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

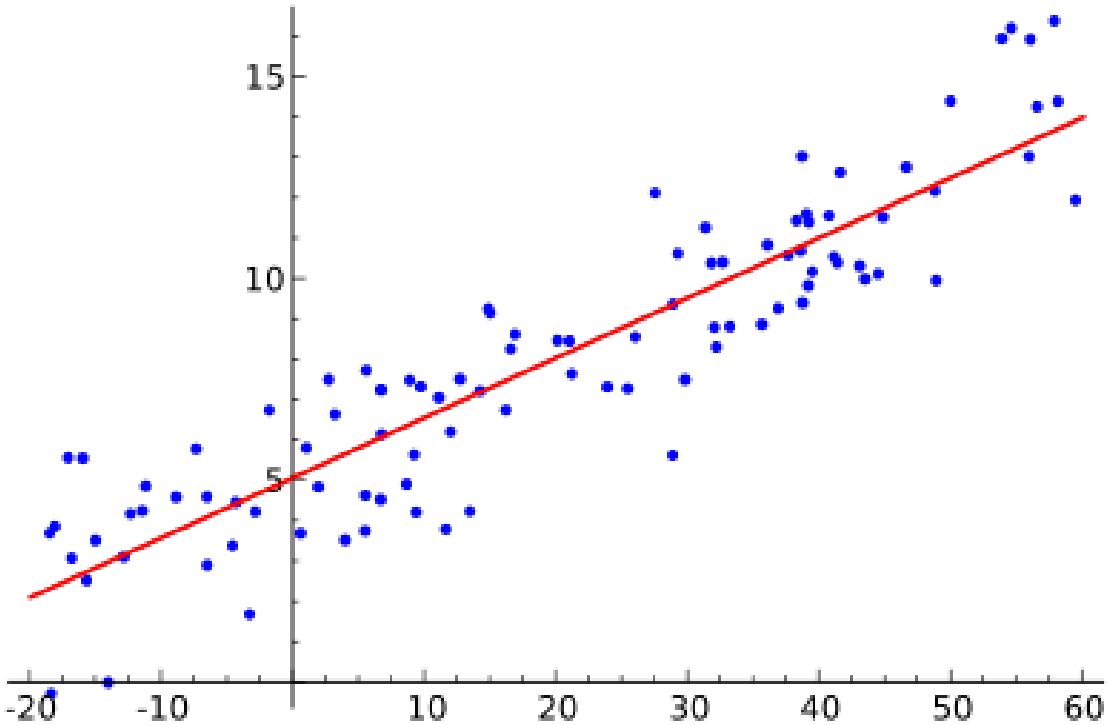
	TV	Radio	Newspaper	Sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

- Variáveis aleatórias:
  - $\mathbf{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ 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: $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$

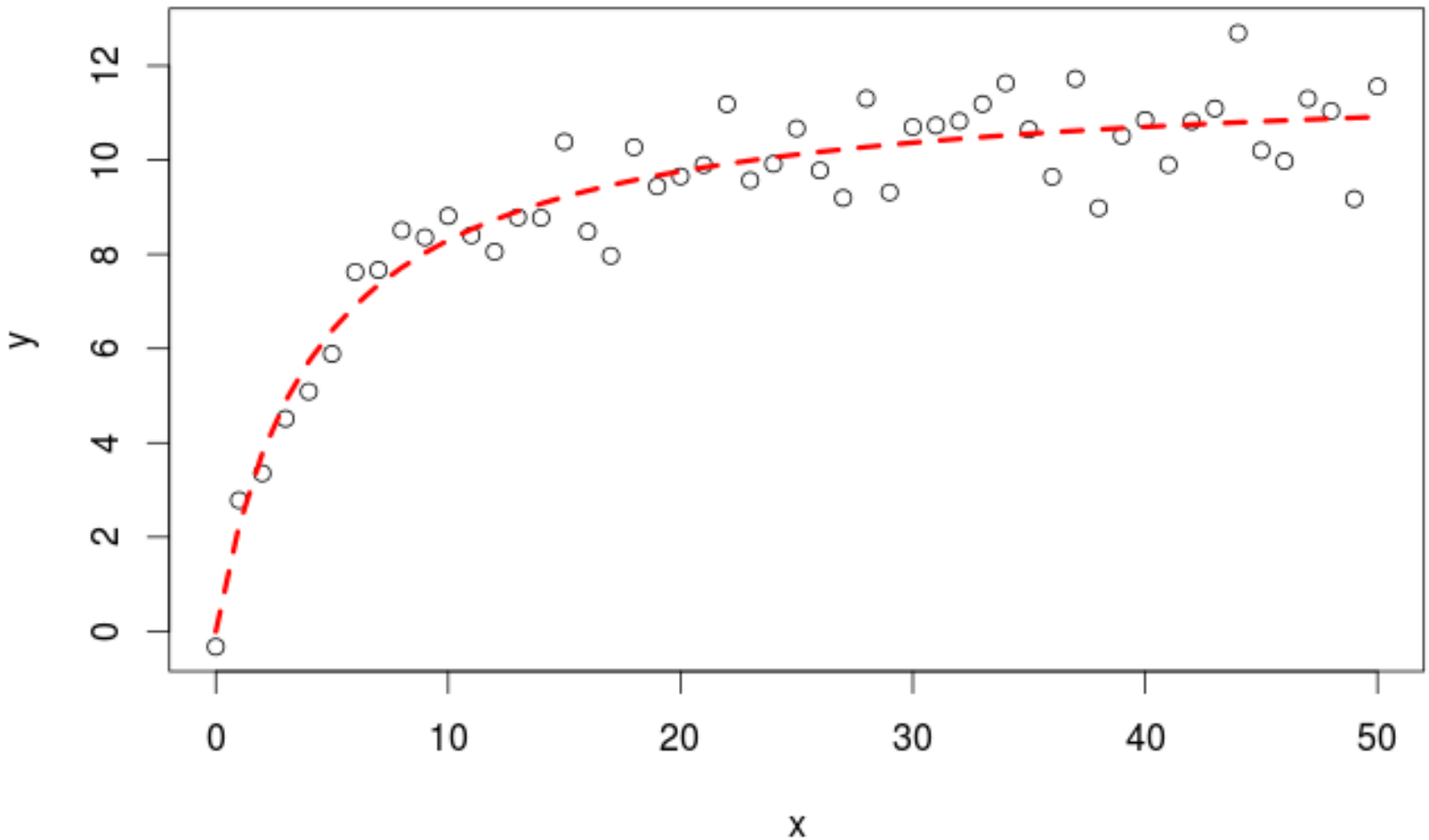
Regressão

- Um modelo de aprendizado determina uma função hipótese $f : \mathbb{R}^n \to \mathbb{R}$ 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

