrar que, graças à formulação vetorial do estimador dos mínimos quadrados, a implementação deste método no Matlab é bem simples e direta.

No próximo capítulo este mesmo conjunto de dados, adicionado de mais um par de pontos (x_i, y_i) relativo ao consumo de cigarros nos Estados Unidos, será usado para introduzir conceitos de regressão robusta, ou seja, na presença de pontos discrepantes (*outliers*).

Referências Bibliográficas

ANDERSON, J. A simple neural network generating an interactive memory. *Mathematical Biosciences*, v. 14, n. 3-4, p. 197–220, 1972.

ANDREWS, D. F. A robust method for multiple linear regression. *Technometrics*, v. 16, n. 4, p. 523–531, 1974.

AUGUSTEIJN, M. F.; FOLKERT, B. A. Neural network classification and novelty detection. *International Journal of Remote Sensing*, v. 23, n. 14, p. 2891–2902, 2002.

BADDELEY, R. et al. Responses of neurons in primary and inferior temporal visual cortices to natural scenes. In: *Proc. R. Soc. Lond.* [S.l.: s.n.], 2005. p. 1775–1783.

BAEK, D.; OH, S.-Y. Improving optimal linear associative memory using data partitioning. In: *Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics (SMC'06)*. [S.l.: s.n.], 2006. v. 3, p. 2251–2256.

BAI, Z. D.; WU, Y. General M-estimation. *Journal of Multivariate Analysis*, v. 63, p. 119–135, 1997.

BARNETT, V.; LEWIS, T. *Outliers in Statistical Data*. 3a.. ed. [S.l.]: John Wiley & Sons, 1994.

BARRETO, G. A. *Redes Neurais Não-Supervisionadas para Processamento de Seqüências Temporais*. Dissertação (Mestrado) — Departamento de Engenharia Elétrica, Universidade de São Paulo, São Carlos, SP, 1998.

BARRETO, G. A.; FROTA, R. A. A unifying methodology for the evaluation of neural network models on novelty detection tasks. *Pattern Analysis and Applications*, v. 16, n. 1, p. 83–972, 2013.

BELLHUMER, P. N.; HESPANHA, J.; KRIEGMAN, D. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Special Issue on Face Recognition*, v. 17, n. 7, p. 711–720, 1997.

BEN-GAL, I. Outlier detection. In: MAIMON, O.; ROKACH, L. (Ed.). *Data Mining and Knowledge Discovery Handbook*. [S.l.]: Springer, 2005. p. 131–146.

BLATNÁ, D. Outliers in regression. In: *Proceedings of the 9th International Scientific Conference on Applications on Mathematics and Statistics in Economy (AMSE'2006)*. [S.l.: s.n.], 2006. p. 1–6.

BOCCATO, L. *Novas Propostas e Aplicaçõs de Redes Neurais com Estados de Eco*. Tese (Doutorado) — Faculdade de Engenharia Elétrica e de Computação (FEEC), Universidade de Campinas, São Paulo, 2013.

BOULESTEIX, A.-L. PLS dimension reduction for classification with microarray data. *Statistical Applications in Genetics and Molecular Biology*, v. 3, n. 1, p. 1–32, 2004.

BRANHAM, R. L. Alternatives to least squares. *The Astronomical Journal*, v. 87, n. 6, p. 928–937, 1982.

BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. In: *Proceedings of the IEEE Swarm Intelligence Symposium*. Honolulu, Hawaii: [s.n.], 2007. p. 120–127.

CHATTERJEE, S.; MÄCHLER, M. Robust regression: a weighted least squares approach. *Communications in Statistics - Theory and Methods*, v. 26, n. 6, p. 1381–1394, 1997.

CHERKASSKY, V.; FASSETT, K.; VASSILAS, N. Linear algebra approach to neural associative memories and noise performance of neural classifiers. *IEEE Transactions on Computers*, v. 40, n. 12, p. 1429–1435, 1991.

CUDMORE, R. H.; DESAI, N. S. Intrinsic plasticity. *Scholarpedia*, v. 3, n. 2, p. 1363, 2008.

DENG, W.; ZHENG, Q.; CHEN, L. Regularized extreme learning machine. In: *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM'09)*. [S.l.: s.n.], 2009. p. 389–395.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2nd. ed. [S.l.]: John Wiley & Sons, 2006.

EICHMANN, G.; KASPARIS, T. Pattern classification using a linear associative memory. *Pattern Recognition*, v. 22, n. 6, p. 733–740, 1989.

EMMERICH, C.; REINHART, F.; STEIL, J. Recurrence enhances the spatial encoding of static inputs in reservoir networks. In: *Proceedings of the 20th International Conference on Artificial Neural Networks*. [S.l.]: Springer, 2010. LNCS 6353, p. 148–153.

FAWZYA, A.; MOKHTAR, H. M.; HEGAZY, O. Outliers detection and classification in wireless sensor networks. *Egyptian Informatics Journal*, 2013.

FOX, J. *Applied Regression Analysis, Linear Models, and Related Methods*. [S.l.]: Sage Publications, 1997.

FOX, J. *Robust Regression: Appendix to An R and S-PLUS Companion to Applied Regression*. [S.l.], 2002.

FRANK, A.; ASUNCION, A. *UCI Machine Learning Repository*. 2010. Disponível em: <http://archive.ics.uci.edu/ml>.

FREEDMAN, D.; PISANI, R.; PURVES, R. *Statistics*. 4a.. ed. [S.l.]: W. W. Norton & Company, 2007.

GAUSS, C. F. *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium*. [S.l.]: Perthes et I. H. Besser, Hamburgi, 1809.

GLAVIN, F. G.; MADDEN, M. G. Analysis of the effect of unexpected outliers in the classification of spectroscopy data. In: *Proceedings of the 20th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'09)*. [S.l.: s.n.], 2010. p. 124–133.

GOLUB, G. H.; VAN LOAN, C. F. *Matrix Computations*. 3a.. ed. [S.l.]: The Johns Hopkins University Press, 1996.

HAMPEL, F. Robust statistics: a brief introduction and overview. In: *Robust Statistics and Fuzzy Techniques in Geodesy and GIS*. [S.l.: s.n.], 2001. p. 1–6.

HEBB, D. *The Organization of Behavior*. [S.l.]: New York: Wiley, 1949.

HILL, R. W.; HOLLAND, P. W. Two robust alternatives to least-squares regression. *Journal of the American Statistical Association*, v. 72, n. 360, p. 828–833, 1977.

HINES, W. W. et al. *Probabilidade e Estatística na Engenharia*. Quarta. [S.l.]: LTC, 2006.

HODGE, V. J.; AUSTIN, J. A survey of outlier detection methodologies. *Artificial Intelligence Review*, v. 22, n. 2, p. 85–126, 2004.

HOERL, A. E.; KENNARD, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, v. 42, n. 1, p. 80–86, 1970.

HORATA, P.; CHIEWCHANWATTANA, S.; SUNAT, K. Robust extreme learning machine. *Neurocomputing*, v. 102, p. 31–44, 2012.

HOWELL, A. J. *Automatic Face Recognition using Radial Basis Function Networks*. Tese (Doutorado) — University of Sussex, Brighton, UK, September 1997.

HUANG, G.-B.; WANG, D. H.; LAN, Y. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, v. 2, p. 107–122, 2011.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: Theory and applications. *Neurocomputing*, v. 70, p. 489–501, 2006.

HUBER, P. J. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, v. 35, n. 1, p. 73–101, 1964.

HUNT, B. et al. Synthesis of a nonrecurrent associative memory model based on a nonlinear transformation in the spectral domain. *IEEE Transactions on Neural Networks*, v. 4, n. 5, p. 873–878, 1993.

JOHNSON, W.; GEISSER, S. A predictive view of the detection and characterization of influential observations in regression analysis. *Journal of the American Statistical Association*, v. 78, p. 137–144, 1983.

KENNEDY, J.; EBERHART, R. C. Particle swarm optimization. In: *Proceedings of the IEEE International Conference on Neural Networks*. Piscataway, NJ, USA: [s.n.], 1995. v. 4, p. 1942–1948.

KIM, H.-C.; GHAHRAMANI, Z. Outlier robust gaussian process classification. In: *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition (SSPR)'08*. [S.l.: s.n.], 2008. p. 896–905.

KOHONEN, T. Correlation matrix memory. *IEEE Transactions on Computers*, C-21, n. 4, p. 353–359, 1972.

