

# Regressão Linear

Prof. Danilo Silva

EEL7514/EEL7513 - Tópico Avançado em Processamento de Sinais

EEL410250 - Aprendizado de Máquina

EEL / CTC / UFSC

# Tópicos

- ▶ Regressão: Conceitos gerais
- ▶ Regressão linear
- ▶ Função custo
- ▶ Otimização analítica (equação normal)
- ▶ Regressão linear com funções de base
- ▶ Overfitting e regularização
- ▶ Escolha de hiperparâmetros

# **Regressão: Conceitos Gerais**

# Regressão

	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

► Variáveis aleatórias:

- $\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^n$  é o **vetor de entrada** ou **vetor de atributos**
- $y \in \mathbb{R}$  é a **variável de saída** ou **valor-alvo** ou **rótulo** correspondente a  $\mathbf{x}$

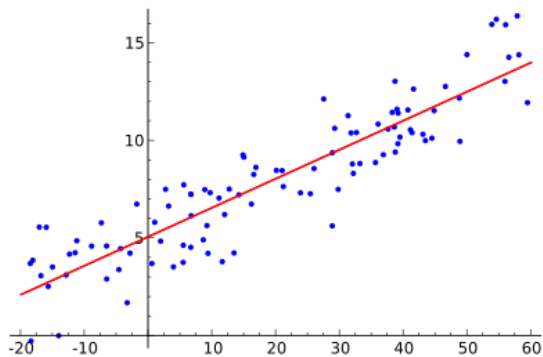
► A distribuição  $p(\mathbf{x}, y)$  é desconhecida, mas conhecemos  $m$  amostras  $(\mathbf{x}^{(i)}, y^{(i)})$  dessa distribuição chamadas de **exemplos de treinamento**

► **Conjunto de treinamento:**  $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$

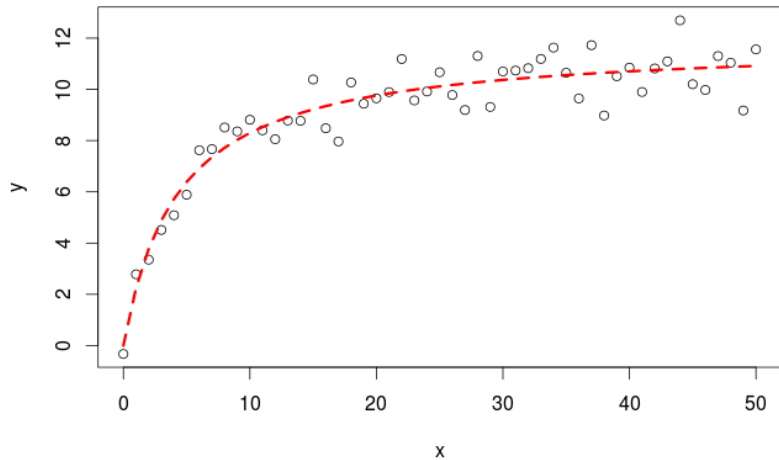
# Regressão

- ▶ Um **modelo de aprendizado** determina uma função **hipótese**  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  que realiza a predição  $\hat{y} = f(\mathbf{x})$ 
  - ▶ A função  $f$  é escolhida dentre um **espaço de hipóteses**  $\mathcal{H}$
- ▶ É intuitivo exigir  $f(\mathbf{x}) \approx y$  para  $(\mathbf{x}, y) \in \mathcal{D}$ , mas o que realmente importa é qualidade da predição para **novos exemplos**  $(\mathbf{x}, y) \notin \mathcal{D}$  fora do conjunto de treinamento