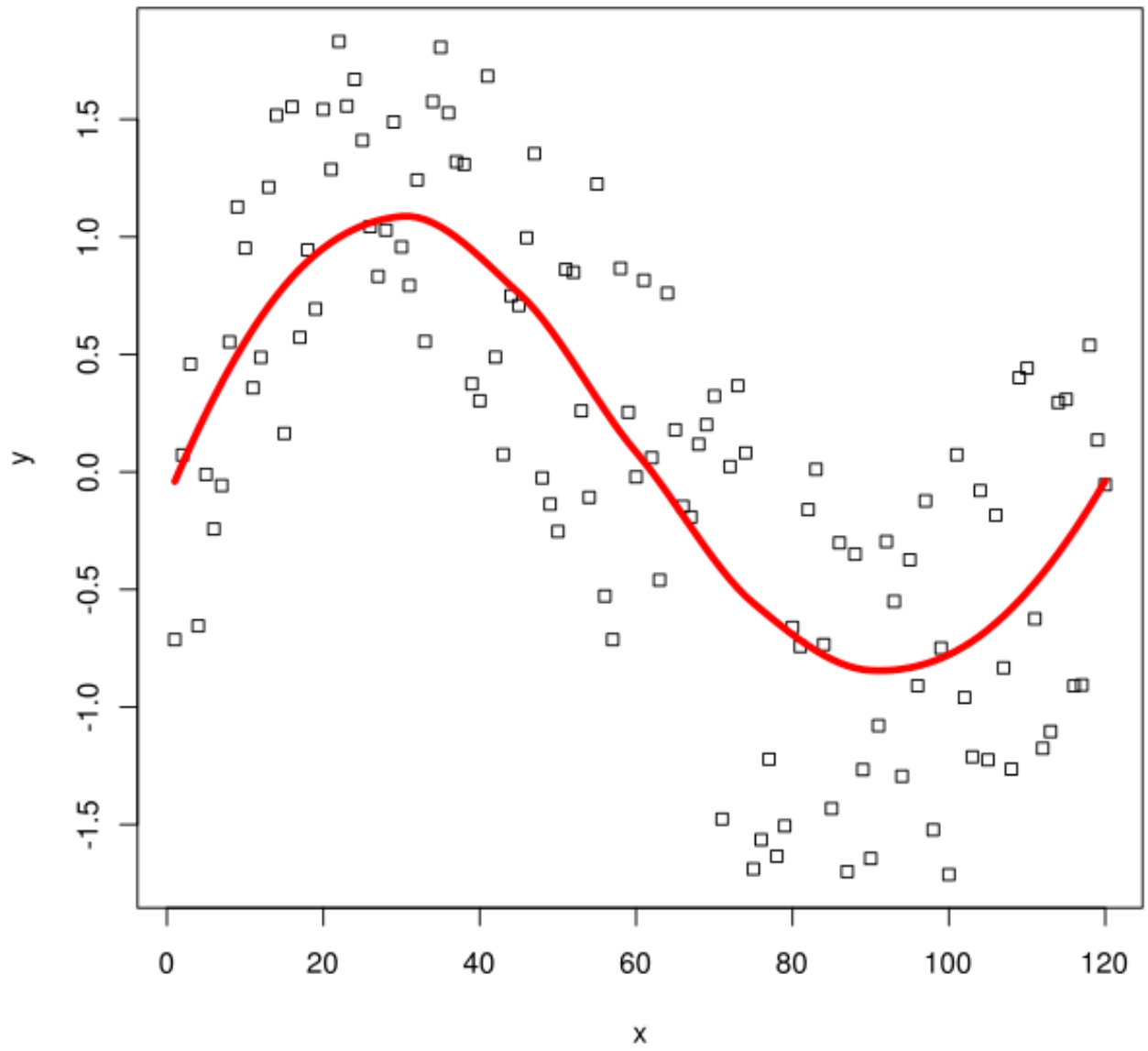
Exemplos



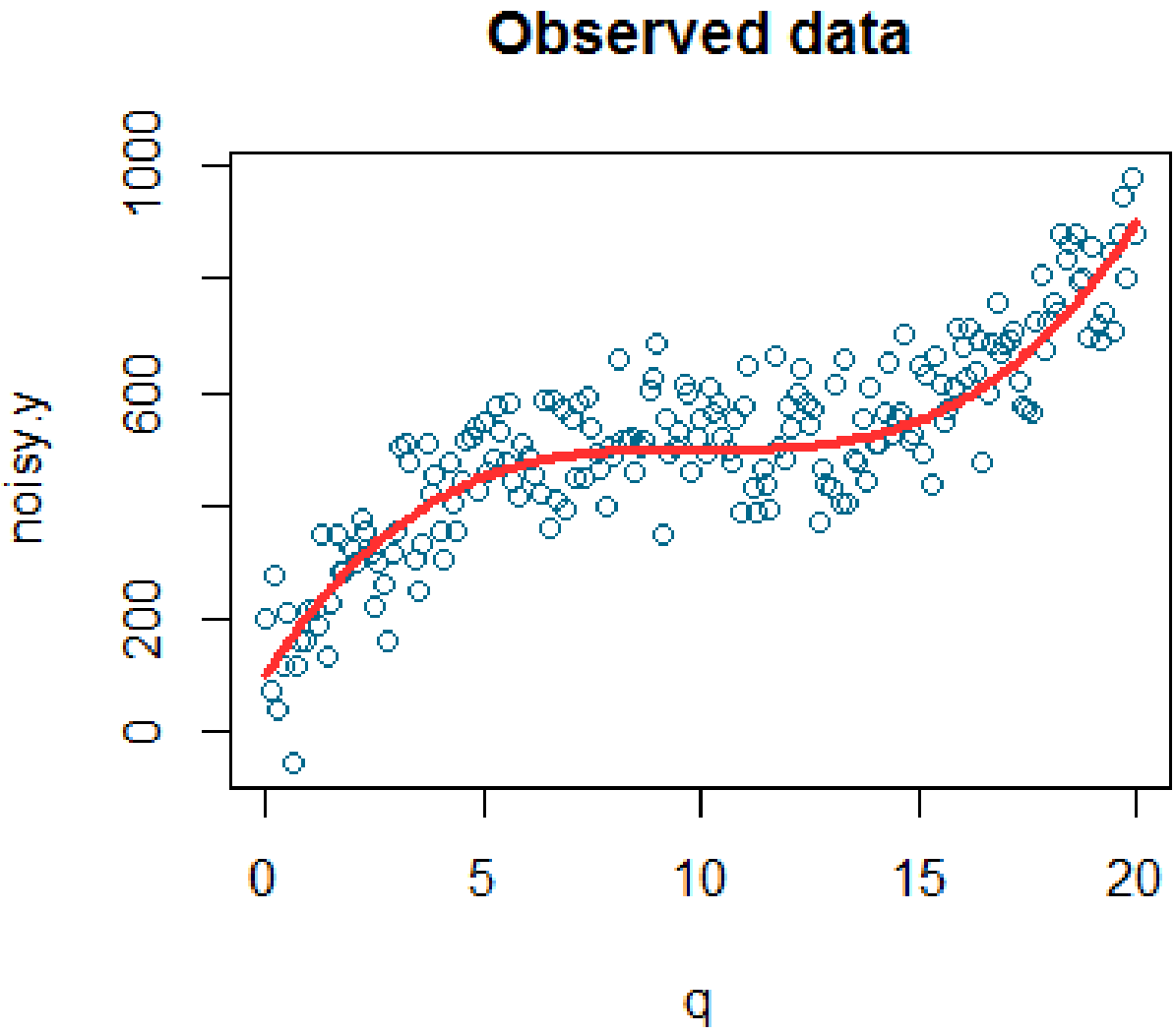
Exemplos



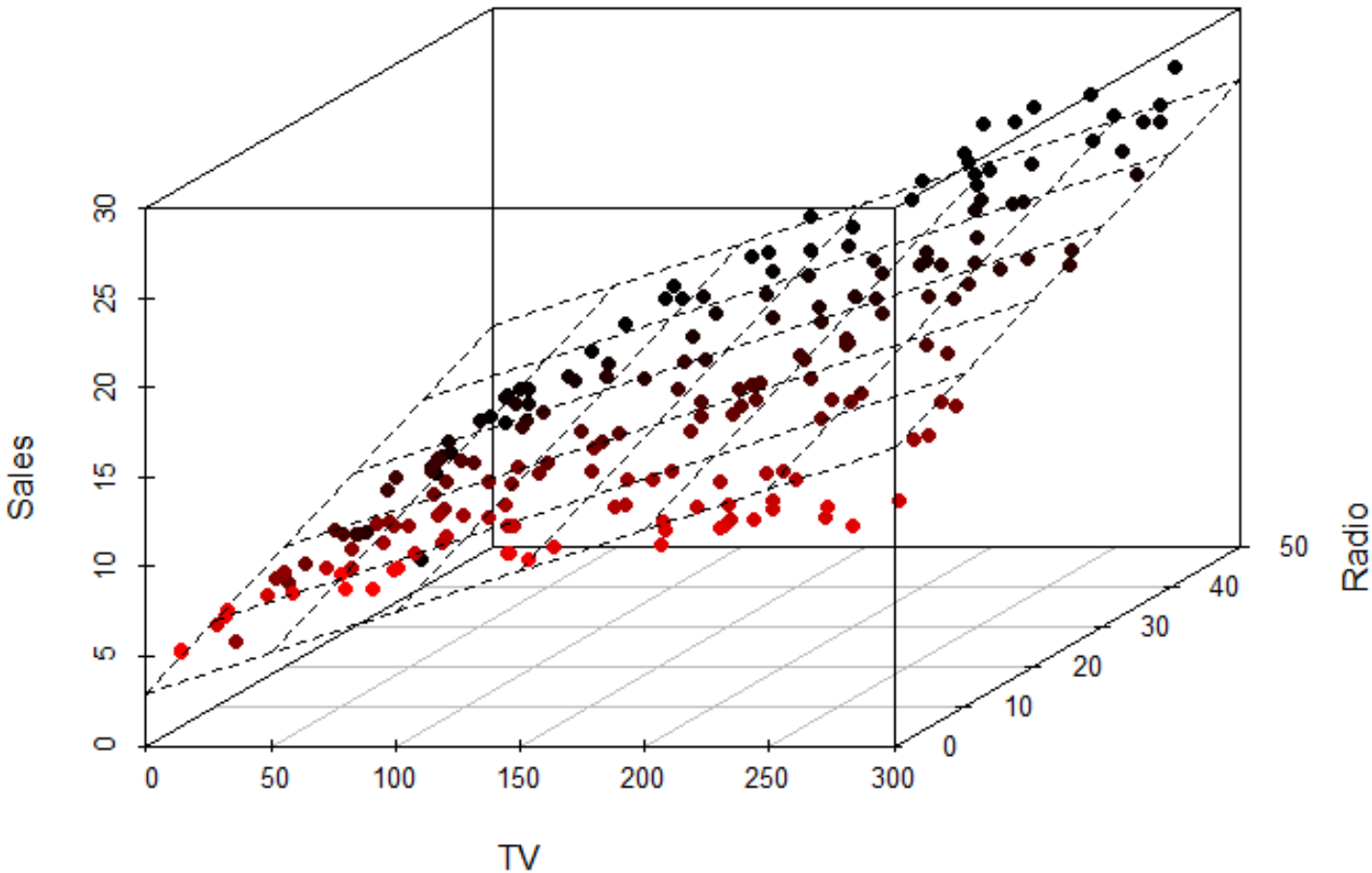
Exemplos



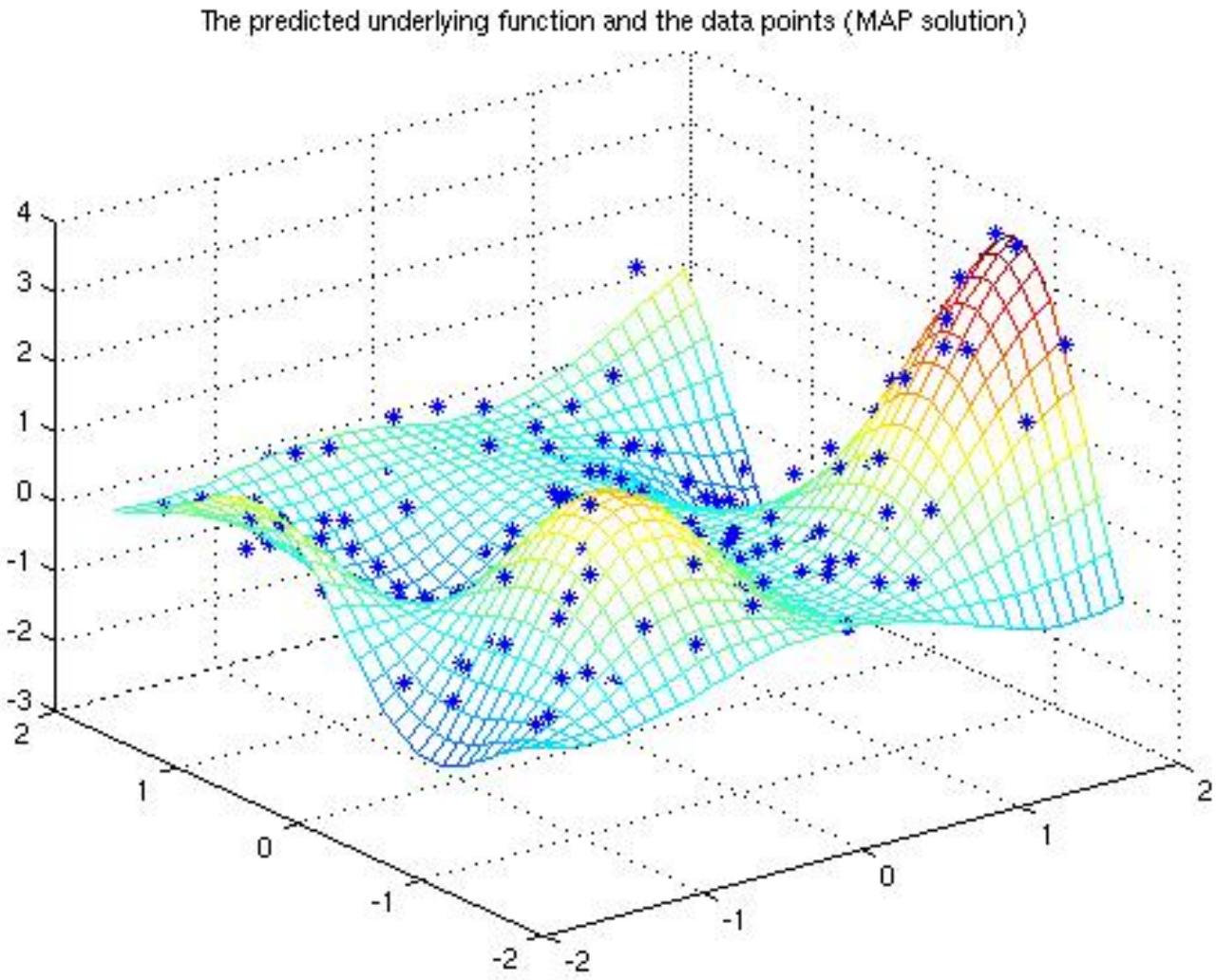
Exemplos



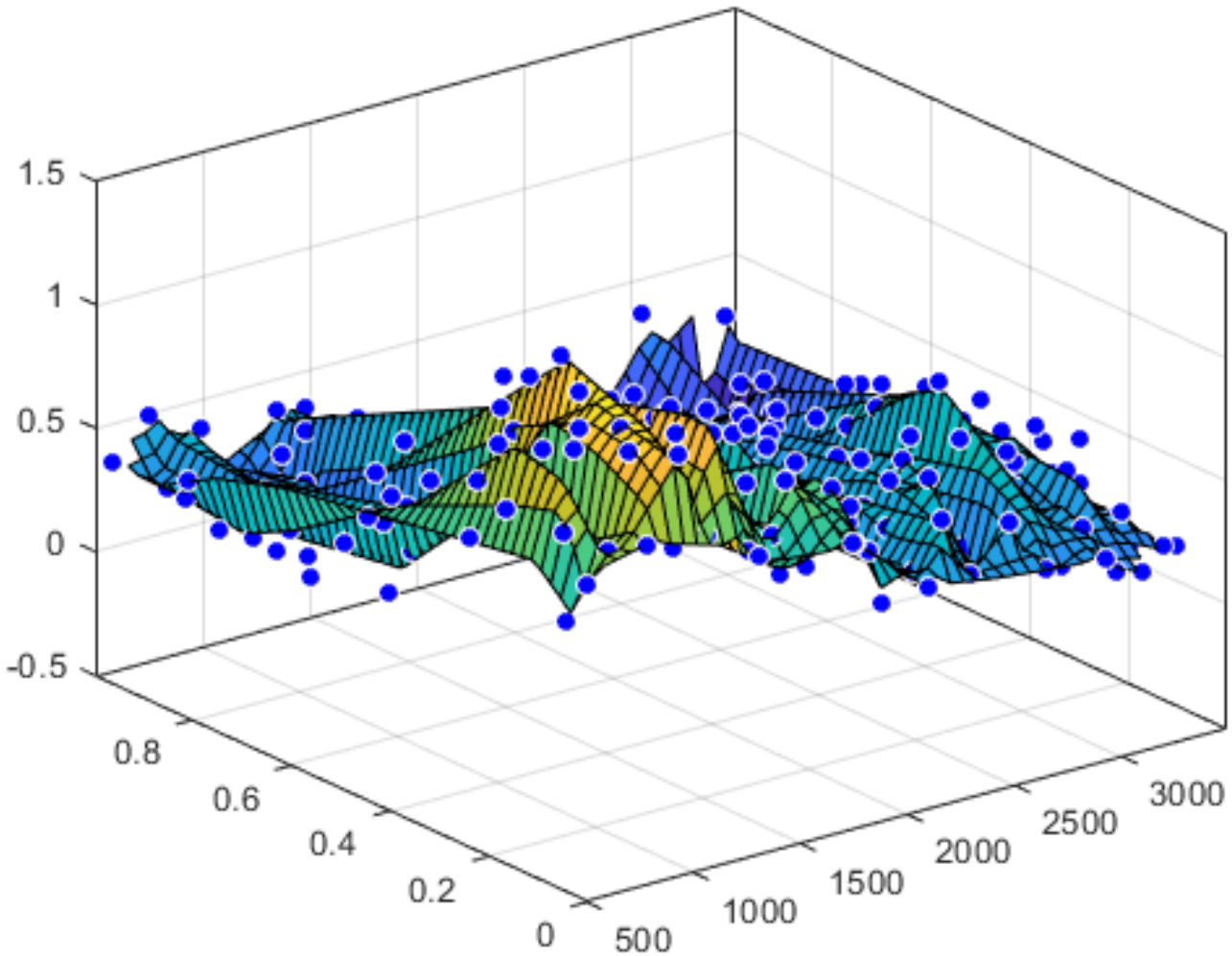