KOHONEN, T.; OJA, E. Fast adaptive formation of orthogonalizing filters and associative memory in recurrent networks of neuron-like elements. *Biological Cybernetics*, v. 25, p. 85–95, 1976.

KOHONEN, T.; RUOHONEN, M. Representation of associated data by matrix operators. *IEEE Transactions on Computers*, v. 22, n. 7, p. 701–702, 1973.

LEE, C.-C. et al. Noisy time series prediction using *m*-estimator based robust radial basis function neural networks with growing and pruning techniques. *Expert Systems and Applications*, v. 36, n. 3, p. 4717–4724, 2009.

LEE, C.-C. et al. Robust radial basis function neural networks. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, v. 29, n. 6, p. 674–685, 1999.

LEE, D. D.; SEUNG, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature*, v. 401, p. 788–791, 1999.

LEE, D. D.; SEUNG, H. S. Algorithms for non-negative matrix factorization. In: PRESS, M. (Ed.). *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*. [S.l.: s.n.], 2000. p. 556–562.

LEGENDRE, A. M. *Nouvelles Méthodes pour la Détermination des Orbites des Comètes*. [S.l.]: Courcier, Paris, 1805.

LI, C. A model of neuronal intrinsic plasticity. *IEEE Transactions on Autonomous Mental Development*, v. 3, n. 4, p. 277–284, 2011.

LI, D.; HAN, M.; WANG, J. Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems*, v. 23, n. 5, p. 787–799, 2012.

LIU, N.; WANG, H. Ensemble based extreme learning machine. *IEEE Signal Processing Letters*, v. 17, n. 8, p. 754–757, 2010.

MARONNA, R. A.; MARTIN, D. R.; YOHAI, V. J. *Robust Statistics: Theory and Methods*. 1a.. ed. [S.l.]: John Wiley & Sons, 2006.

MESQUITA, M. E. R. V. Introdução às memórias associativas lineares, morfológicas e fuzzy. In: *Anais do 2o. Colóquio de Matemática da Região Sul (ColMatSul'2012*. [S.l.: s.n.], 2012.

MICHE, Y. et al. OP-ELM: Optimally pruned extreme learning machine. *IEEE Transactions on Neural Networks*, v. 21, n. 1, p. 158–162, 2010.

MICHE, Y. et al. TROP-ELM: a double-regularized ELM using LARS and Tikhonov regularization. *Neurocomputing*, v. 74, n. 16, p. 2413–2421, 2011.

MOHAMMED, A. et al. Human face recognition based on multidimensional PCA and extreme learning machine. *Pattern Recognition*, v. 44, n. 10–11, p. 2588–2597, 2011.

NAKANO, K. Associatron: A model of associative memory. *IEEE Transactions on on Systems, Man, Cybernetics*, SMC-2, n. 3, p. 380–388, 1972.

NETO, A. R. R.; BARRETO, G. A. On the application of ensembles of classifiers to the diagnosis of pathologies of the vertebral column: A comparative analysis. *IEEE Latin America Transactions*, v. 7, n. 4, p. 487–496, 2009.

NEUMANN, K.; STEIL, J. Optimizing extreme learning machines via ridge regression and batch intrinsic plasticity. *Neurocomputing*, v. 102, p. 23–30, 2013.

OES, R. S. B.; LIMA, V. M. C. Comparação de estimadores de regressão. In: *Anais do XIX Simpósio Nacional de Probabilidade e Estatística (SINAPE'2010)*. [S.l.: s.n.], 2010. p. 1–6.

PEDERSEN, M. E. H.; CHIPPERFIELD, A. J. Simplifying particle swarm optimization. *Applied Soft Computing*, v. 10, n. 2, p. 618–628, 2010.

POGGIO, T.; GIROSI, F. Networks for approximation and learning. *Proceedings of the IEEE*, v. 78, n. 9, p. 1481–1497, 1990.

PRASAD, B. et al. A study on associative neural memories. *International Journal of Advanced Computer Science and Applications*, v. 1, n. 6, p. 124–133, 2010.

RAMALHO, G. L. B.; MEDEIROS, F. N. S. Using boosting to improve oil spill detection in sar images. In: *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'2006)*. [S.l.: s.n.], 2006. v. 2, p. 1066–1069.

RAO, C. R.; TOUTENBURG, H. *Linear Models: Least Squares and Alternatives*. 2nd. ed. [S.l.]: Springer, 1999.

RIPLEY, B. D. *Robust Statistics*. [S.l.], 2004.

RITTER, G.; GALLEGOS, M. T. Outliers in statistical pattern recognition and an application to automatic chromosome classification. *Pattern Recognition Letters*, v. 18, n. 6, p. 525–539, 1997.

RONCHETTI, E. The historical development of robust statistics. In: *Proceedings of the 7th International Conference on Teaching Statistics (ICOTS-7)*. [S.l.: s.n.], 2006. p. 1–4.

ROUSSEEUW, P. J.; LEROY, A. M. *Robust Regression and Outlier Detection*. 1a.. ed. [S.l.]: John Wiley & Sons, 1987.

SCHONHOFF, T.; GIORDANO, A. *Detection and Estimation Theory*. 1a.. ed. [S.l.]: Prentice Hall, 2006.

SINGH, S.; MARKOU, M. An approach to novelty detection applied to the classification of image regions. *IEEE Transactions on Knowledge and Data Engineering*, v. 16, n. 4, p. 396–407, 2004.

SMITH, M. R.; MARTINEZ, T. Improving classification accuracy by identifying and removing instances that should be misclassified. In: *Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN'2011)*. [S.l.: s.n.], 2011. p. 2690–2697.

STANIMIROVA, I.; WALCZAK, B. Classification of data with missing elements and outliers. *Talanta*, v. 76, n. 3, p. 602–609, 2008.

STEVENS, J. P. Outliers and influential data points in regression analysis. *Psychological Bulletin*, v. 95, n. 2, p. 334–344, 1984.

STILES, G.; DENQ, D.-L. A quantitative comparison of the performance of three discrete distributed associative memory models. *IEEE Transactions on Computers*, v. 36, n. 3, p. 257–263, 1987.

STILES, G. S.; DENQ, D. On the effect of noise on the Moore-Penrose generalized inverse associative memory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 7, n. 3, p. 358–360, 1985.

TONG, T. T. *Diagnostics for Outliers and Influential Points*. [S.l.], 2010.

TRIESCH, J. A gradient rule for the plasticity of a neuron's intrinsic excitability. In: *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN)*. [S.l.: s.n.], 2005.

TUKEY, J. W. The future of data analysis. *Annals of Mathematical Statistics*, v. 33, n. 1, p. 1–67, 1964.

VASCONCELOS, G. C.; FAIRHURST, M. C.; BISSET, D. L. Investigating feedforward neural networks with respect to the rejection of spurious patterns. *Pattern Recognition Letters*, v. 16, p. 207–212, 1995.

WEBB, A. *Statistical Pattern Recognition*. 2. ed. [S.l.]: John Wiley & Sons, LTD, 2002.

ZONG, W.; HUANG, G.-B. Face recognition based on extreme learning machine. *Neurocomputing*, v. 74, n. 16, p. 2541–2551, 2011.

Representações de um Sistema de Equações Lineares

Prof. Dr. Guilherme de Alencar Barreto

Março/2018

Departmento de Engenharia de Teleinformática
Universidade Federal do Ceará (UFC), Fortaleza-CE

gbarreto@ufc.br

1 Objetivo

O objetivo maior destanota de aula é reforçar o entendimento de diversas representações matemáticas de um sistema de equações lineares, dada a importância deste tópico não só para a teoria de redes neurais artificiais de modo mais específico; como também para a teoria de reconhecimento de padrões, de um modo mais geral.

As primeiras arquiteturas de redes neurais artificiais que iremos estudar são modelos lineares e o entendimento do funcionamento destes é extremamente facilitado pelas representações matemáticas de sistemas de equações lineares que iremos apresentar ao longo deste documento. Além disso, arquiteturas neurais mais complexas, ou seja, aquelas de natureza não-linear, também usam tais representações, de modo que o entendimento do conteúdo deste capítulo é fundamental para grande parte das arquiteturas de redes neurais artificiais que iremos estudar em nossa disciplina.