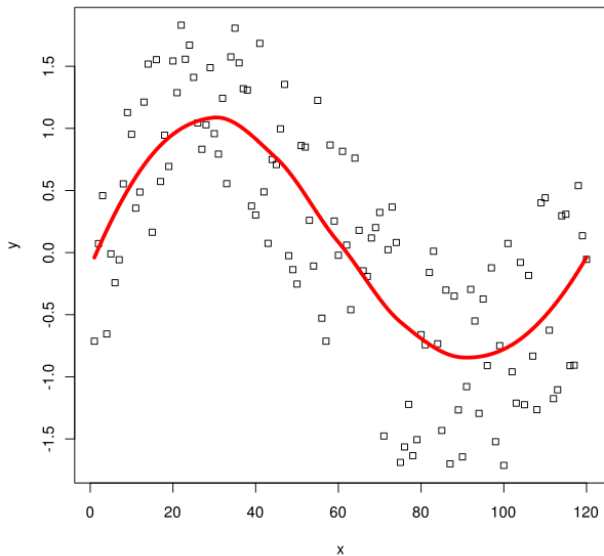
# Exemplos



# Exemplos

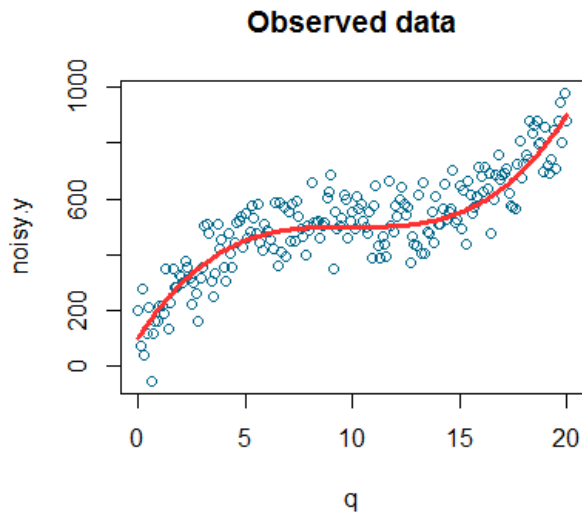


# Exemplos

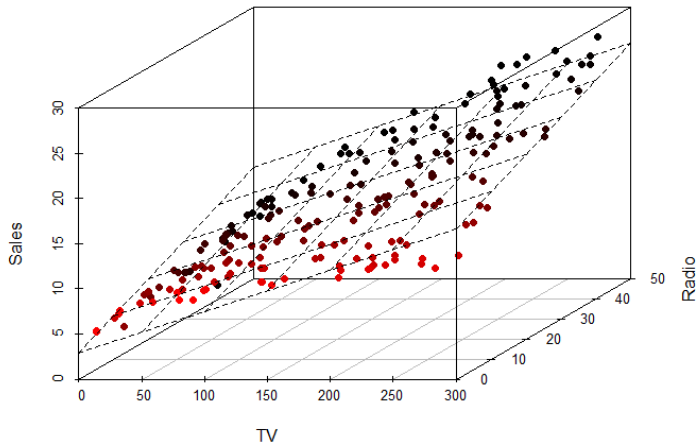




## Exemplos

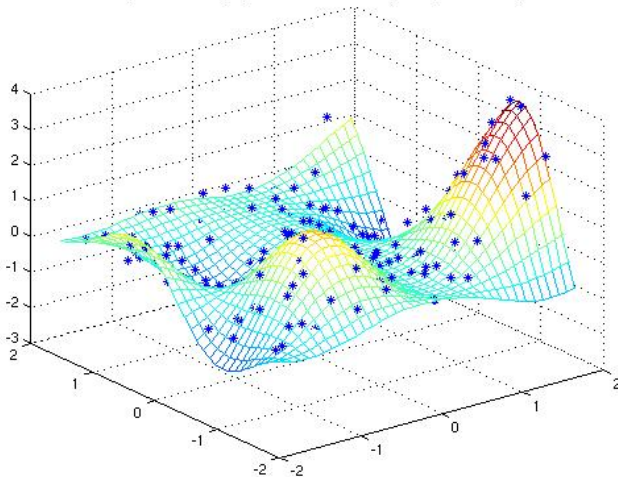


# Exemplos

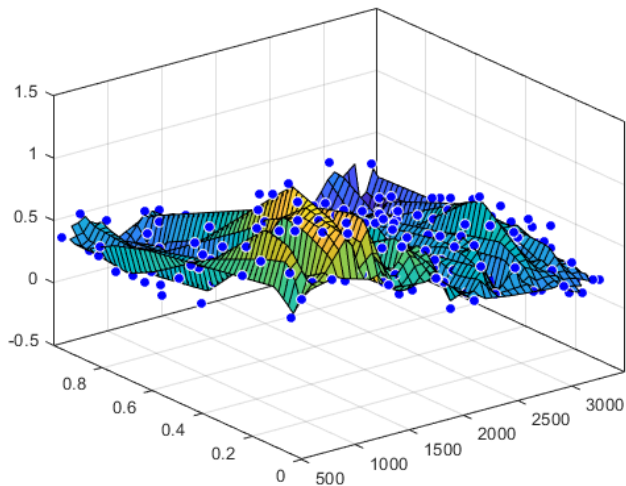


# Exemplos

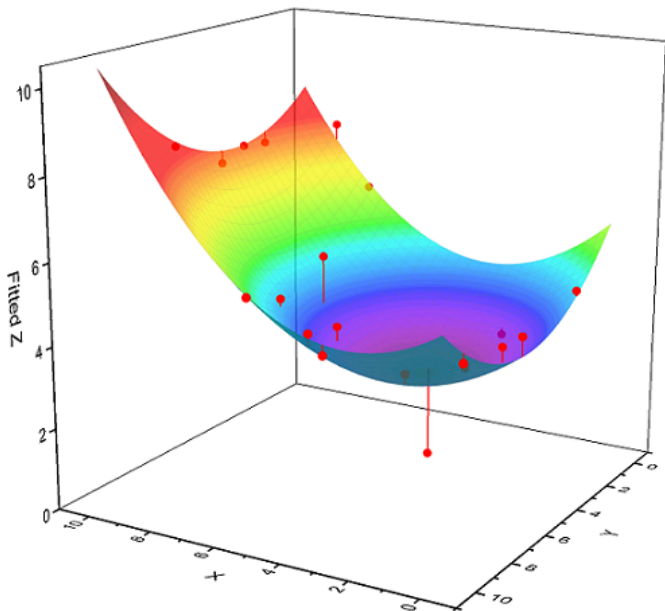
The predicted underlying function and the data points (MAP solution)



# Exemplos



# Exemplos



# Função Perda

- ▶ Para avaliar um modelo é preciso definir uma métrica de desempenho, isto é, uma **função perda** (*loss function*)  $L(y, \hat{y})$ 
  - ▶ Indica quão “ruim” é a predição  $\hat{y}$  quando o valor-alvo correto é  $y$
- ▶ Exemplos:
  - ▶ Erro absoluto (perda  $\ell_1$ ):

$$L(\hat{y}, y) = |\hat{y} - y|$$

- ▶ Erro quadrático (perda  $\ell_2$ ):

$$L(\hat{y}, y) = (\hat{y} - y)^2$$

- ▶ **Vantagem**: Mais conveniente matematicamente, diferenciável em todos os pontos
    - ▶ **Desvantagens**: Mais sensível a *outliers*, unidade quadrática

# Erro Médio de um Modelo

- ▶ Para medir o desempenho de um modelo no conjunto de treinamento, é usual calcular a perda média (ou erro médio) sobre todo o conjunto:

$$J(f) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}} L(f(\mathbf{x}), y) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}^{(i)}), y^{(i)})$$

- ▶ Exemplos:

- ▶  $L = \text{erro quadrático} \implies J = \text{erro quadrático médio (MSE)}$
- ▶  $L = \text{erro absoluto} \implies J = \text{erro absoluto médio (MAE)}$

# Treinamento

- ▶ O erro no conjunto de treinamento é usado para determinar os parâmetros do modelo (através de otimização numérica), isto é, para selecionar a hipótese  $f \in \mathcal{H}$  (dentro um espaço de hipóteses pré-definido) que melhor se ajusta aos dados de treinamento
  - ▶ Treinamento também é chamado de ajuste (*fit*)
- ▶ A função  $J(f)$  é usada como função objetivo (ou função custo) de um algoritmo de otimização:

$$\min_{f \in \mathcal{H}} J(f)$$



# Avaliação do Modelo Treinado

- ▶ Para avaliar o **poder preditivo** de um modelo (generalização), deve-se medir o desempenho sobre um **conjunto de teste**

$$\mathcal{D}_{\text{test}} = \{(\mathbf{x}_{\text{test}}^{(1)}, y_{\text{test}}^{(1)}), \dots, (\mathbf{x}_{\text{test}}^{(m_{\text{test}})}, y_{\text{test}}^{(m_{\text{test}})})\}$$

gerado a partir da mesma distribuição  $p(\mathbf{x}, y)$  de forma **independente** do conjunto de treinamento:

$$J_{\text{test}}(f) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x}, y) \in \mathcal{D}_{\text{test}}} L(f(\mathbf{x}), y) = \frac{1}{m_{\text{test}}} \sum_{i=1}^{m_{\text{test}}} L(f(\mathbf{x}_{\text{test}}^{(i)}), y_{\text{test}}^{(i)})$$

- ▶ Em alguns casos podemos estar interessados em métricas de teste diferentes da usada no treinamento
  - ▶ Ex: MSE para treinamento / RMSE (raiz do MSE) e MAE para teste

# **Regressão Linear**

# Modelo Linear para Regressão

- ▶ Função-hipótese:

$$\hat{y} = f(\mathbf{x}) = w_0 + w_1x_1 + \cdots + w_nx_n$$

- ▶ Parâmetros do modelo:  $w_0, w_1, \dots, w_n$
- ▶ Se  $n = 1$ , temos uma reta

$$\hat{y} = f(x) = w_0 + w_1x = b + wx$$

onde  $w = w_1$  é o **coeficiente de inclinação** e  $b = w_0$  é a intersecção com o eixo  $y$  (*intercept term*), também chamado de **bias**

# Notação Vetorial

- ▶ Em notação vetorial, temos

$$\hat{y} = f(\mathbf{x}) = w_0 + [w_1 \quad \cdots \quad w_n] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = b + \mathbf{w}^T \mathbf{x}$$

onde  $b = w_0$  e  $\mathbf{w} = [w_1 \quad \cdots \quad w_n]^T$

- ▶ Matematicamente, é mais conveniente considerar  $b = 0$  e incluir o atributo constante  $x_0 = 1$  como parte do vetor  $\mathbf{x}$ :

$$\hat{y} = f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$$

onde  $\mathbf{w} = [w_0 \quad w_1 \quad \cdots \quad w_n]^T$  e  $\mathbf{x} = [1 \quad x_1 \quad \cdots \quad x_n]^T$

- ▶ Usaremos essa notação daqui para frente, exceto quando mencionado o contrário

# Função Custo

- ▶ Como  $f(\cdot)$  é parametrizada por  $\mathbf{w}$ , denotamos a função custo do treinamento (erro médio no conjunto de treinamento) por

$$J(\mathbf{w}) = J(f) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}^{(i)}), y^{(i)}) = \frac{1}{m} \sum_{i=1}^m L(\mathbf{w}^T \mathbf{x}^{(i)}, y^{(i)})$$

- ▶ Assumindo como função perda o **erro quadrático**  $L(\hat{y}, y) = (\hat{y} - y)^2$ :

$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$$

- ▶ Como encontrar  $\mathbf{w}$  que minimiza  $J(\mathbf{w})$ ?

# Função Custo

- ▶ Como  $f(\cdot)$  é parametrizada por  $\mathbf{w}$ , denotamos a função custo do treinamento (erro médio no conjunto de treinamento) por

$$J(\mathbf{w}) = J(f) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}^{(i)}), y^{(i)}) = \frac{1}{m} \sum_{i=1}^m L(\mathbf{w}^T \mathbf{x}^{(i)}, y^{(i)})$$

- ▶ Assumindo como função perda o **erro quadrático**  $L(\hat{y}, y) = (\hat{y} - y)^2$ :

$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$$

- ▶ Como encontrar  $\mathbf{w}$  que minimiza  $J(\mathbf{w})$ ?
- ▶ **R:** Fazendo  $\frac{\partial J(\mathbf{w})}{\partial w_j} = 0$

# Função Custo

- ▶ Como  $f(\cdot)$  é parametrizada por  $\mathbf{w}$ , denotamos a função custo do treinamento (erro médio no conjunto de treinamento) por