Exemplos


Sales
0
5
10
15
20
25
30
TV
0
50
100
150
200
250
300
Radio
0
10
20
30
40
50

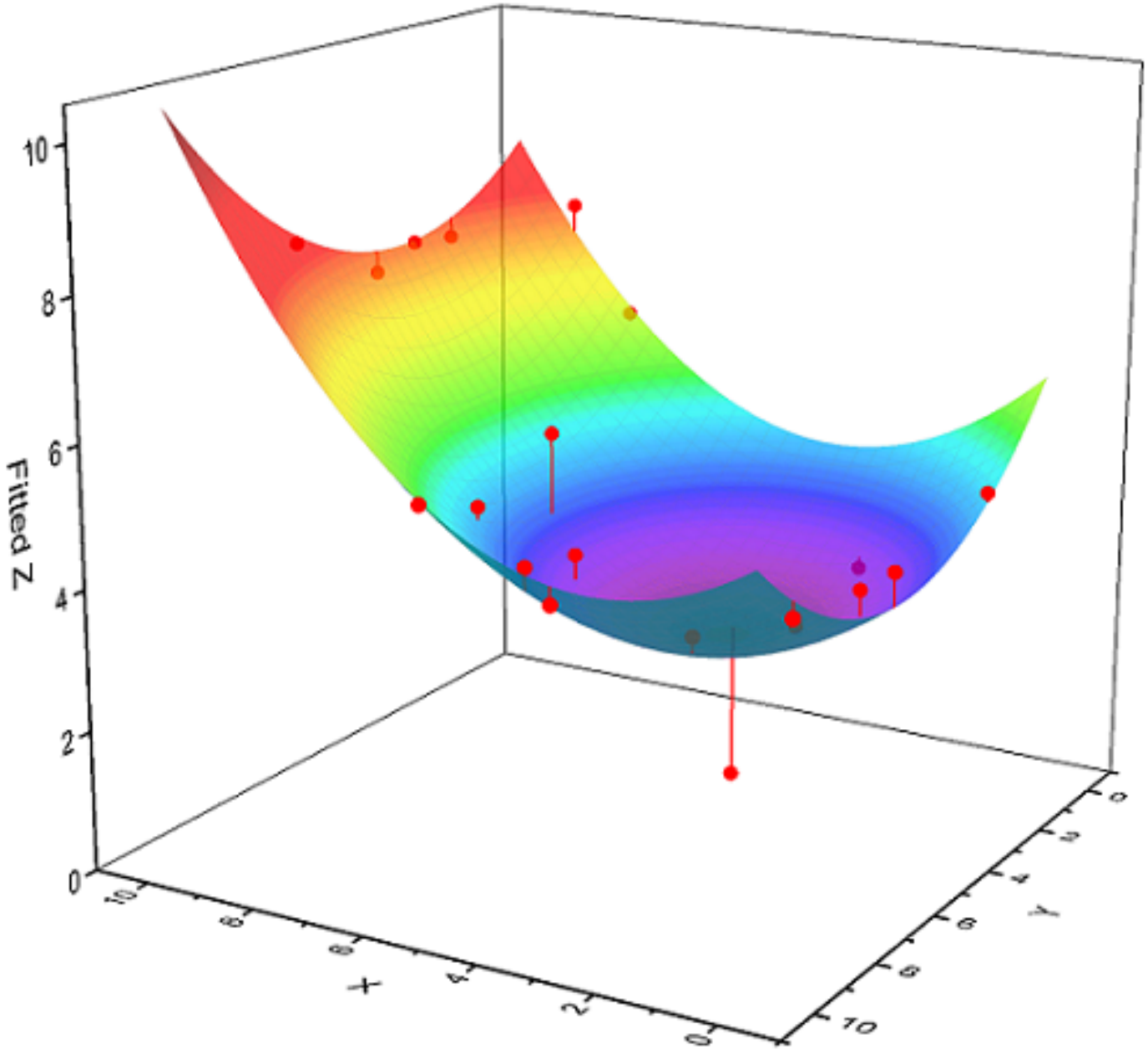
Exemplos


The predicted underlying function and the data points (MAP solution)
4
3
2
1
0
-1
-2
-3
2
1
0
-1
-2
-2
-1
0
1
2