Abordaremos a seguir a representação clássica de um sistema de equações lineares, chamada aqui de representação escalar. Em seguida, partiremos para uma representação vetor-matriz, que é bem mais compacta, facilitando a escrita de um sistema com muitas variáveis. Também apresentaremos uma representação gráfica de um sistema de equações lineares, que será interpretada mais adiante como uma arquitetura de rede neural linear de uma única camada com n entradas e m neurônios de saída. Por fim, discutiremos as intepretações geométricas decorrentes das diversas representações apresentadas.

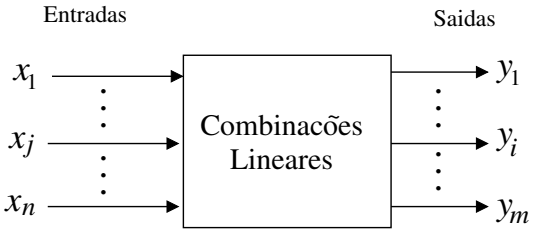


Figura 1: Representação do sistema de equações lineares mostrado em (1) como um diagrama de bloco.

2 Representação Clássica

Considere o seguinte sistema de equações lineares com m variáveis de saída, $y_i \in \mathbb{R}$, $i = 1, 2, \ldots, m$, e com n variáveis de entrada, $x_j \in \mathbb{R}$, $j = 1, 2, \ldots, n$:

$$\begin{aligned}
y_1 &= a_{10} + a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = a_{10} + \sum_{j=1}^{n} a_{1j}x_j \\
y_2 &= a_{20} + a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = a_{20} + \sum_{j=1}^{n} a_{2j}x_j \\
&\vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\
y_m &= a_{m0} + a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = a_{m0} + \sum_{j=1}^{n} a_{mj}x_j
\end{aligned} \tag{1}$$

em que os coeficientes $a_{ij} \in \mathbb{R}$, são chamados de coeficientes (ou parâmetros) do sistema e, para os nossos interesses, devem ser calculados a partir dos dados disponíveis usando algum método de estimação de parâmetros.

No jargão da área de redes neurais, é o processo de determinação destes coeficientes que chamamos de *aprendizado*, enquanto o método de estimação dos parâmetros em si é chamado de *regra* ou *algoritmo de aprendizado*.

Podemos representar o sistema de equações lineares mostrado em (1) por meio do diagrama de bloco mostrado na Figura 1, em que à esquerda do bloco indicamos as n variáveis de entrada e à direita, as m variáveis de saída.

O que deve ser entendido na Figura é que as n variáveis de entrada são combinadas linearmente, ou seja, são multiplicadas por constantes e depois somadas, para formar as m variáveis de saída.

A título de exemplo, vamos considerar um sistema de equações lineares com $n = 1$ (apenas uma variável de entrada) e $m = 2$ (duas variáveis de saída). Este sistema é escrito como

$$\begin{aligned}
y_1 &= a_{10} + a_{11}x_1 \\
y_2 &= a_{20} + a_{21}x_1
\end{aligned}$$

Note que a expressão $y_1 = a_{10} + a_{11}x_1$ é equivalente à equação de uma reta no plano cartesiano $X_1 \times Y_1$, em que a constante a_{10} define o intercepto (*bias*) e a constante a_{11} é chamada de coeficiente angular ou inclinação (*slope*) da reta. Raciocínio similar se aplica à equação $y_2 = a_{20} + a_{21}x_1$.

Vamos considerar agora um sistema de equações lineares com $n = m = 2$ (duas variáveis de entrada e duas de saída). Este sistema é escrito como

$$\begin{aligned} y_1 &= a_{10} + a_{11}x_1 + a_{12}x_2 \\ y_2 &= a_{20} + a_{21}x_1 + a_{22}x_2 \end{aligned}$$

Desta vez, podemos interpretar a expressão para y_1 como a equação de um plano no $\mathbb{R}^3$, representando pontos no espaço cartesiano $X_1 \times X_2 \times Y_1$. O mesmo raciocínio se aplica à equação para y_2.

De um modo mais geral, a i-ésima linha do sistema de equações lineares em (1), representada como

$$y_i = a_{i0} + a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n, \tag{2}$$

pode ser interpretada como a equação de um *hiperplano*[1] no $\mathbb{R}^{n+1}$.

3 Representação Vetor-Matriz

Uma forma muito útil de escrever o sistema de equações mostrado em (1) faz uso de uma notação vetor-matriz. Para este fim, precisamos antes fazer as seguintes definições:

1. Vetor de entrada $\mathbf{x} \in \mathbb{R}^{n+1}$, contendo as n variáveis de entrada.

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{(n+1)\times 1} \tag{3}$$

2. Vetor de saída $\mathbf{y} \in \mathbb{R}^m$, contendo as m variáveis de saída.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}_{m\times 1} \tag{4}$$

3. Matriz de transformação $\mathbf{A} \in \mathbb{R}^{m\times n}$, que "opera" sobre o vetor de entrada $\mathbf{x}$ para gerar o vetor $\mathbf{y}$.

$$\mathbf{A} = \begin{bmatrix} a_{10} & a_{11} & a_{12} & \cdots & a_{1n} \\ a_{20} & a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m0} & a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}_{m\times(n+1)} \tag{5}$$

Lembre-se que operações ou transformações lineares sobre um vetor podem alterar a sua norma (i.e. o comprimento) e/ou sua orientação (i.e. direção). Obviamente, se o sistema for quadrado (i.e. $n = m$) e a matriz $\mathbf{A}$ for a matriz identidade de ordem adequada, então nem

[1] O prefixo *hiper-* é usado para designar generalizações de figuras geométricas no $\mathbb{R}^3$ que ocorreriam em espaços de dimensão maior que 3, ou seja, $\mathbb{R}^4$, $\mathbb{R}^5$ e assim por diante.

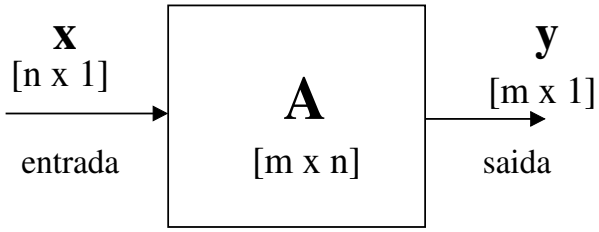


Figura 2: Representação do sistema linear $\mathbf{y} = \mathbf{A}\mathbf{x}$ em um diagrama de bloco.

a norma do vetor, nem sua orientação serão alteradas, pois a matriz identidade é o elemento neutro da multiplicação de matrizes.

Assim, o sistema de equações mostrado em (1) pode escrito como

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{6}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{10} & a_{11} & a_{12} & \cdots & a_{1n} \\ a_{20} & a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m0} & a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{7}$$

Com isto, o sistema $\mathbf{y} = \mathbf{A}\mathbf{x}$ passa a ser representada em diagrama de bloco conforme mostrado na Figura 2.

Prosseguindo com a notação vetor-matriz $\mathbf{y} = \mathbf{A}\mathbf{x}$, é possível perceber que cada linha da matriz $\mathbf{A}$ é formada por um conjunto de coeficientes que multiplicam cada componente do vetor $\mathbf{x}$. Assim, podemos escrever a i-ésima saída y_i, $i = 1, \ldots, m$, como

$$\begin{aligned} y_i &= [a_{i0} \;\; a_{i1} \;\; a_{i2} \;\; \cdots \;\; a_{in}]^T \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \\ &= a_{i0} + a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n, \\ &= \mathbf{a}_i^T \mathbf{x}, \end{aligned} \tag{8}$$

em que o vetor de coeficientes $\mathbf{a}_i \in \mathbb{R}^{n+1}$ tem a mesma dimensão do vetor de entrada $\mathbf{x}$. O sobrescrito T denota o vetor transposto.

Importante!! - Perceba que a notação $y_i = \mathbf{a}_i^T \mathbf{x}$ permite entender a i-ésima linha do vetor $\mathbf{y}$ como o resultado do *produto escalar* entre o vetor de coeficientes $\mathbf{a}_i$ e o vetor de entrada $\mathbf{x}$.

4 Representação Arquitetural

Vimos nas seções anteriores que a i-ésima linha do sistema de equações em (1) pode ser escrita matematicamente de três maneiras diferentes, porém equivalentes.