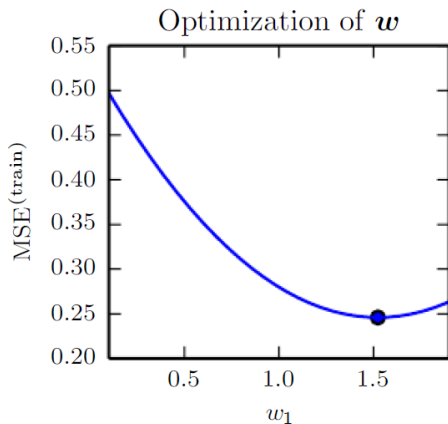
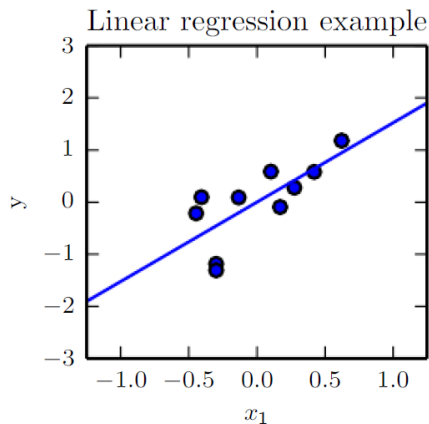
$$J(\mathbf{w}) = J(f) = \frac{1}{m} \sum_{i=1}^m L(f(\mathbf{x}^{(i)}), y^{(i)}) = \frac{1}{m} \sum_{i=1}^m L(\mathbf{w}^T \mathbf{x}^{(i)}, y^{(i)})$$

- ▶ Assumindo como função perda o **erro quadrático**  $L(\hat{y}, y) = (\hat{y} - y)^2$ :

$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2$$

- ▶ Como encontrar  $\mathbf{w}$  que minimiza  $J(\mathbf{w})$ ?
- ▶ **R:** Fazendo  $\frac{\partial J(\mathbf{w})}{\partial w_j} = 0$
- ▶ Também conhecido como **Método dos Mínimos Quadrados** (*Ordinary Least Squares*)

# Exemplo





# Otimização dos Parâmetros do Modelo

- Podemos escrever

$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^m (\mathbf{w}^T \mathbf{x}^{(i)} - y^{(i)})^2 = \frac{1}{m} \sum_{i=1}^m (e^{(i)})^2$$

onde  $e^{(i)} = \mathbf{x}^{(i)T} \mathbf{w} - y^{(i)}$

- Agrupando em um vetor coluna, temos  $\mathbf{e} = \mathbf{X}\mathbf{w} - \mathbf{y}$ , onde

$$\mathbf{X} = \begin{bmatrix} \text{---} (\mathbf{x}^{(1)})^T \text{---} \\ \vdots \\ \text{---} (\mathbf{x}^{(m)})^T \text{---} \end{bmatrix} \quad \mathbf{e} \quad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

são a **matriz de projeto** (*design matrix*) e o vetor de rótulos

- Portanto,

$$J(\mathbf{w}) = \frac{1}{m} \|\mathbf{e}\|^2 = \frac{1}{m} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

# Otimização dos Parâmetros do Modelo

- ▶ Função custo:  $J(\mathbf{w}) = \frac{1}{m} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$
- ▶ Pode-se mostrar que

$$\nabla J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial w_0} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial w_n} \end{bmatrix} = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{y})$$

- ▶ Decorre que o valor ótimo de  $\mathbf{w}$  é dado pela solução do sistema

$$\mathbf{X}^T \mathbf{X} \mathbf{w} = \mathbf{X}^T \mathbf{y}$$

o qual admite uma solução analítica

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- ▶ Conhecida como **equação normal** (*normal equation*)
- ▶ Requer que  $\mathbf{X}^T \mathbf{X}$  seja inversível

# Interpretação Geométrica

- ▶ Predição no conjunto de treinamento:  $\hat{y}^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} = \mathbf{x}^{(i)T} \mathbf{w}$
- ▶ Seja  $\hat{\mathbf{y}} = (\hat{y}^{(1)}, \dots, \hat{y}^{(m)})^T = \mathbf{X}\mathbf{w}$  o vetor de predições
  - ▶ Note que  $\hat{\mathbf{y}} \in \mathcal{S}$ , onde  $\mathcal{S}$  é o subespaço gerado pelas colunas de  $\mathbf{X}$
- ▶ Erro de predição médio sobre o conjunto de treinamento:

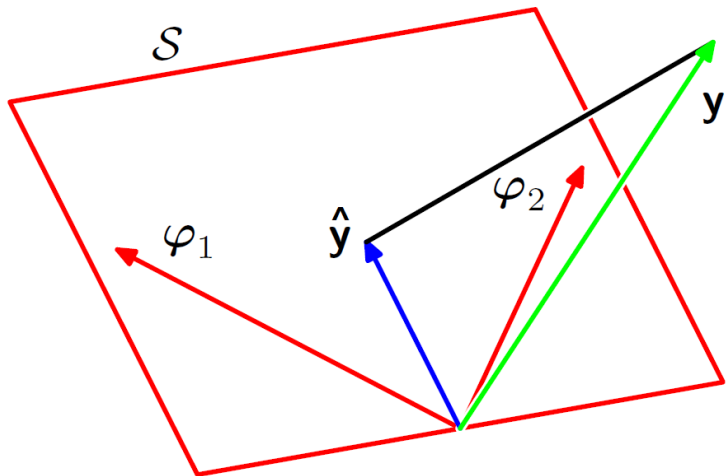
$$J(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{m} \|\hat{\mathbf{y}} - \mathbf{y}\|^2$$

- ▶ Desejamos encontrar o vetor  $\hat{\mathbf{y}} \in \mathcal{S}$  mais próximo de  $\mathbf{y}$
- ▶ Solução é dada pela projeção ortogonal de  $\mathbf{y}$  em  $\mathcal{S}$ :

$$\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$$

onde  $\mathbf{P} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$  é a matriz de projeção