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 ℓ_1):

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

  - Erro quadrático (perda ℓ_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 = erro quadrático $\implies$ J = erro quadrático médio (MSE)
  - L = erro absoluto $\implies$ J = 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 $f \in \mathcal{H}$ (dentre 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

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

  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:

$$\hat{y} = f(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_n x_n$$

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

$$\hat{y} = f(x) = w_0 + w_1 x = b + wx$$

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

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

onde $b = w_0$ e $\mathbf{w} = \begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix}^T$

- 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 $\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_n \end{bmatrix}^T$ e $\mathbf{x} = \begin{bmatrix} 1 & x_1 & \cdots & x_n \end{bmatrix}^T$

  - 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

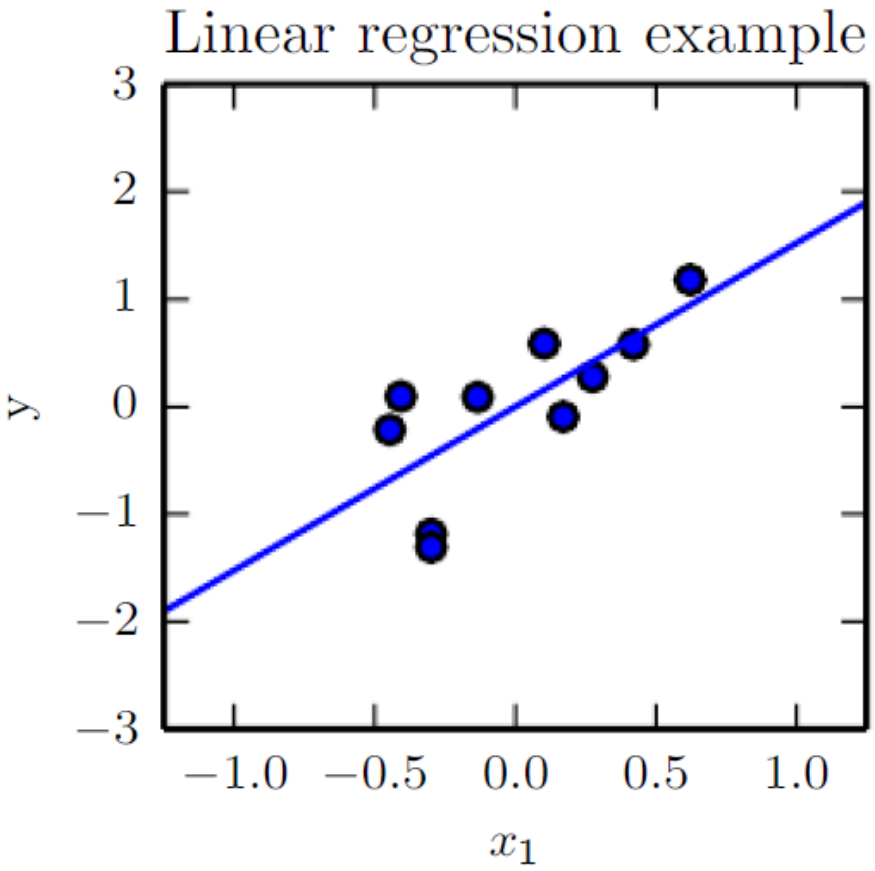
$$J(\mathbf{w}) = J(f) = \frac{1}{m}\sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}), y^{(i)}) = \frac{1}{m}\sum_{i=1}^{m} L(\mathbf{w}^T\mathbf{x}^{(i)}, y^{(i)})$$

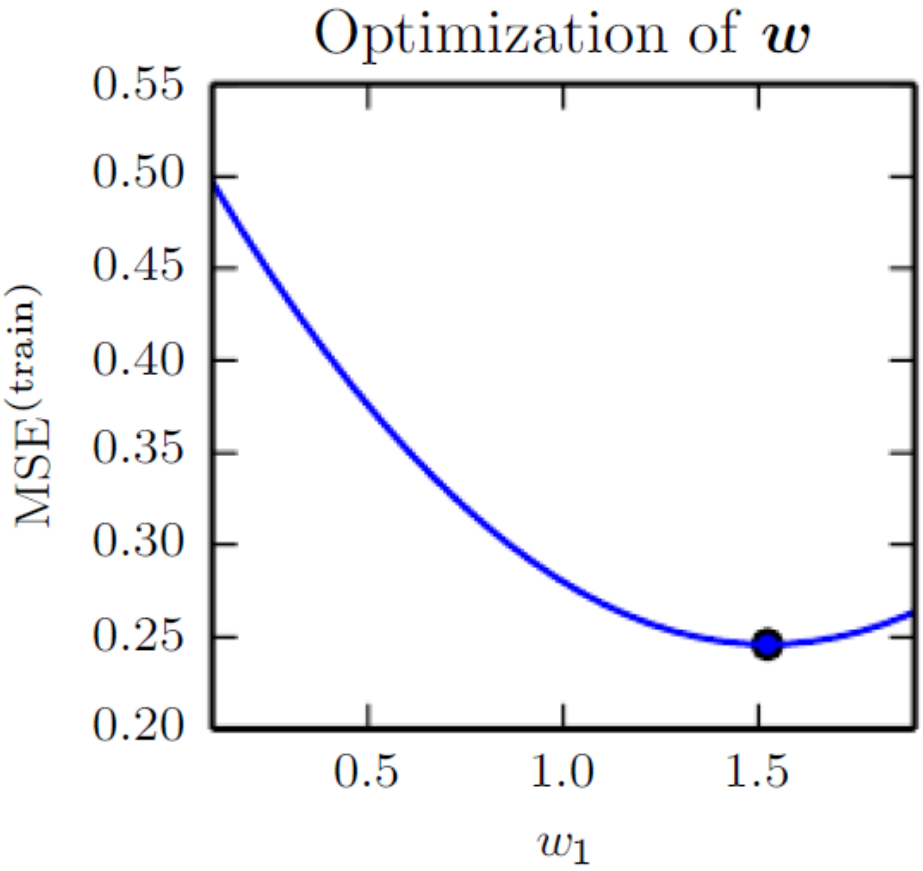
- Assumindo como função perda o erro quadrático $L(\hat{y}, y) = (\hat{y} - y)^2$:

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

- 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


Linear regression example
3
2
1
0
−1
−2
−3
y
−1.0
−0.5
0.0
0.5
1.0
x_1


Optimization of $\boldsymbol{w}$
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
$\mathrm{MSE}^{(\mathrm{train})}$
0.5
1.0
1.5
w_1

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

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

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 $\hat{\mathbf{y}} = (\hat{y}^{(1)}, \ldots, \hat{y}^{(m)})^T = \mathbf{X}\mathbf{w}$ 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:

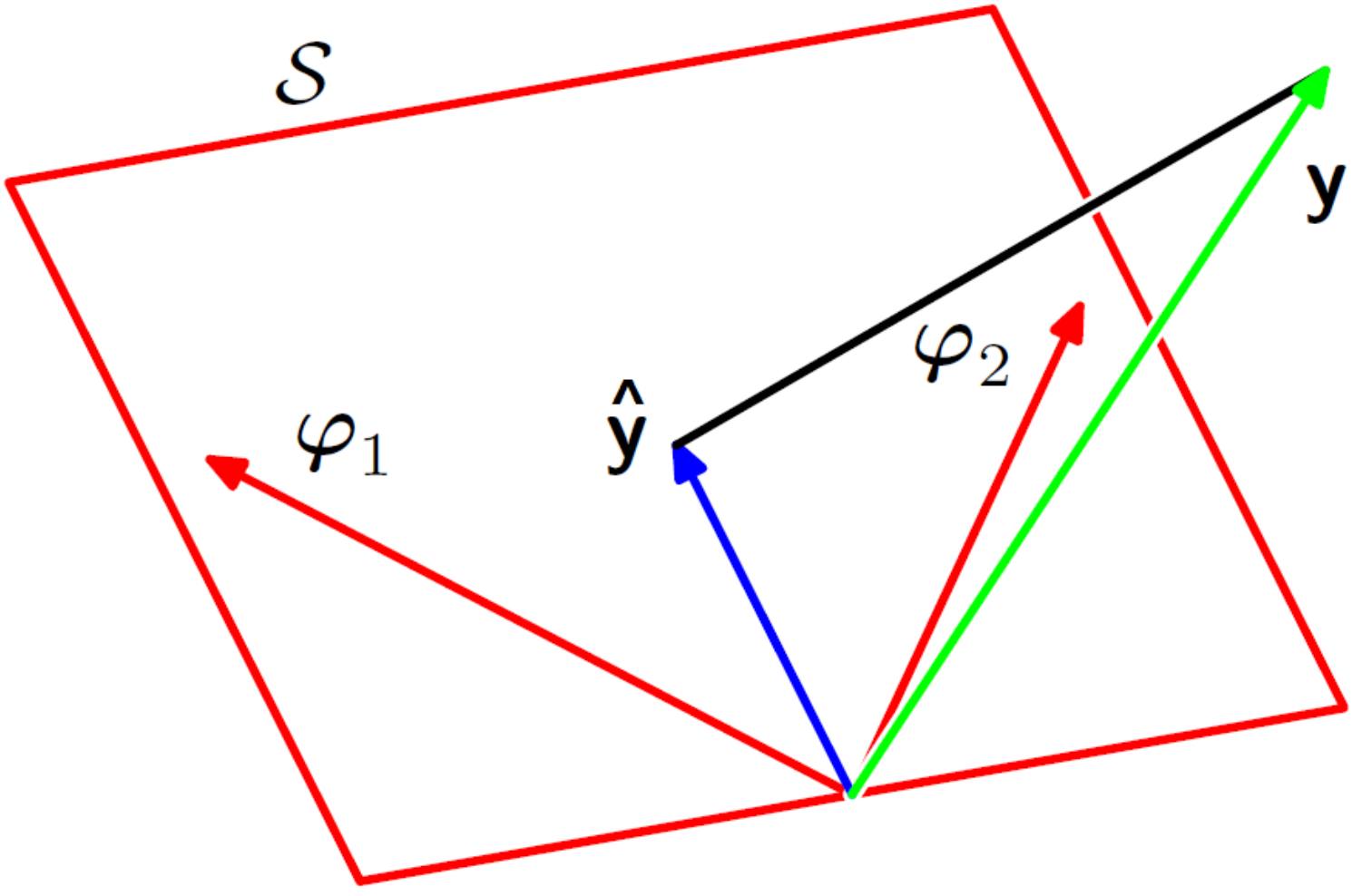
$$J(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2 = \frac{1}{m} \|\hat{\mathbf{y}} - \mathbf{y}\|^2$$

- 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 $\hat{y} = w_0 + w_1 x$?
- Regressão polinomial:

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d$$

- 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: $\{1, x, x^2, x^3, \ldots\}$
  - Base de Fourier: $\{1, \cos(\frac{2\pi}{T_0}jx), \text{sen}(\frac{2\pi}{T_0}jx), j = 1, 2, 3, \ldots\}$
- 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 $\mathbf{x} = (x_1, \ldots, x_n)^T$ através de funções arbitrárias $\varphi_i(\mathbf{x})$:

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

- 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

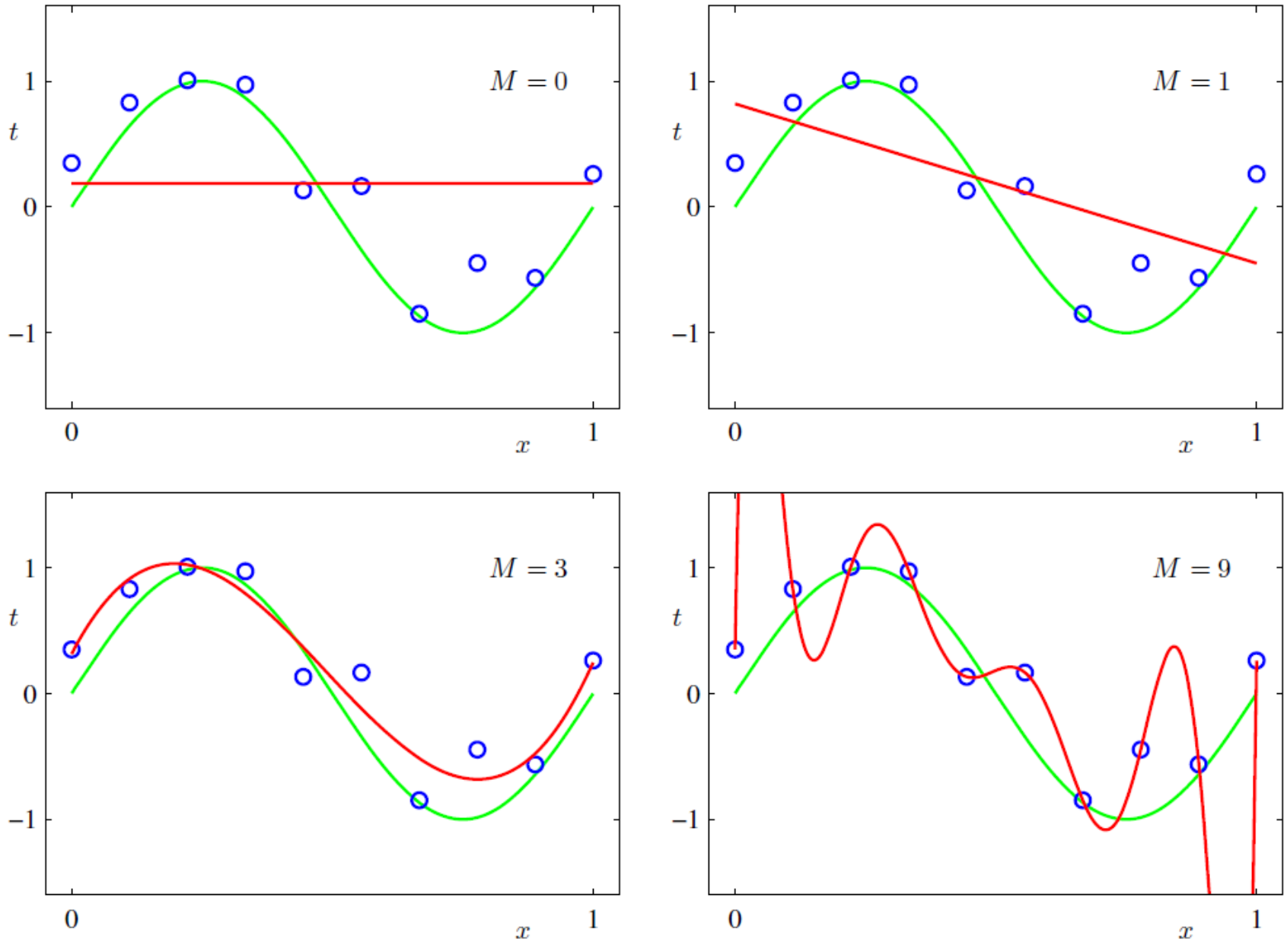
- O uso da equação normal requer que $\mathbf{X}^T\mathbf{X}$ seja inversível
- No entanto, $\mathbf{X}^T\mathbf{X}$ será singular (não-inversível):
  - se $n \geq m$; ou
  - se houver atributos redundantes (linearmente dependentes)
- Além disso, o aumento de n pode causar overfitting
- Uma solução para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