1. **Forma expandida da equação do hiperplano**:

$$y_i = a_{i0} + a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \tag{9}$$

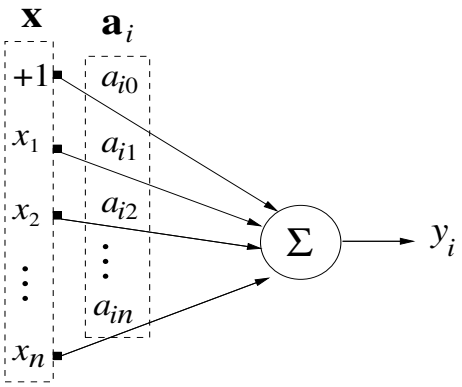


Figura 3: Representação arquitetural da Eq. (2) conhecida como *combinador linear*.

2. **Forma compacta da equação do hiperplano**:

$$y_i = a_{i0} + \sum_{j=1}^{n} a_{ij} x_j \tag{10}$$

3. **Produto-escalar entre $\mathbf{a}_i$ e $\mathbf{x}$**:

$$y_i = \mathbf{a}_i^T \mathbf{x} \tag{11}$$

Para finalizar esta nota de aula, vamos mostrar a chamada *representação arquitetural* da i-ésima linha do sistema de equações mostrado em (1). Esta representação, chamada genericamente de *combinador linear*, está mostrada na Figura 3

Em outras notas de aula, veremos que o combinador linear será usado como base teórica para um dos modelos mais usados até hoje de um neurônio artificial: o neurônio de McCulloch-Pitts [1].

A seguir vamos usar os conhecimentos de Álgebra Linear discutidos até agora nesta nota de aula para projetar um sistema computacional capaz de aprender as funções lógicas AND e OR.

5 Portas AND/OR e o Neurônio de McCulloch-Pitts

Em geral, nos cursos de Engenharia[2], o primeiro contato do estudante com os princípios da lógica booleana dá-se na disciplina de eletrônica digital ou equivalentes. Neste sentido, é de amplo conhecimento deste estudante as tabelas-verdades das funções lógicas AND e OR.

Isto posto, cabe aqui a seguinte pergunta:

> Qual a relação entre portas lógicas e Inteligência Computacional?

Talvez uma olhada no título do livro de George Boole[3] (vide Figura 4) mostrado a seguir possa ajudar a responder tal questão.

> George Boole (1854). *An investigation into the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities*. Cambridge: MacMillan and Co.

[2]Refiro-me especificamente às Engenharias Elétrica, Eletrônica, de Telecomunicações, de Computação, Automação e Controle, Mecatrônica e Biomédica

[3]`https://archive.org/details/investigationofl00boolrich`

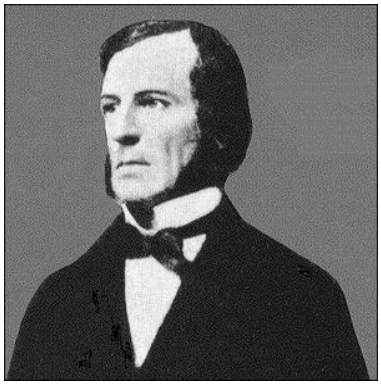


Figura 4: George Boole (02/11/1815 - 08/12/1864). Matemático e filósofo britânico. É o criador da Álgebra Booleana, base da atual aritmética computacional.

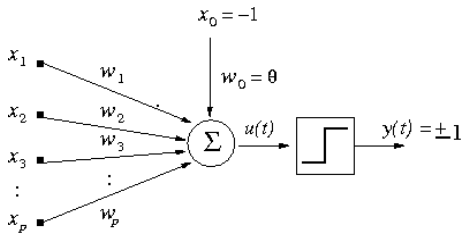


Figura 5: Representação esquemática moderna do modelo do neurônio de McCulloch-Pitts.

Note que Boole estava interessado em estabelecer uma lógica formal para as leis do pensamento. Em outras palavras, estava usando a lógica como linguagem para descrever (i.e. modelar) o processo de raciocínio humano. Porém, ele não estava interessado em mostrar como tal lógica poderia ser efetivamente implementada no cérebro humano, até porque naquela época ainda não se tinha conhecimento de que a unidade individual do sistema nervoso é o neurônio.

Tal conhecimento só ficou claro nos trabalhos do médico e histologista espanhol, Santiago Ramón y Cajal, ganhador do prêmio Nobel de Medicina em 1906 juntamente com o também médico e histologista italiano Camillo Golgi.

Somente em 1943, com o trabalho de Warren McCulloch (neurocientista) e Walter Pitts (logicista), é que surgiu uma primeira hipótese de como tais portas lógicas poderiam ser implementadas no cérebro, dando origem ao modelo matemático conhecido hoje como *neurônio artificial de Mc-Culloch-Pitts* [1].

O modelo do neurônio de McCulloch-Pitts é um combinador linear com uma função limitadora (e.g. função sinal) na sua saída, tal como ilustrado na Figura 5. Compare esta figura com a Figura 3.

Bom, agora ainda permanece em aberto a seguinte questão:

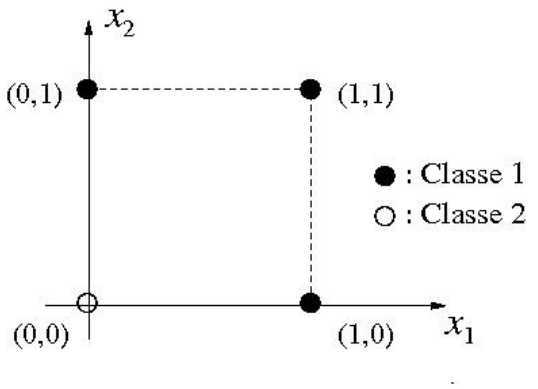
> É possível construir/implementar/projetar uma porta lógica usando o modelo do neurônio de McCulloch-Pitts (M-P)?

A resposta é sim! Na análise que se segue, vamos considerar apenas a parte linear do neurônio M-P. Usaremos a função sinal apenas para limitar/quantizar a saída para os valores ± 1. Considere agora a tabela verdade da função lógica OR para duas variáveis de entrada x_1 e x_2, mostrada na Figura 6a, e sua representação no plano cartesiano $X_1 \times X_2$, mostrada na Figura 6b.

Porta OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

(a)



(b)

Figura 6: Duas representações diferentes da função lógica OR. (a) Tabela-verdade. (b) Plano cartesiano.

Note que se representarmos no plano cartesiano as coordenadas (x_1, x_2) da tabela-verdade da função lógica OR que produzem saída "1" como um pequeno círculo cheio e as coordenadas que produzem "0" com um pequeno círculo vazio, obteremos a Figura 6b. Esta figura nos permite formular o problema de implementar uma porta lógica como um problema de classificação de padrões!

Na classe 1 devem ficar os pontos de coordenadas $\{(1,0);(0,1);(1,1)\}$, enquanto na classe 2 deve ficar apenas o ponto de coordenada (0,0). É possível notar também que a representação no plano cartesiano nos faz perceber que uma simples reta pode ser usada para separar as duas classes. Deste modo, podemos dizer que o problema de classificação que conduz à implementação da função lógica OR é um *problema linearmente separável*.

A reta em questão será implementada pelo neurônio de McCulloch-Pits por meio da seguinte expressão:

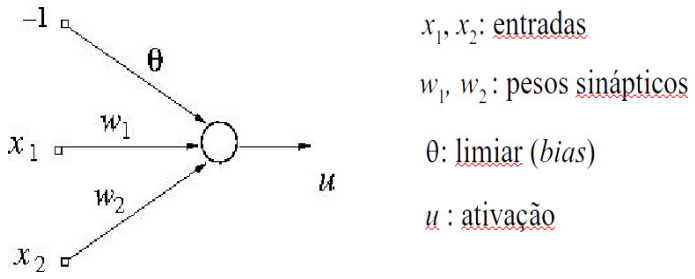
$$\begin{aligned} u &= w_0 x_0 + w_1 x_1 + w_2 x_2, \\ &= -\theta + w_1 x_1 + w_2 x_2, \end{aligned} \tag{12}$$

para a qual assumimos $x_0 = -1$ e $w_0 = \theta$. A representação arquitetural desta reta como um neurônio está representada na Figura 7a e graficamente no plano cartesiano na Figura 7b.