# Interpretação Geométrica



# **Regressão Linear: Extensões**

# Regressão Linear com Funções de Base

- ▶ Suponha  $n = 1$ . Se desejarmos ajustar um modelo mais flexível do que uma reta  $\hat{y} = w_0 + w_1x$ ?
- ▶ Regressão polinomial:

$$\hat{y} = w_0 + w_1x + w_2x^2 + \cdots + w_dx^d$$

- ▶ A determinação de  $\mathbf{w}$  continua a mesma: basta definir  $x_i = x^i$ , isto é,  $\mathbf{x} = (1, x, x^2, x^3, \cdots, x^d)^T$  e  $n = d$ . Em particular:

$$\mathbf{X} = \begin{bmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \cdots & (x^{(1)})^d \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x^{(m)} & (x^{(m)})^2 & \cdots & (x^{(m)})^d \end{bmatrix}$$

# Regressão Linear com Funções de Base

- ▶ Em geral, podemos considerar como atributos  $x_i = \varphi_i(x)$ , i.e.,

$$\hat{y} = w_0 + w_1\varphi_1(x) + w_2\varphi_2(x) + \cdots + w_n\varphi_n(x)$$

onde  $\varphi_i(x)$  são **funções de base** quaisquer

- ▶ Nesse caso,

$$\mathbf{X} = \begin{bmatrix} 1 & \varphi_1(x^{(1)}) & \varphi_2(x^{(1)}) & \cdots & \varphi_d(x^{(1)}) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \varphi_1(x^{(m)}) & \varphi_2(x^{(m)}) & \cdots & \varphi_d(x^{(m)}) \end{bmatrix}$$

e

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

- ▶ Matematicamente, o modelo continua idêntico, apenas partindo de atributos diferentes

# Regressão Linear com Funções de Base

- ▶ A determinação de  $\mathbf{w}$  não muda pois o modelo continua sendo **linear nos parâmetros**:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

- ▶ Do ponto de vista computacional, não importa se  $x_i$  é um atributo “natural” ou produzido “artificialmente” a partir de outros atributos
- ▶ Exemplos:
  - ▶ Base polinomial:  $\{1, x, x^2, x^3, \dots\}$
  - ▶ Base de Fourier:  $\{1, \cos(\frac{2\pi}{T_0} jx), \sin(\frac{2\pi}{T_0} jx), j = 1, 2, 3, \dots\}$
- ▶ Nesse caso, o número de funções de base é um **hiperparâmetro**, isto é, um parâmetro que não pode ser aprendido no treinamento e deve ser definido previamente
  - ▶ Hiperparâmetros alteram o espaço de hipóteses  $\mathcal{H}$



# Regressão Linear com Funções de Base

- ▶ De maneira geral, podemos transformar um vetor  $\mathbf{x} = (x_1, \dots, x_n)^T$  através de funções arbitrárias  $\varphi_i(\mathbf{x})$ :

$$\mathbf{x}' = (x'_1, \dots, x'_n)^T = \boldsymbol{\varphi}(\mathbf{x}) = (\varphi_1(\mathbf{x}), \dots, \varphi_{n'}(\mathbf{x}))^T$$

- ▶ Treinamento e predição não mudam, basta substituir  $\mathbf{X}$  por  $\mathbf{X}'$
- ▶ Exemplo ( $n = 2$ , base polinomial de grau  $d = 3$ ):

$$\begin{array}{llll} \varphi_0(\mathbf{x}) = 1 & \varphi_1(\mathbf{x}) = x_1 & \varphi_3(\mathbf{x}) = x_1^2 & \varphi_6(\mathbf{x}) = x_1^3 \\ & \varphi_2(\mathbf{x}) = x_2 & \varphi_4(\mathbf{x}) = x_1 x_2 & \varphi_7(\mathbf{x}) = x_1^2 x_2 \\ & & \varphi_5(\mathbf{x}) = x_2^2 & \varphi_8(\mathbf{x}) = x_1 x_2^2 \\ & & & \varphi_9(\mathbf{x}) = x_2^3 \end{array}$$

- ▶ Escolher bons atributos (ou transformações de atributos) é um problema de *feature engineering*

# Desafios

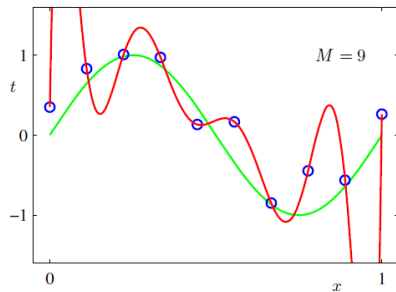
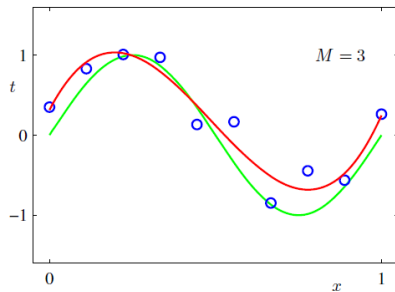
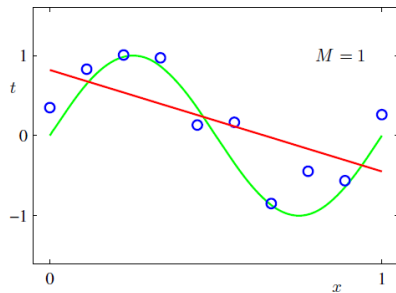
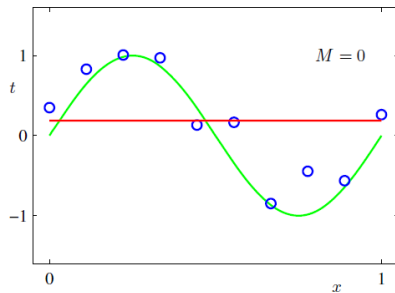
- ▶ O uso da equação normal requer que  $\mathbf{X}^T \mathbf{X}$  seja inversível
- ▶ No entanto,  $\mathbf{X}^T \mathbf{X}$  será singular (não-inversível):
  - ▶ se  $n \geq m$ ; ou
  - ▶ se houver atributos redundantes (linearmente dependentes)
- ▶ Além disso, o aumento de  $n$  pode causar overfitting
- ▶ Uma solução para ambos os problemas é utilizar regularização