Overfitting e Regularização

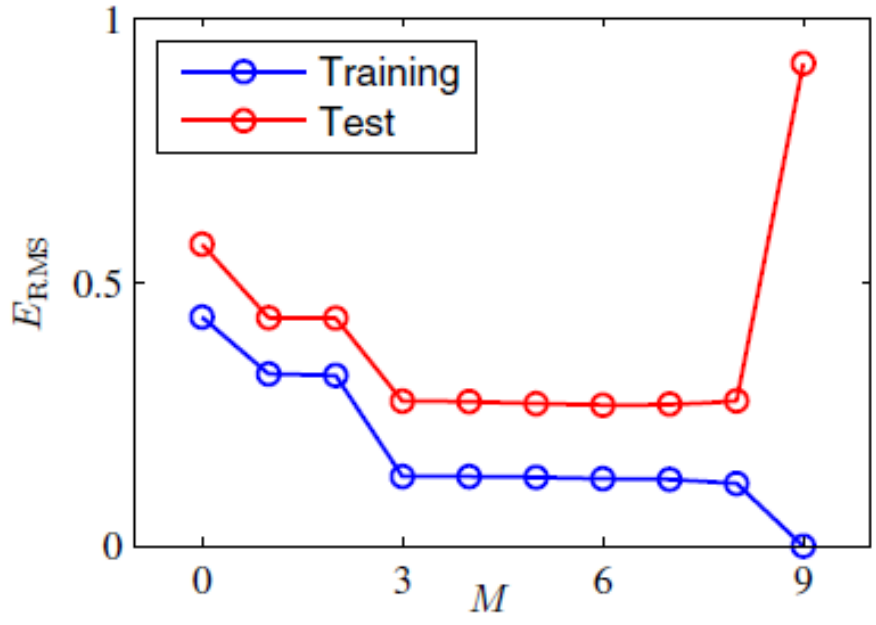
Overfitting

- 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

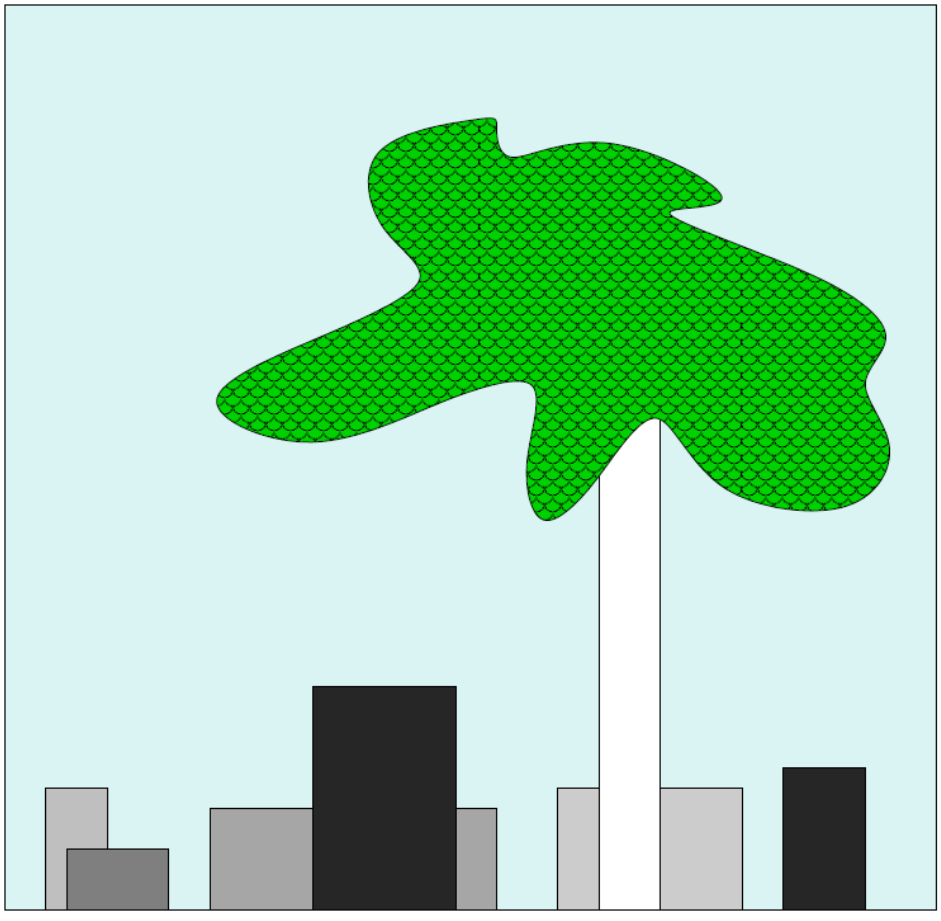


Exemplo



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

Navalha de Occam



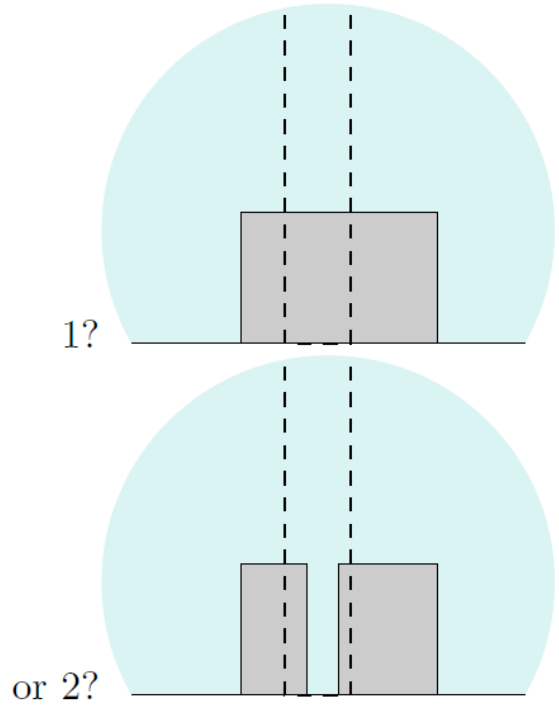


Figure 28.2. How many boxes are behind the tree?

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

Regularização

- Consiste em penalizar parâmetros que assumem valores muito elevados:

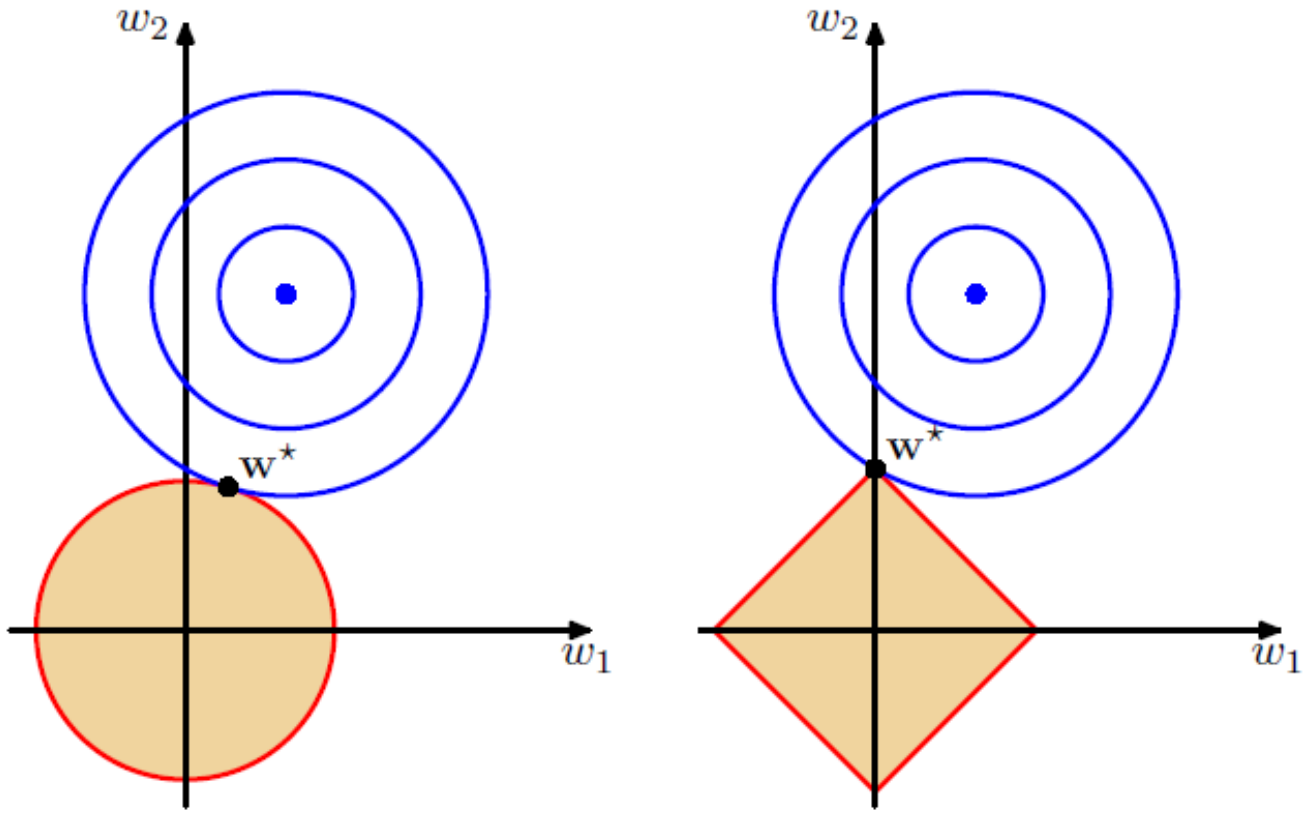
$$J(\mathbf{w}) = J_{\text{train}}(\mathbf{w}) + \lambda\Omega(\mathbf{w})$$

  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
  - λ 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 ℓ_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

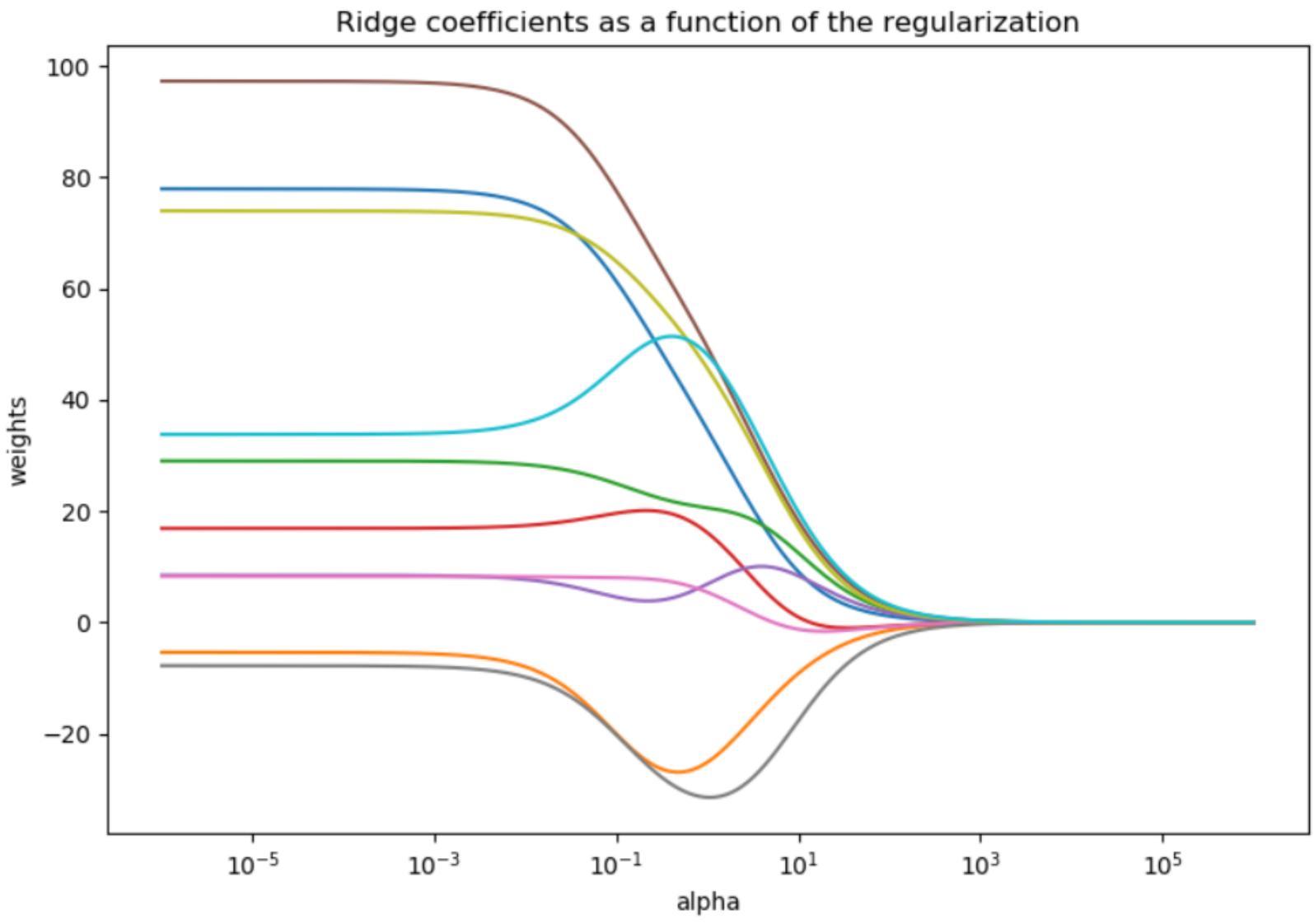


- Regularização ℓ_1 ou *lasso regression*:

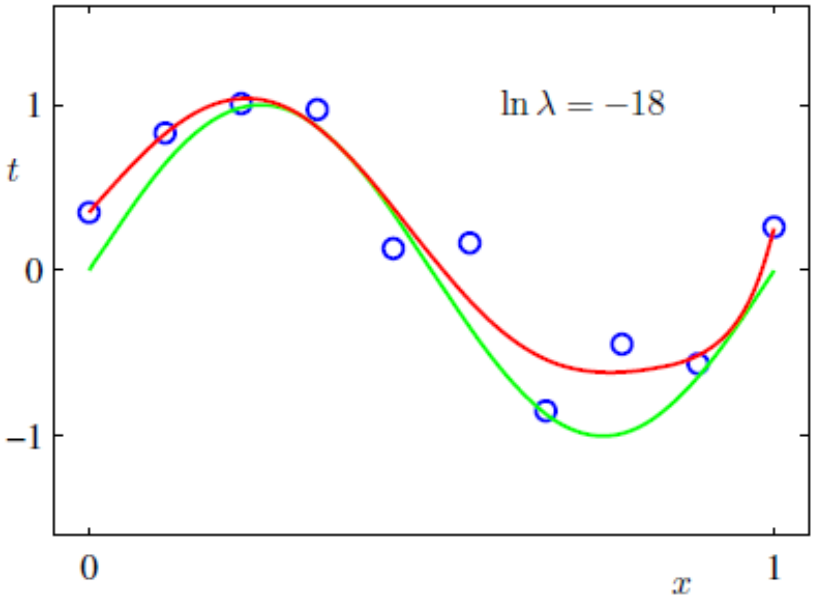
$$\Omega(\mathbf{w}) = \|\mathbf{w}\|_1 = |w_0| + |w_1| + \cdots + |w_n|$$

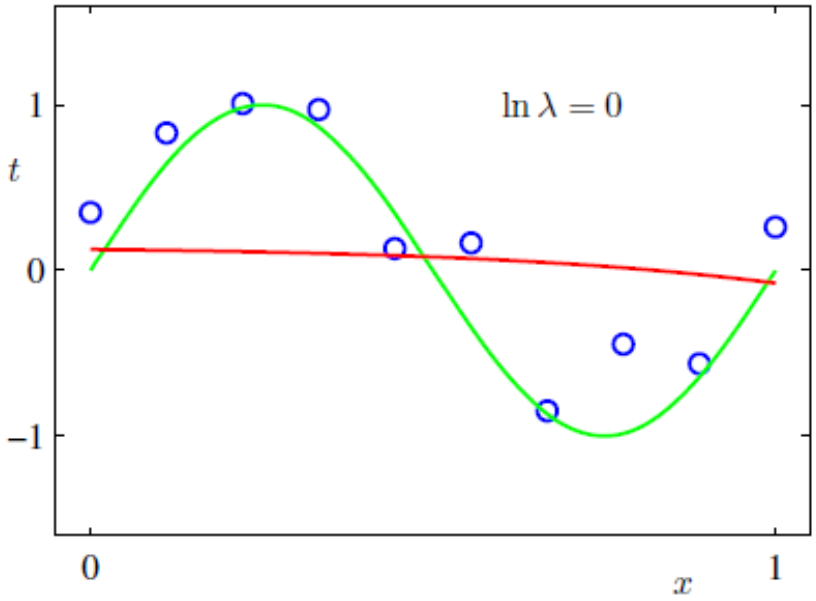
  - Favorece modelos com poucos coeficientes não-nulos

Efeito da regularização ℓ_2



Exemplo (com regularização ℓ_2)





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

Mínimos Quadrados com Regularização ℓ_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 ℓ_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

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

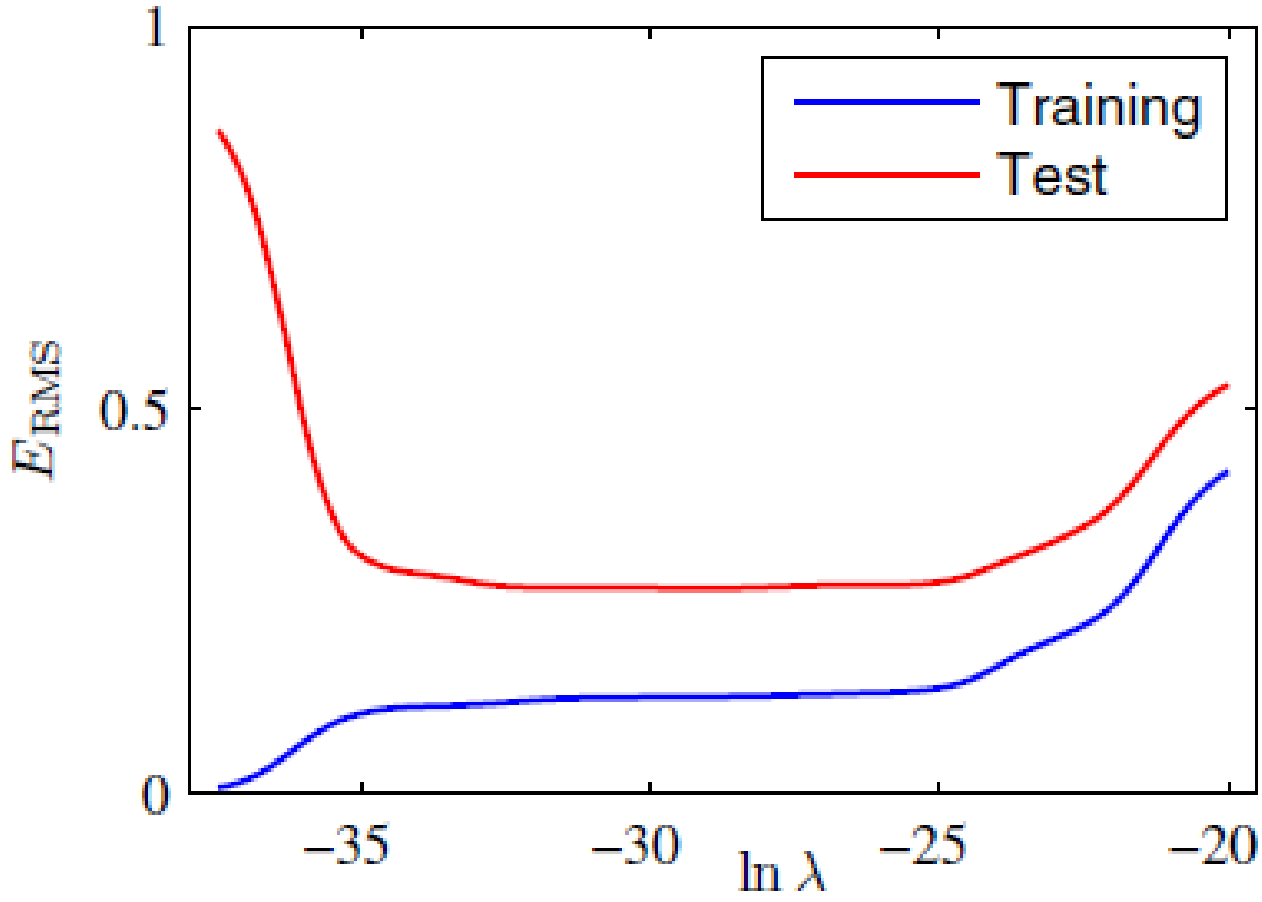
$$J_{\text{train}}(\mathbf{w}) = \frac{1}{m}\|\mathbf{X}_{\text{train}}\mathbf{w} - \mathbf{y}_{\text{train}}\|^2$$

- 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”