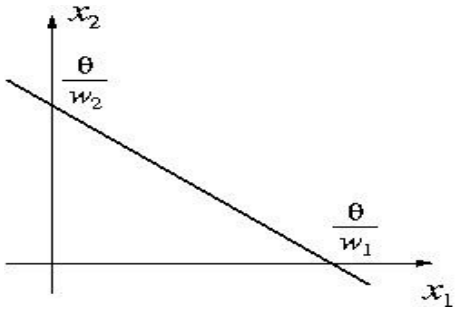
Das aulas de Geometria Analítica lembraremos que uma reta divide o plano cartesiano em duas regiões chamadas semiplanos. Em um dos semiplanos, teremos os pontos que satisfazem $u > 0$, ou seja, são os pontos localizados acima da reta. No outro semiplano, teremos os pontos que satisfazem $u < 0$ (pontos abaixo da reta). Para os pontos na reta teremos $u = 0$.

Nosso objetivo ao desenvolver um modelo matemático que funcione como a função lógica OR consiste em posicionar uma reta que coloque os pontos $\{(1,0);(0,1);(1,1)\}$ no semiplano em que $u > 0$, enquanto o ponto $(0,0)$ fique o semiplano em que $u < 0$. Para este fim, não usaremos a representação binária da função lógica OR mostrada na Figura 6, mas sim a representação *bipolar* mostrada na Figura 8. A única diferença entre as duas representações é o fato de usarmos -1 em vez de 0 (zero) na tabela-verdade correspondente.

Assim, a partir da representação bipolar da função lógica OR, podemos usar a Eq. (12) para,



(a)



(b)

Figura 7: Representação do neurônio MP para implementar a função lógica OR com 2 entradas. (a) Representação arquitetural. (b) Reta no plano cartesiano $X_1 \times X_2$.

linha a linha da tabela-verdade, construir o seguinte sistema de equações lineares:

$$\begin{aligned} -\theta + w_1(-1) + w_2(-1) &= -1 \\ -\theta + w_1(+1) + w_2(-1) &= +1 \\ -\theta + w_1(-1) + w_2(+1) &= +1 \\ -\theta + w_1(+1) + w_2(+1) &= +1 \end{aligned} \tag{13}$$

em que as incógnitas são os parâmetros do modelo do neurônio MP: o limiar (θ) e os pesos sinápticos w_1 e w_2. Vamos reescrever o sistema acima na forma matricial $\mathbf{Aw} = \mathbf{b}$, ou seja

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \\ -1 & +1 & +1 \end{bmatrix} \begin{bmatrix} \theta \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -1 \\ +1 \\ +1 \\ +1 \end{bmatrix}, \tag{14}$$

em que

$$\mathbf{A} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \\ -1 & +1 & +1 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \theta \\ w_1 \\ w_2 \end{bmatrix} \quad \text{e} \quad \mathbf{b} = \begin{bmatrix} -1 \\ +1 \\ +1 \\ +1 \end{bmatrix}. \tag{15}$$

Note que o vetor $\mathbf{w}$ contém os parâmetros ajustáveis do neurônio MP. A matriz de coeficientes $\mathbf{A}$ e o vetor de saídas $\mathbf{b}$ são construídos a partir da tabela-verdade da função lógica OR. Por fim, a primeira coluna da matriz $\mathbf{A}$, formada somente de -1's surge por termos usado a entrada fixa $x_0 = -1$.

Portanto, se conseguirmos resolver o sistema linear acima, ou seja, obter uma solução numérica para $\mathbf{w}$, teremos nosso neurônio MP que implementa a função lógica OR. Surge então a seguinte questão:

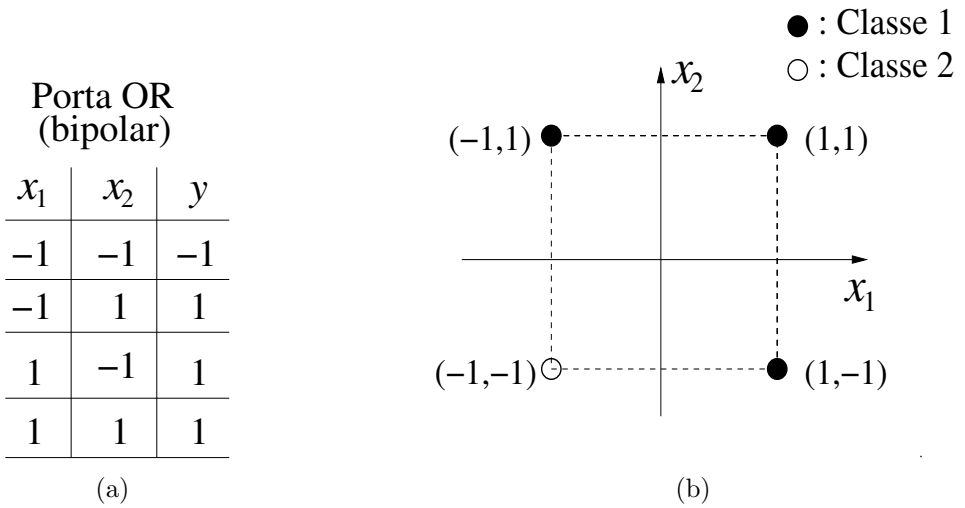


Figura 8: Representação bipolar da função lógica OR. (a) Tabela-verdade. (b) Plano cartesiano.

> Como saber se o sistema em questão admite solução e, em caso afirmativo, saber se ela é única?

Motivados por esta questão, vamos revisar rapidamente a seguir alguns conceitos importantes de Álgebra Linear.

5.1 Dicas de Álgebra Linear

Sempre que um sistema de equações do tipo $\mathbf{A}_{m\times(n+1)}\mathbf{w}_{(n+1)\times 1} = \mathbf{b}_{m\times 1}$ aparecer em um problema de modelagem de um fenômeno, é interessante que analizemos algumas propriedades da matriz $\mathbf{A}$ a fim de verificar se o sistema tem solução. Das aulas de Álgebra Linear sabemos que a existência de solução do sistema está associada ao valor do *posto* da matriz $\mathbf{A}$.

Para quem não lembra, o posto fornece uma medida do grau de *independência linear* das linhas (ou colunas) da matriz. Grosso modo, o posto nos dá o número de linhas (ou colunas) linearmente independentes da matriz, ou seja, quantas linhas (ou colunas) existem que não podem ser escritas como uma combinação linear das outras linhas (ou colunas). Em qualquer matriz o número de linhas linearmente independentes coincide com o número de colunas linearmente independentes.

Formalmente, o posto da matriz $\mathbf{A}_{m\times(n+1)}$ é definido como

$$\rho(\mathbf{A}) = posto(\mathbf{A}) \leq \min(m, n+1), \tag{16}$$

ou seja, o maior valor que o posto de uma matriz pode assumir é igual à menor das dimensões desta matriz. Por exemplo, se uma matriz tem dimensão 4×7, o valor máximo que pode alcançar o posto desta matriz é 4, visto que este é o mínimo dentre 4 e 7.

Uma forma prática de se calcular o posto de uma matriz é colocá-la na forma escalonada reduzida por meio de operações elementares sobre as linhas desta matriz. O posto da matriz será igual ao número de linhas não-nulas da forma escalonada reduzida.

Sabendo como calcular o posto de uma matriz, o nosso interesse maior está em como determinar se um sistema linear do tipo $\mathbf{Aw} = \mathbf{b}$ tem ou não solução. E se existir solução, se ela é única ou se possui infinitas soluções. Existe um resultado clássico em Álgebra Linear, na forma de um teorema, que nos fornece esta resposta. Não iremos demonstrar o teorema, pois nos interessa aqui apenas seu enunciado.

Teorema: Considere o sistema de equações lineares $\mathbf{A}_{m\times(n+1)}\mathbf{w}_{(n+1)\times 1} = \mathbf{b}_{m\times 1}$.

Existência de solução - Tal sistema admite solução se, e somente se, o posto da matriz ampliada $\tilde{\mathbf{A}} = [\mathbf{A} \mid \mathbf{b}]$ é igual ao posto da matriz dos coeficientes $\mathbf{A}$.

Unicidade da solução - Se as duas matrizes têm o mesmo posto p e $p = n + 1$, a solução será única.

Infinidade de soluções - Se as duas matrizes têm o mesmo posto p e $p < n + 1$, podemos escolher $n - p + 1$ incógnitas e as outras p incógnitas serão dadas em função destas. Neste caso, dizemos que o grau de liberdade do sistema é $n - p$, o que significa que o sistema possui $n - p + 1$ variáveis livres.

Inexistência de solução - Por exclusão, se os postos das duas matrizes diferem o sistema não possui solução exata.

Usaremos este teorema para avaliar a existência de solução para o sistema mostrado na Equação (14).