# **Overfitting e Regularização**

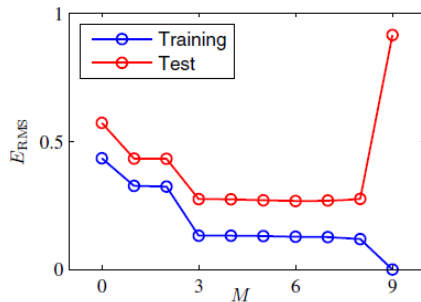
# Overfitting

- ▶ O uso de um grande número de atributos torna o modelo mais suscetível a overfitting
- ▶ Em alguns casos, pode ser interessante reduzir o número de atributos, seja de forma manual ou através de algoritmos
- ▶ Em outros casos, podemos preferir manter todos os atributos—por exemplo, se todos os atributos são ligeiramente úteis individualmente mas bastante úteis em conjunto
  - ▶ Nesse caso, como evitar overfitting?

# Exemplo



# Exemplo



	$M = 0$	$M = 1$	$M = 6$	$M = 9$
$w_0^*$	0.19	0.82	0.31	0.35
$w_1^*$		-1.27	7.99	232.37
$w_2^*$			-25.43	-5321.83
$w_3^*$			17.37	48568.31
$w_4^*$				-231639.30
$w_5^*$				640042.26
$w_6^*$				-1061800.52
$w_7^*$				1042400.18
$w_8^*$				-557682.99
$w_9^*$				125201.43

# Navalha de Occam

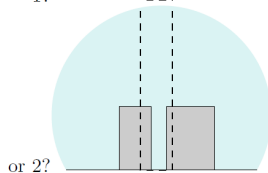
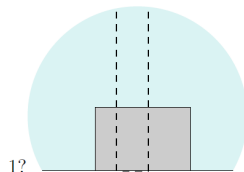
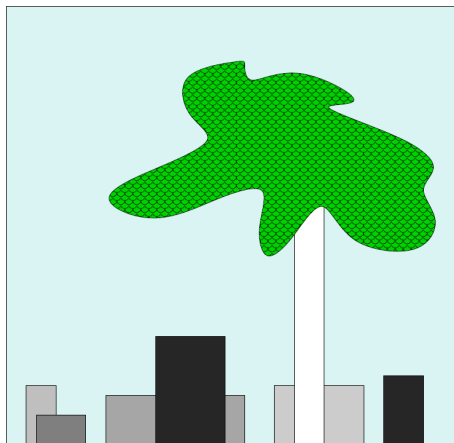


Figure 28.2. How many boxes are behind the tree?

- **Navalha de Occam** (*Occam's razor*): dentre todas as explicações consistentes com os dados, a **mais simples** é a mais plausível
  - Menores valores de  $w_0, w_1, \dots, w_n$  = explicação “mais simples”

# Regularização

- ▶ Consiste em **penalizar** parâmetros que assumem valores muito elevados:

$$J(\mathbf{w}) = J_{\text{train}}(\mathbf{w}) + \lambda \Omega(\mathbf{w})$$

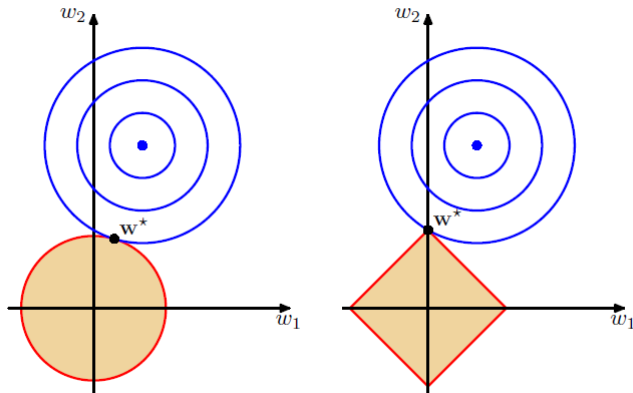
onde

- ▶  $J_{\text{train}}(\mathbf{w})$  é o erro médio sobre o conjunto de treinamento
  - ▶  $\Omega(\mathbf{w})$  é a função de penalidade, chamada de **regularizador**
  - ▶  $\lambda$  é o **parâmetro de regularização** (controla nossa preferência por parâmetros “menores”, i.e., que sejam menos penalizados)
  - ▶  $J(\mathbf{w})$  é a função objetivo da otimização
- ▶ Regularização  $\ell_2$  ou *ridge regression* ou *weight-decay*:

$$\Omega(\mathbf{w}) = \|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{w} = w_0^2 + w_1^2 + \cdots + w_n^2$$



# Regularização

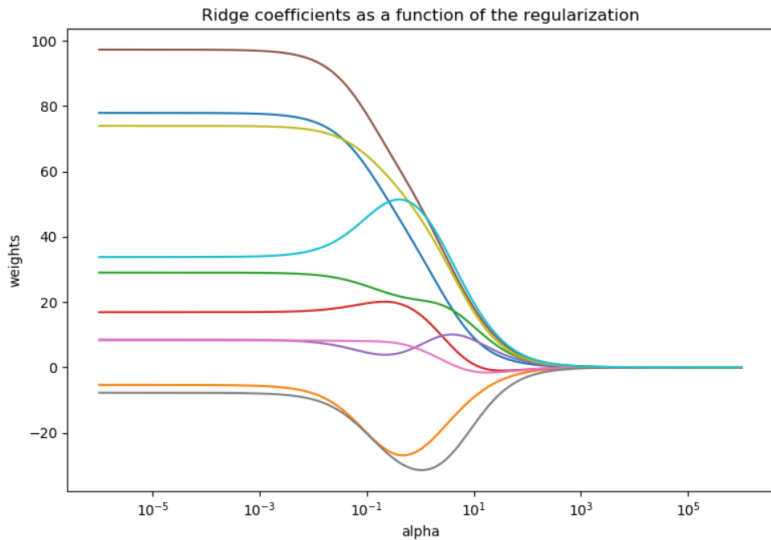


- Regularização  $\ell_1$  ou *lasso regression*:

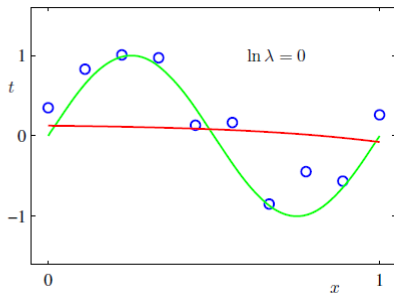
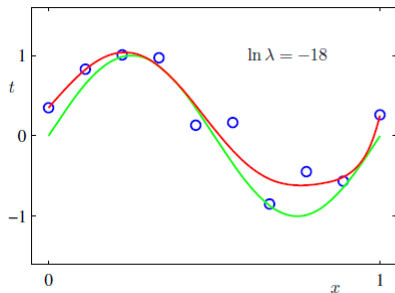
$$\Omega(\mathbf{w}) = \|\mathbf{w}\|_1 = |w_0| + |w_1| + \cdots + |w_n|$$

- Favorece modelos com poucos coeficientes não-nulos

## Efeito da regularização $\ell_2$



## Exemplo (com regularização $\ell_2$ )



	$\ln \lambda = -\infty$	$\ln \lambda = -18$	$\ln \lambda = 0$
$w_0^*$	0.35	0.35	0.13
$w_1^*$	232.37	4.74	-0.05
$w_2^*$	-5321.83	-0.77	-0.06
$w_3^*$	48568.31	-31.97	-0.05
$w_4^*$	-231639.30	-3.89	-0.03
$w_5^*$	640042.26	55.28	-0.02
$w_6^*$	-1061800.52	41.32	-0.01
$w_7^*$	1042400.18	-45.95	-0.00
$w_8^*$	-557682.99	-91.53	0.00
$w_9^*$	125201.43	72.68	0.01

# Mínimos Quadrados com Regularização $\ell_2$

- ▶ Função custo:

$$J(\mathbf{w}) = \frac{1}{m} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda \frac{1}{m} \|\mathbf{w}\|^2$$

- ▶ Gradiente:

$$\nabla J(\mathbf{w}) = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \frac{2}{m} \mathbf{w}$$

- ▶ Equação normal:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

- ▶ **Obs:** resolve o problema da invertibilidade

## Mínimos Quadrados com Regularização $\ell_2$

- ▶ Por convenção, normalmente **não se aplica regularização em  $w_0$** :

$$\Omega(\mathbf{w}) = w_1^2 + \cdots + w_n^2$$

- ▶ Equações adaptáveis definindo  $\mathbf{L} = \begin{bmatrix} 0 & \mathbf{0}_{1 \times n} \\ \mathbf{0}_{n \times 1} & \mathbf{I}_n \end{bmatrix}$

- ▶ Função custo:

$$J(\mathbf{w}) = \frac{1}{m} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda \frac{1}{m} \mathbf{w}^T \mathbf{L} \mathbf{w}$$

- ▶ Gradiente:

$$\nabla J(\mathbf{w}) = \frac{2}{m} \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \frac{2}{m} \mathbf{L} \mathbf{w}$$

- ▶ Equação normal:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{L})^{-1} \mathbf{X}^T \mathbf{y}$$

# Avaliação do Modelo

- ▶ A função custo regularizada  $J(\mathbf{w})$  é usada como **função objetivo** do problema de otimização
- ▶ No entanto, o **desempenho do modelo** (ex: MSE) deve continuar sendo avaliado pela função custo **sem regularização**

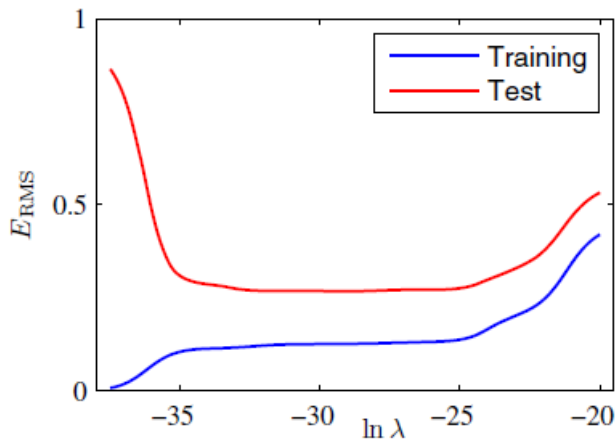
$$J_{\text{test}}(\mathbf{w}) = \frac{1}{m} \|\mathbf{X}_{\text{test}} \mathbf{w} - \mathbf{y}_{\text{test}}\|^2$$
$$J_{\text{train}}(\mathbf{w}) = \frac{1}{m} \|\mathbf{X}_{\text{train}} \mathbf{w} - \mathbf{y}_{\text{train}}\|^2$$

- ▶ Regularização é apenas um “viés” (ou preferência) em favor de modelos mais simples introduzido durante o projeto, não interfere na avaliação do modelo no “mundo real”