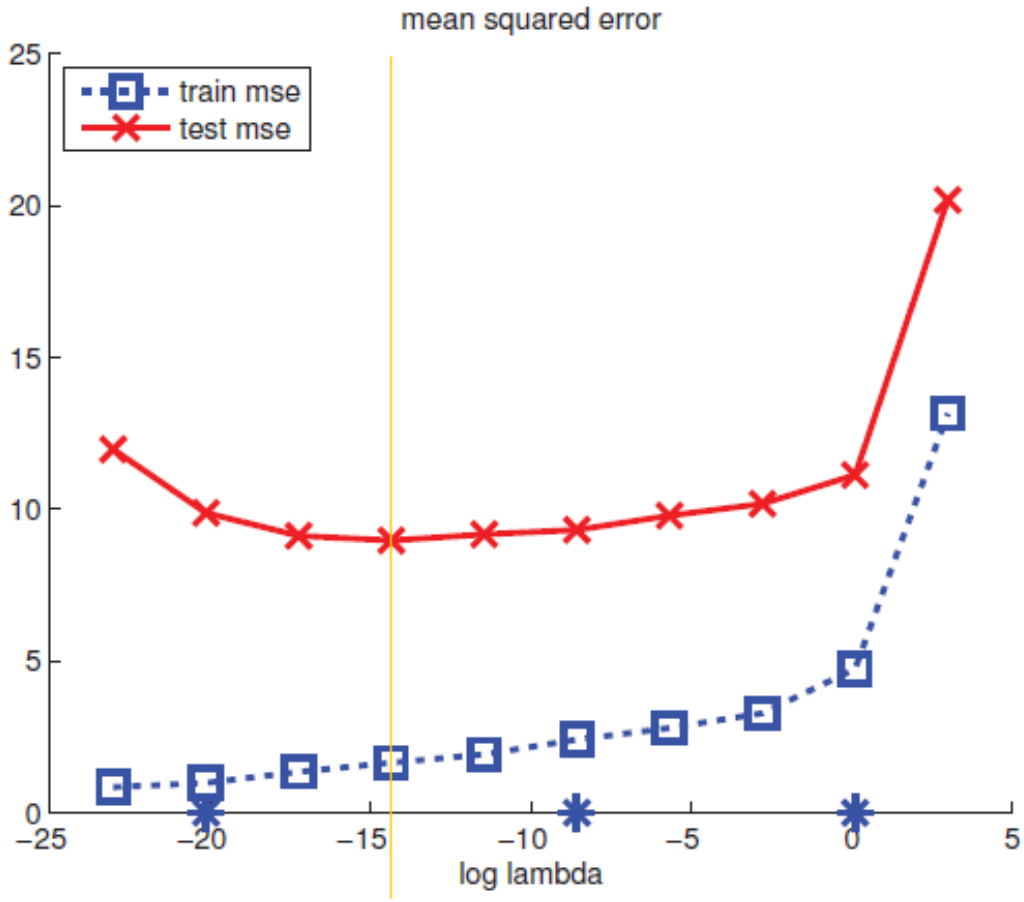
Escolha de Hiperparâmetros

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

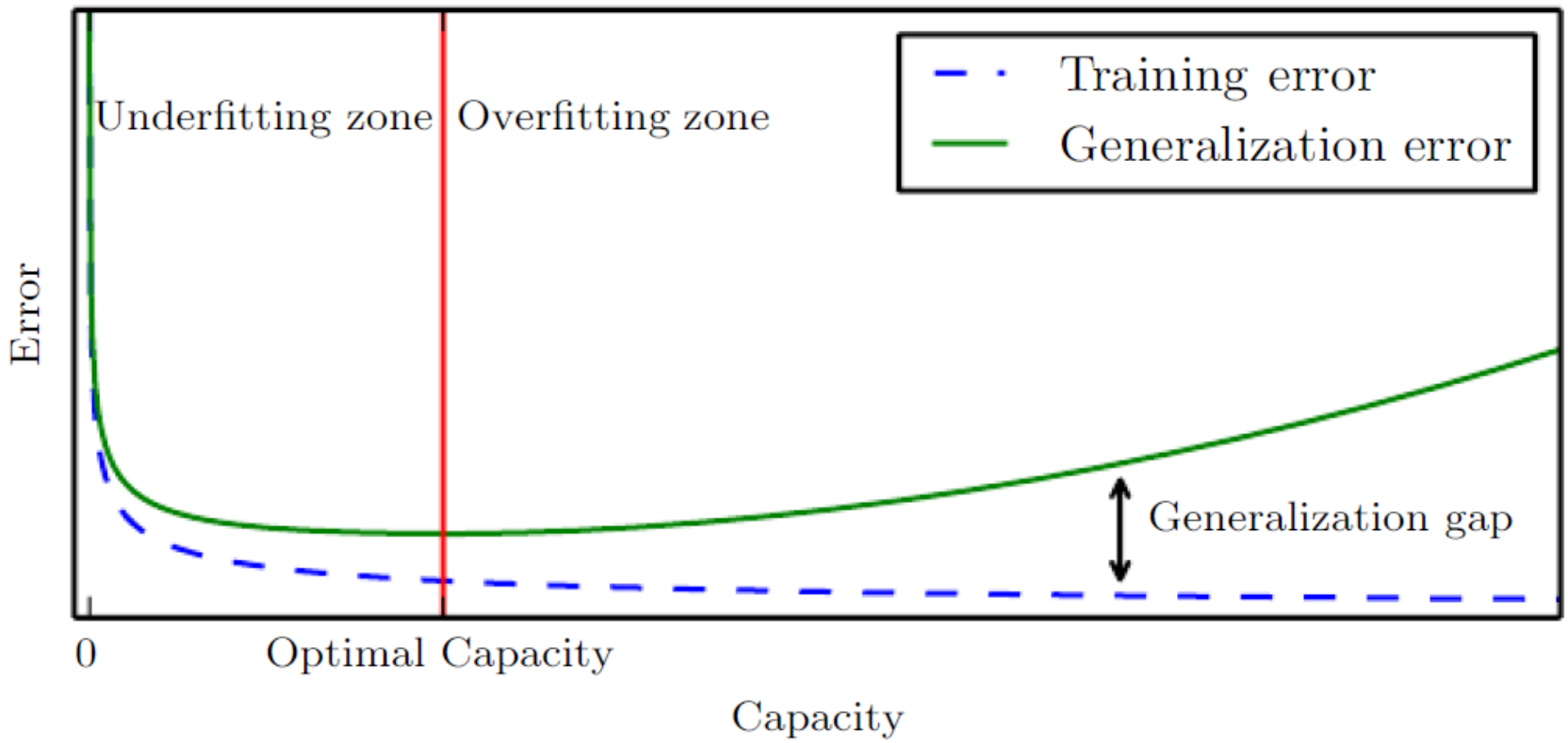
Exemplo



Exemplo


mean squared error
train mse
test mse
25
20
15
10
5
0
−25
−20
−15
−10
−5
0
5
log lambda

Exemplo



- Obs: aumento de λ 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 $J_{\text{val}}(\mathbf{w})$:

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

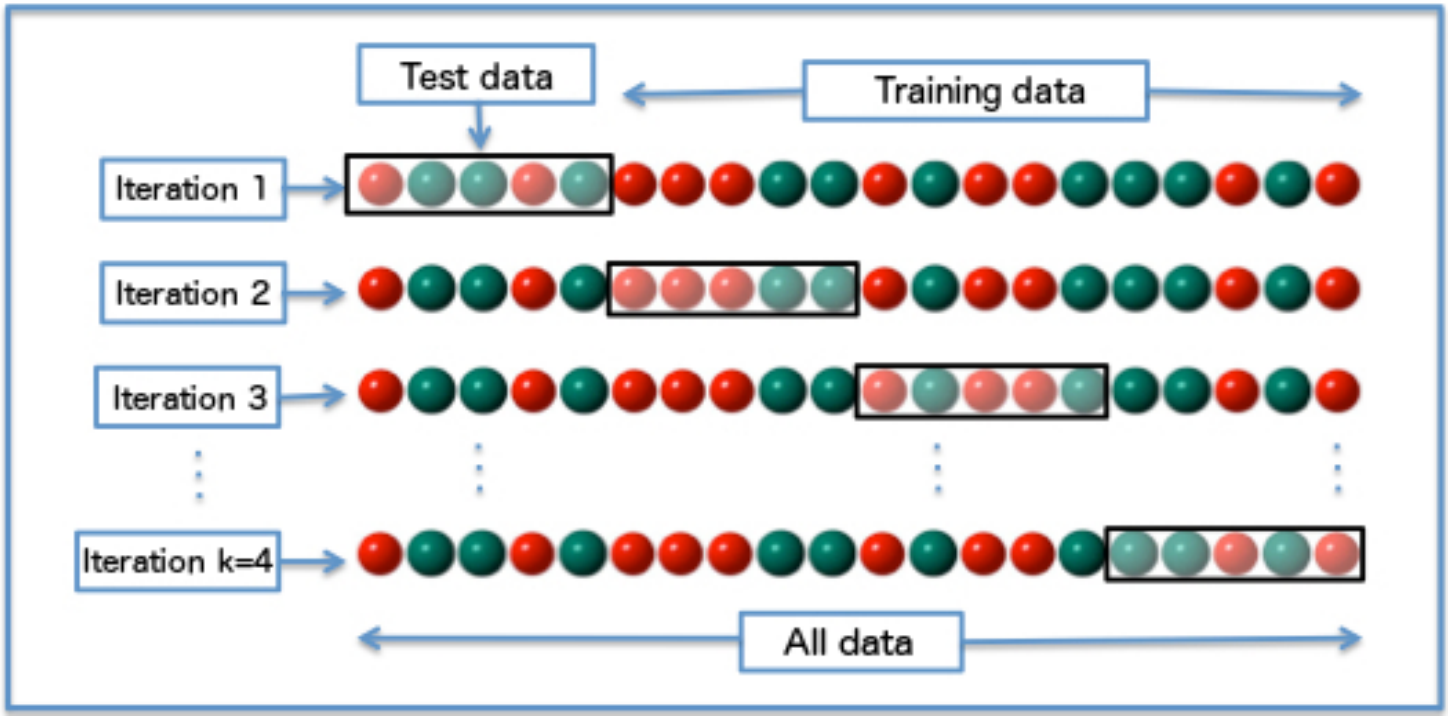
onde

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

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

- Uma alternativa a usar conjuntos fixos é calcular a média de J_{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 $\longrightarrow$ 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 λ, pode ser feito um último treinamento usando todos os dados disponíveis para treinamento + validação