6 Solução pelo Método dos Mínimos Quadrados

Para saber se o sistema tem solução ou não, vamos determinar o posto das matrizes $\mathbf{A}$ e $\tilde{\mathbf{A}} = [\mathbf{A} \mid \mathbf{b}]$. para o problema da implementação da função lógica OR, estas são dadas respectivamente por

$$\mathbf{A} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \\ -1 & +1 & +1 \end{bmatrix} \quad \text{e} \quad \tilde{\mathbf{A}} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 \\ -1 & +1 & +1 & +1 \end{bmatrix}. \tag{17}$$

Usando o Matlab ou o Octave, podemos facilmente determinar o posto dessas duas matrizes usando o comando `rank`. Os resultados obtidos são os seguintes:

```
>> A=[-1 -1 -1;-1 1 -1;-1 -1 1;-1 1 1]; % matriz A
>> b=[-1;1;1;1];
>> AA=[A b]; % matriz ampliada
>> pa=rank(A) % posto da matriz A
pa =
       3
>> paa=rank(AA) % posto da matriz ampliada
paa =
       4
```

Como `pa` $\neq$ `paa`, podemos concluir, sem resolver o sistema, que ele não tem solução exata. Isto significa dizer que não há um vetor $\mathbf{w}$ que ao ser multiplicado pela matriz $\mathbf{A}$ produza *exatamente* o vetor $\mathbf{b}$. Em outras palavras, quando não o sistema não tem solução, sempre teremos $\mathbf{Aw} \neq \mathbf{b}$, $\forall \mathbf{w}$.

Matematicamente, isto é equivalente à seguinte expressão:

$$\mathbf{b} - \mathbf{Aw} \neq \mathbf{0}, \tag{18}$$

que pode ser transformada em uma igualdade e, então, ser escrita como

$$\mathbf{b} - \mathbf{Aw} = \mathbf{e}, \tag{19}$$

com o vetor $\mathbf{e} \in \mathbb{R}^m$ tendo pelo menos uma componente não-nula.

O vetor $\mathbf{e}$ pode ser entendido como o vetor de erros, ou seja, o vetor cujas componentes são os valores que faltaram para o sistema ter solução. Se todas as componentes do vetor $\mathbf{e}$ fossem nulas, não haveria erros e o sistema teria uma solução exata!

Não temos uma solução exata, mas podemos encontrar uma solução aproximada que produza o menor erro possível. Neste caso, demos expressar esta intenção através de uma função-custo ou função-objetivo que dependa do erro. Esta função é a norma quadrática do erro, dada por

$$J(\mathbf{w}) = \|\mathbf{e}\|^2 = \mathbf{e}^T\mathbf{e} = \sum_{i=1}^{m} e_i^2 = (\mathbf{b} - \mathbf{A}\mathbf{w})^T(\mathbf{b} - \mathbf{A}\mathbf{w}), \tag{20}$$

em que e_i denota a i-ésima componente do vetor $\mathbf{e}$ e $\|\cdot\|$ simboliza a norma euclidiana do argumento.

A função-custo $J(\mathbf{w})$ pode ser entendida como uma função que busca encontrar o vetor de parâmetros $\hat{\mathbf{w}}$ do neurônio MP que produz o vetor $\mathbf{e}$ de menor norma quadrática.

A Eq. (20) pode ser decomposta em

$$\begin{aligned} J(\mathbf{w}) &= \mathbf{b}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{w} - \mathbf{w}^T\mathbf{A}^T\mathbf{b} + \mathbf{w}^T\mathbf{A}^T\mathbf{A}\mathbf{w} \\ &= \mathbf{b}^T\mathbf{b} - 2\mathbf{w}^T\mathbf{A}^T\mathbf{b} + \mathbf{w}^T\mathbf{A}^T\mathbf{A}\mathbf{w} \end{aligned} \tag{21}$$

uma vez que $\mathbf{w}^T\mathbf{A}^T\mathbf{b} = \mathbf{b}^T\mathbf{A}\mathbf{w}$ resulta no mesmo escalar.

Daí, para encontrar a estimativa de mínimos quadrados para o vetor de pesos $\mathbf{w}$ temos que determinar o vetor gradiente de $J(\mathbf{w})$ e igualá-lo ao vetor nulo de dimensão adequada, ou seja,

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{A}^T\mathbf{b} + 2\mathbf{A}^T\mathbf{A}\mathbf{w} = \mathbf{0}, \tag{22}$$

em que $\mathbf{0}$ é um vetor de zeros de dimensão $(n+1) \times 1$. Rearranjando os termos da Eq. (22) resulta em

$$\mathbf{A}^T\mathbf{A}\mathbf{w} = \mathbf{A}^T\mathbf{b}. \tag{23}$$

Note que a matriz $\mathbf{A}^T\mathbf{A}$ é quadrada, de dimensão $(n+1) \times (n+1)$. Portanto, para isolar o vetor $\mathbf{w}$ na Eq. (23) basta multiplicar ambos os lados da Eq. (23) pela inversa de $\mathbf{A}^T\mathbf{A}$. Desta forma, a estimativa de mínimos quadrados de $\mathbf{w}$ é dada por

$$\mathbf{w} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}. \tag{24}$$

Portanto, a ativação do neurônio de McCulloch-Pitts, para um dado vetor de entrada $\mathbf{x}$, é calculada por meio da seguinte expressão:

$$\mathbf{u} = \mathbf{A}\mathbf{w}, \tag{25}$$

de modo que a saída quantizada pela função `sign`[4], é dada por

$$\mathbf{y} = \text{sign}(\mathbf{u}), \tag{26}$$

em que a função sinal é aplicada individualmente a cada componente do vetor $\mathbf{u}$.

De posse da matriz $\mathbf{A}$ e do vetor $\mathbf{b}$, podemos estimar os parâmetros do neurônio MP facilmente usando os seguintes comandos no Matlab/Octave.

[4]Em português, chamada de função sinal, sendo definida como $y = 1$, se $u > 0$; $y = -1$, se $u \leq 0$.

```
>> w=inv(A'*A)*A'*b
w =
       -0.50000
        0.50000
        0.50000
```

Note que se multiplicarmos $\mathbf{A}$ e $\mathbf{w}$ não resultará em $\mathbf{b}$:

$$\mathbf{Aw} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \\ -1 & +1 & +1 \end{bmatrix} \begin{bmatrix} -0.5 \\ +0.5 \\ +0.5 \end{bmatrix} = \begin{bmatrix} -0.5 \\ +0.5 \\ +0.5 \\ +1.5 \end{bmatrix} \neq \begin{bmatrix} -1 \\ +1 \\ +1 \\ +1 \end{bmatrix} \tag{27}$$

Note, contudo, que o produto $\mathbf{Aw}$ refere-se à parte linear do neurônio MP. Se aplicarmos a função limitadora (i.e. função sinal) às componentes do vetor $\mathbf{u}$ obteremos o vetor de saídas correto, ou seja

$$\text{sign}(\mathbf{Aw}) = \text{sign}\left(\begin{bmatrix} -0.5 \\ +0.5 \\ +0.5 \\ +1.5 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ +1 \\ +1 \\ +1 \end{bmatrix}. \tag{28}$$

6.1 Comandos Alternativos no Matlab/Octave

A solução de mínimos quadrados no exemplo da implementação da função lógica OR pelo neurônio MP foi obtida através da implementação direta da Eq. (24), que chamaremos aqui de equação de *livro-texto* do método dos mínimos quadrados. Em função do seu elevado custo computacional, principlamente para vetores/matrizes de elevada dimensão, e de uma maior susceptibilidade a erros numéricos, tal implementação não é recomendada.

Para fins mais profissionais, sugere-se que seja dada preferência a um dos seguintes comandos: barra invertida ('\') ou `pinv`. Vamos testá-los abaixo.

```
>> w=A\b
w =
       -0.50000
        0.50000
        0.50000
>> w=pinv(A)*b
w =
       -0.50000
        0.50000
        0.50000
```

Podemos notar que os resultados são os mesmos para os três comandos (incluindo o do livro-texto). Contudo, é importante frisar que há diferenças na forma em que são implementados no Matlab/Octave. O comando da barra invertida ('\') utiliza a decomposição QR, enquando o comando `pinv` usa a técnica de decomposição em valores singulares (SVD, sigla em inglês). Mais detalhes sobre estas técnicas podem ser encontradas em qualquer bom livro de Álgebra Linear. Recomendo o livro do Davyd Lay [2].