# Escolha de Hiperparâmetros

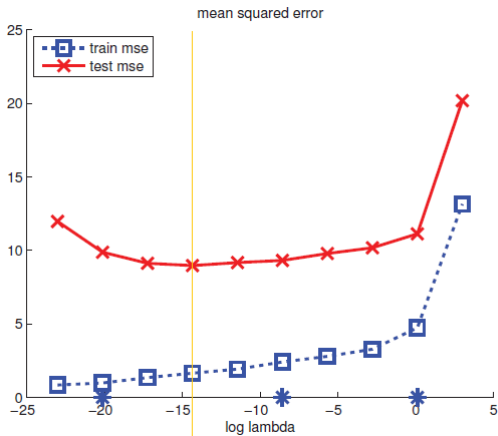
- ▶  $\lambda$  é um parâmetro do modelo que não pode ser determinado durante o treinamento, o que é chamado de **hiperparâmetro**
- ▶ O aumento de  $\lambda$  sempre piora o desempenho no conjunto de treinamento (**por quê?**) mas tende a melhorar a generalização, isto é, tende a reduzir o gap  $J_{\text{test}} - J_{\text{train}}$ 
  - ▶  $\lambda$  muito pequeno pode causar overfitting
  - ▶  $\lambda$  muito grande pode causar underfitting
- ▶ O valor ótimo de  $\lambda$  é o que minimiza  $J_{\text{test}}$ —mas como escolhê-lo durante o projeto?
  - ▶ O conjunto de teste **nunca** deve ser usado durante o desenvolvimento do modelo (**por quê?**)

## Exemplo

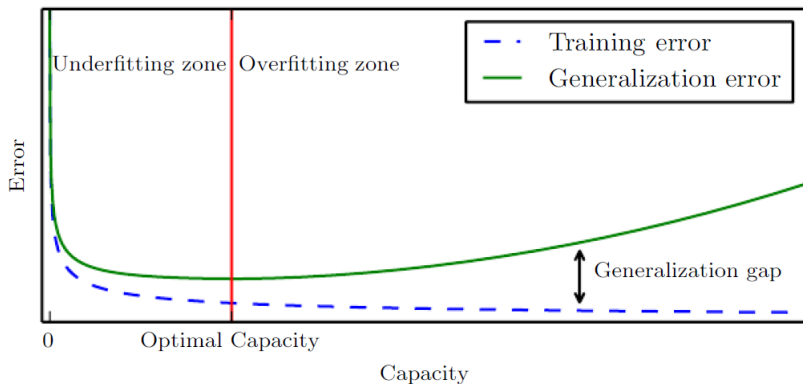




# Exemplo



# Exemplo



- ▶ Obs: aumento de  $\lambda$  reduz a capacidade do modelo
- ▶ Ex:  $C = 1/\lambda$

# Escolha de Hiperparâmetros

- ▶ A escolha de hiperparâmetros deve ser feita usando um conjunto separado (diferente dos conjuntos de treinamento e teste), chamado de **conjunto de validação** (ou **conjunto de desenvolvimento**)

$$J_{\text{val}}(\mathbf{w}) = \frac{1}{m} \|\mathbf{X}_{\text{val}} \mathbf{w} - \mathbf{y}_{\text{val}}\|^2$$

- ▶ Exemplo: 60% treinamento, 20% validação, 20% teste
- ▶ Este método é chamado de *validação cruzada* **hold-out**
- ▶ Escolhem-se os hiperparâmetros que minimizam  $J_{\text{val}}(\mathbf{w})$ :

$$\min_{\lambda} J_{\text{val}}(\mathbf{w}(\lambda))$$

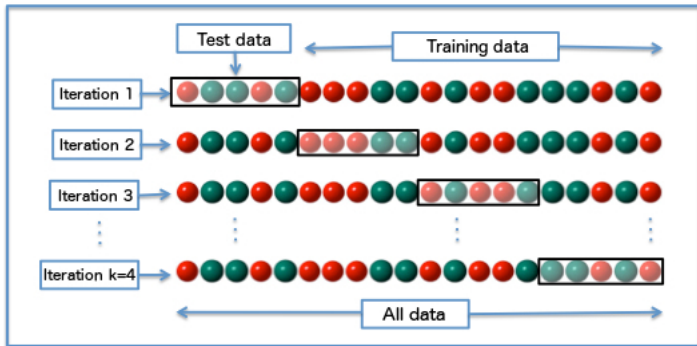
onde

$$\mathbf{w}(\lambda) = \underset{\mathbf{w}}{\operatorname{argmin}} J(\mathbf{w}; \lambda)$$

## Outros Métodos de Validação Cruzada (*Cross-Validation*)

- ▶ Uma alternativa a usar conjuntos fixos é calcular a média de  $J_{\text{val}}$  ao longo de  $k$  rodadas de divisão dos dados em conjuntos de treinamento e validação (com proporção fixa)
  - ▶ **Vantagem:** Evita “desperdiçar” amostras de treinamento  $\rightarrow$  útil quando o conjunto de dados é pequeno
  - ▶ **Desvantagem:** Mais complexo do que o hold-out ( $k = 1$ )
- ▶ **Método de subamostragem aleatória:** A cada rodada, o conjunto de validação é escolhido aleatoriamente dentre os dados disponíveis
- ▶ **Método  $k$ -fold:** os  $k$  conjuntos de validação usados nas  $k$  rodadas formam uma **partição** dos dados disponíveis (proporção  $1/k$ )
  - ▶ Tipicamente,  $k = 5$

## Validação Cruzada: Método $k$ -fold



Obs: nesta figura, Test data = Validation data

Obs: **All data** = todos os dados disponíveis para treinamento+validação, i.e., excetuando-se o conjunto de teste

# Re-Treinamento

- ▶ Opcionalmente, após a escolha de  $\lambda$ , pode ser feito um último treinamento usando todos os dados disponíveis para treinamento + validação