Fica a minha recomendação para uso da função `pinv` caso o Matlab/Octave esteja sendo usado ou, de modo mais geral, a técnica baseada em SVD para obtenção da solução de mínimos

quadrados para um sistema linear. Esta técnica pode ser usada mesmo que a matriz $\mathbf{A}$ não tenha posto completo, o que não acontece para o caso do comando baseado na barra invertida (que usa decomposição QR).

Um comentário final: o nome do comando `pinv` se baseia no termo *pseudoinversa*, pois este é o termo pelo qual a matriz $\mathbf{A}^{\dagger} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T$ é conhecida em Álgebra Linear.

Exercício Computacional: Com base na teoria exposta nesta nota de aula, implemente a função lógica AND usando o neurônio de McCulloch-Pitts e o método dos mínimos quadrados.

Referências

[1] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

[2] D. C. Lay. *Álgebra Linear e suas Aplicações*. Editora LTC, 2013.

Tutorial: Regressão Linear Simples no Matlab/Octave

Prof. Dr. Guilherme de Alencar Barreto

Março /2014

Departmento de Engenharia de Teleinformática (DETI)
Programa de Pós-Graduação em Engenharia de Teleinformática (PPGETI)
Universidade Federal do Ceará (UFC), Campus do Pici, Fortaleza-CE

gbarreto@ufc.br

Objetivo - Apresentar uma sequência de comandos do Matlab/Octave para gerar dados correspondentes à simulação de um sistema linear com ruído, a respectiva estimação dos parâmetros do sistema a partir dos dados gerados e a validação do modelo de regressão encontrado através da análise dos resíduos.

1 Comandos no Matlab/Octave

- **Passo 1**: Especificar os parâmetros β_0 e β_1 do sistema linear a ser simulado, bem como a variância do ruído σ_ε^2.

```
>> B0=2; B1=0.8;
>> Vnoise=0.25;
```

Comentário 1: Caso vocês desejem visualizar os valores definidos em `B0`, `B1` e `Vnoise` basta retirar o ponto-e-vírgula que aparece ao final de cada comando. Caso você já tenha digitado com o ponto-e-vírgula, basta chamar o vetor correspondente digitando seu símbolo na linha de comando, individualmente (e.g. `>> B0`) ou separados por vírgulas (`>> B0,B1`). Experimente!

- **Passo 2**: Definir a faixa de valores para x. Por exemplo, de 0 a 5, em incrementos de 0,01.

```
>> x=0:0.01:5;
>> n=length(x);
```

- **Passo 3**: Simular o sistema linear $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$.

```
>> for i=1:n, ...
y(i)=B0+B1*x(i)+normrnd(0,sqrt(Vnoise)); ...
end
```

Comentário 2: O comando `normrnd` gera um número aleatório normalmente distribuído de média 0 e desvio-padrão igual $\sqrt{Vnoise}$. Note que especificamos anteriormente a variância do ruído. Para obter o desvio-padrão, basta extrair a raiz quadrada da variância. Para mais detalhes do comando `normrnd`, digite `>> help normrnd`.

- **Passo 4**: Usar os dados gerados para implementar as seguintes equações de estimação de β_0 e β_1:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \quad (1)$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2} \quad (2)$$

```
>> mx=mean(x); my=mean(y); Sxy=dot(x,y); Sx=sum(x); Sy=sum(y);
Sx2=sum(x.^2); S2x=Sx^2;
>> num=Sxy - Sx*Sy/n;
>> den=Sx2 - S2x/n;
>> B1h=num/den
>> B0h=my-B1h*mx;
>> B0h, B1h
B0h=
  2.0201
B1h=
  0.7875
```

Comentário 3: Note que na sua simulação os valores de $\hat{\beta}_0$ e $\hat{\beta}_1$ não serão os mesmos que os mostrados acima, pois os valores do ruído gerados pelo comando `normrnd` serão diferentes cada vez que você rodar a simulação.

- **Passo 5**: Calcular os valores de $\hat{y}_i$ usando a seguinte reta de regressão:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \quad (3)$$

```
>> for i=1:n, ...
yh(i)=B0h+B1h*x(i); ...
end
```

- **Passo 6**: Calcular os resíduos (i.e. erros de predição) resultantes, $e_i = y_i - \hat{y}_i$, bem como estimar a variância do ruído ($\hat{\sigma}_\varepsilon^2$).

```
>> residuos = y - yh;
>> Vnoiseh= sum(residuos.^2)/(n-2)
Vnoiseh=
       0.2516
```

- **Passo 7**: Avaliar *qualitativamente* a forma da distribuição dos resíduos usando histograma.

```
>> histfit(residuos)
```

Comentário 4: Quanto mais semelhante à distribuição Normal melhor.

- **Passo 8**: Avaliar *quantitativamente* a gaussianidade da distribuição dos resíduos através de um teste de hipótese. Neste exemplo, vamos usar o teste de Kolmogorov-Smirnov. Para isso, temos antes que normalizar a variância dos resíduos para 1. Se o resultado for $H = 0$, a hipótese nula de que a distribuição dos resíduos é $N(0, 1)$ deve ser aceita. Se $H = 1$, tal hipótese deve ser rejeitada (ou como preferem dizer os Estatísticos, não há informação suficiente para aceitar a hipótese nula.)

```
>> residuos_norm=residuos/std(residuos);
>> H=kstest(residuos_norm)
H=
       0
```

- **Passo 9**: Avaliar a distribuição acumulada (FDA) dos resíduos normalizados versus a FDA da distribuição normal padronizada (i. e. $N(0, 1)$) através do comando `cdfplot`.

```
>> figure; cdfplot(residuos_norm); hold on
>> z=randn(n,1); cdfplot(z); hold off
```

Comentário 5: O resultado está mostrado na figura abaixo. Esta análise comparativa das FDAs dos resíduos normalizados e da distribuição Normal padronizada $N(0, 1)$ é basicamente o que o teste de Kolmogorov-Smirnov faz de modo quantitativo.

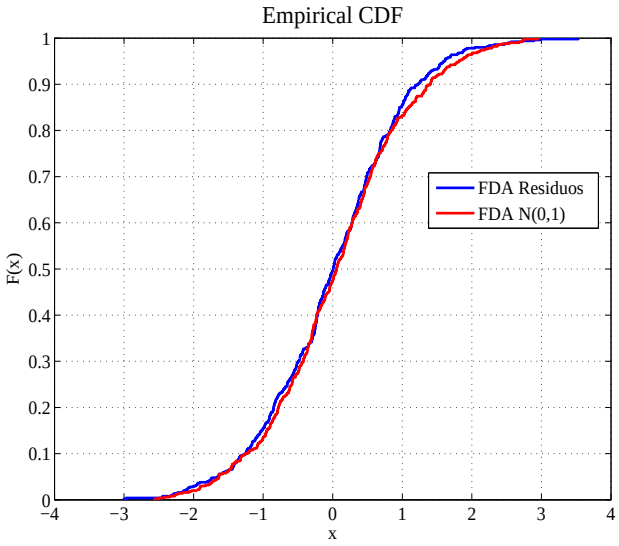


Figura 1: FDAs empíricas: Resíduos normalizados versus amostra de uma distribuição $N(0, 1)$.

Tutorial: Regressao Polinomial no Matlab/Octave

Prof. Dr. Guilherme de Alencar Barreto

Maio /2014

Departmento de Engenharia de Teleinformática (DETI)
Curso de Graduação em Engenharia de Teleinformática (CGETI)
Universidade Federal do Ceará (UFC), Campus do Pici, Fortaleza-CE

gbarreto@ufc.br

OBJETIVO - Usando as medidas de velocidade do vento (m/s) e de potência gerada (kWatts) para um gerador eólico disponibilizados no arquivo `aerogerador.dat`, determinar o melhor modelo de regressão polinomial para estes dados.

1 Implementação via Linguagem Script do Matlab

Passo 1 - Visualização dos dados.

```
>> load aerogerador.dat % carrega arquivo de dados
>> v=aerogerador(:,1); % medidas de velocidades
>> P=aerogerador(:,2); % medidas de potencia
>> figure; plot(v,P,'bo'); grid; % diagrama de dispersao
>> xlabel('Velocidade do vento [m/s]'); ylabel('Potencia gerada [kWatts]');
```

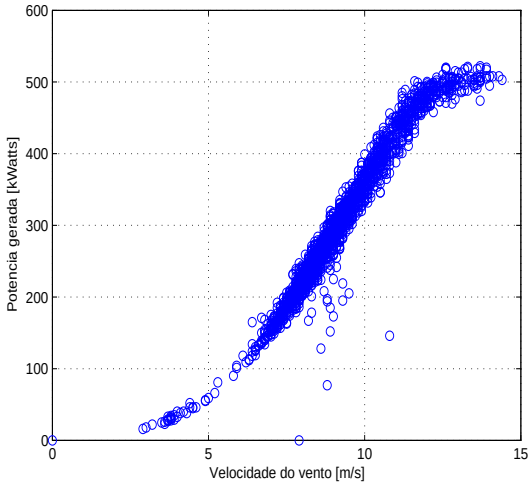


Figura 1: Diagrama de dispersão dos dados do aerogerador.

Observações 1

1. Uma análise da figura revela que a relação entre as duas variáveis é não-linear. Assim, não vale a pena avaliar um modelo de regressão linear simples.
2. Também não vale a pena avaliar um modelo polinomial de ordem 2 (ou seja, quadrático) porque percebemos duas concavidades ou curvaturas no gráfico, uma no início e outra no fim. Modelos quadráticos (i.e. parábolas) só possuem uma curvatura.
3. Portanto, vamos começar nossa análise com um modelo polinomial de ordem 3.

Passo 2 - Construção do modelo linear $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$, em que

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n\times 1}, \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{bmatrix}_{n\times(k+1)} \quad \text{e} \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}_{(k+1)\times 1}$$

Nas equações acima, o par (x_i, y_i), $i = 1, 2, ...n$, corresponde à i-ésima observação conjunta da velocidade do vento (x_i, variável de entrada) e da potência gerada (y_i, variável de saída) correspondente. Além disso, β_j, $j = 1, ..., k$, são os parâmetros a estimar, n é o total de observações e k é a ordem do polinômio.

```
>> k=3; % ordem inicial do polinomio
>> n=length(v); % numero de medidas
>> y=P; % vetor de observacoes da variavel de saida
>> X=[]; for l=1:k+1, X=[X v.^(l-1)]; end; % Constroi recursivamente a matriz X
```

Passo 3 - Estimação de $\boldsymbol{\beta}$ pelo método dos mínimos quadrados (MQ).

```
>> B=inv(X'*X)*X'*y % Formula de livro-texto para MQ
B =
32.6235
-58.7604
15.0519
-0.5924

>> B=X\y % Metodo alternativo via operador 'barra invertida'
B =
32.6235
-58.7604
15.0519
-0.5924
```

Passo 4 - Avaliação quantitativa do modelo por meio do coeficiente de determinação R^2:

$$R^2 = 1 - \frac{SQ_E}{S_{yy}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{1}$$

```
>> ypred = X*B; % Predicao da saida para dados observados
>> erro=y-ypred; % Calcula erros
```

```
>> SEQ=sum(erro.^2); % Calcula a soma dos erros quadraticos
>> ymed=mean(y); % Calcula potencia media
>> Syy=sum((y-ymed).^2); % Soma dos erros para modelo baseado na media
>> R2 = 1 - SEQ/Syy
R2 =
0.9690
```

Passo 5 - Avaliação qualitativa do modelo (visual).

```
>> vv=min(v):0.1:max(v); vv=vv'; % Define faixa de valores para velocidade
>> XX=[]; for l=1:k+1, XX=[XX vv.^(l-1)]; end; % Monta nova matriz de regressao
>> ypred2=XX*B; % predicao da saida correspondente
>> hold on; plot(vv,ypred2,'r-'); hold off; % Sobrepoe curva de regressao aos dados
```

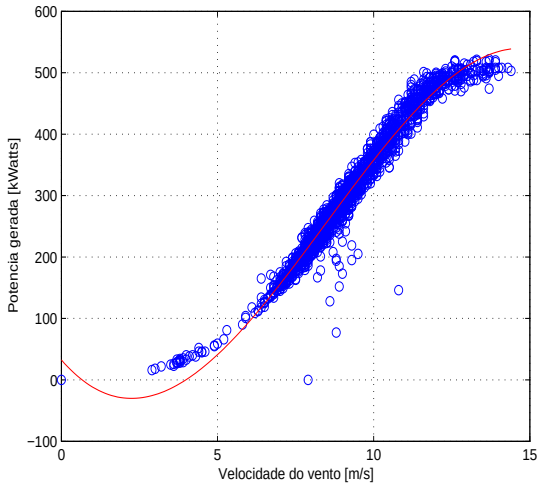


Figura 2: Curva de regressão sobreposta ao diagrama de dispersão dos dados do aerogerador.

Observações 2

1. Uma análise da curva de regressão no gráfico acima revela que o modelo de regressão polinomial de ordem 3 não é adequado, embora possua um coeficiente de determinação R^2 alto (i.e. próximo de 1).

2. O modelo de regressão polinomial de ordem 3 não é adequado porque para algumas velocidades, principalmente na faixa entre 0 e 5 m/s, o modelo prevê potências negativas, o que não é possível para um aerogerador. O modelo está dizendo que o aerogerador consome energia, em vez de gerá-la.

3. Devemos então repetir o procedimento de construção e avaliação do modelo polinomial para outros valores de k, por exemplo, $k = 4, 5, ...$, assim por diante até um valor máximo k_{max} pré-especificado pelo projetista.

4. Para cada valor de k devemos determinar o valor respectivo de $R^2(k)$ e o gráfico correspondente da curva de regressão resultante.

5. Deve-se escolher o menor valor de k que satisfaça as restrições de Engenharia do problema. No presente caso, não gerar energia com valores negativos.

2 Implementação via Funções Polyfit e Polyval

A seqüência de comandos mostrada anteriormente pode ser executada tanto em ambiente Matlab quanto no Octave. O valor da solução apresentada é principalmente didático, pois cada etapa foi discutida e realizada a partir das equações presentes na maioria dos livros-texto. Contudo, o procedimento de se obter e avaliar um modelo de regressão polinomial pode ser consideravelmente simplificado se optarmos por utilizar as funções `polyfit` e `polyval` presentes em ambas as plataformas de programação científica (Matlab e Octave). A nova seqüência de comandos é apresentada a seguir em uma só etapa. Os gráficos são omitidos por serem iguais aos mostrados anteriormente.

```
>> load aerogerador.dat % carrega arquivo de dados
>> v=aerogerador(:,1); % medidas de velocidades
>> P=aerogerador(:,2); % medidas de potencia
>> figure; plot(v,P,'bo'); grid; % diagrama de dispersao
>> xlabel('Velocidade do vento [m/s]'); ylabel('Potencia gerada [kWatts]');
>> k=3; % ordem inicial do polinomio
>> B = polyfit(v,P,k) % Estimacao do vetor de parametros
B =
   -0.5924 15.0519 -58.7604 32.6235
>> ypred=polyval(B,v); % Predicao da saida para dados observados
>> erro=y-ypred; % Calcula erros
>> SEQ=sum(erro.^2); % Calcula a soma dos erros quadraticos
>> ymed=mean(y); % Calcula potencia media
>> Syy=sum((y-ymed).^2); % Soma dos erros para modelo baseado na media
>> R2 = 1 - SEQ/Syy
R2 =
0.9690
>> vv=min(v):0.1:max(v); vv=vv'; % Define faixa de valores para velocidade
>> ypred2=polyval(B,vv); % predicao da saida correspondente
>> hold on; plot(vv,ypred2,'r-'); hold off; % Sobrepoe curva de regressao aos dados
```

Observações 3

1. Note que a ordem dos coeficientes (parâmetros) quando se usa a função `polyfit` está invertida em relação à obtida através da implementação baseada na equação do estimador dos mínimos quadrados.

3 Trabalho Computacional sobre Regressão Polinomial

Questão Única - Gerar um gráfico $R^2(k) \times k$ a fim de decidir pelo valor ótimo de k, ou seja, a fim de escolher a melhor ordem para o polinômio regressor. Em outras palavras, para cada valor de $k = 3, 4, ..., k_{max}$, determinar o valor correspondente de $R^2(k)$ e gerar o gráfico correspondente usando a função `plot` do Matlab/Octave. Em seguida, de posse do valor de k escolhido, gerar o gráfico de dispersão dos dados com a curva de regressão sobreposta a ele. Sugestão: Fazer $k_{max} = 15$.