Regressão Linear com Múltiplas Variáveis

Prof. Danilo Silva

EEL7514/EEL7513 - Tópico Avançado em Processamento de Sinais: Introdução ao Aprendizado de Máquina

EEL / CTC / UFSC

Tópicos

- Regressão linear: revisão
- Normalização de atributos
- Overfitting / regularization / escolha de hiperparâmetros
- Método do gradiente estocástico

Regressão Linear: Revisão

Regressão Linear

- Modelo de regressão linear:

$$\hat{y} = g(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_n x_n = \mathbf{w}^T \mathbf{x}$$

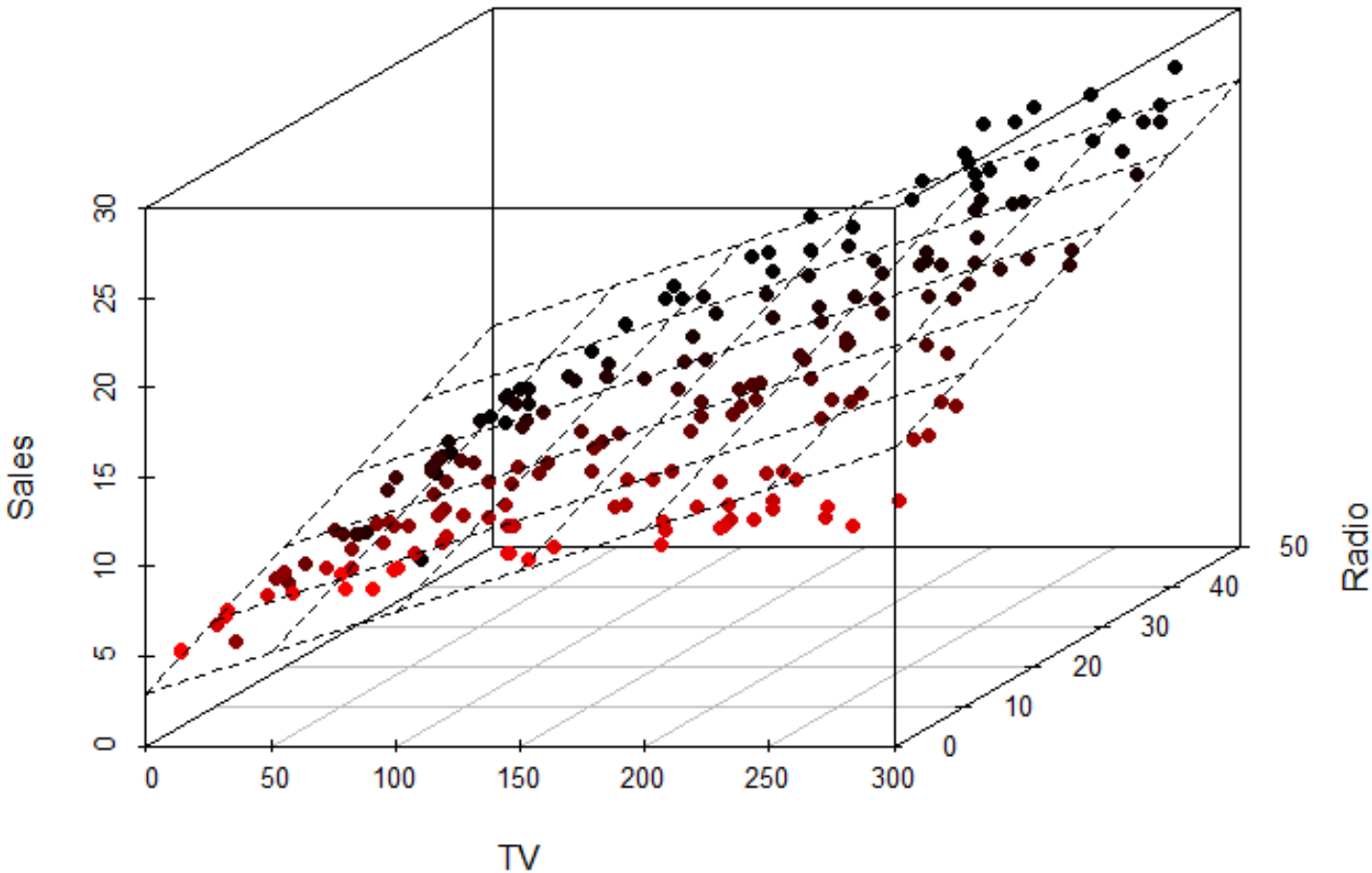
onde

  - $y \in \mathbb{R}$ é o valor-alvo do qual $\hat{y} \in \mathbb{R}$ é uma predição
  - $\mathbf{x} = \begin{bmatrix} 1 & x_1 & \cdots & x_n \end{bmatrix}^T \in \mathbb{R}^{n+1}$ é o vetor de atributos
  - $\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_n \end{bmatrix}^T$ é o vetor de parâmetros

- Conjunto de treinamento $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ organizado em uma matriz de projeto e um vetor de rótulos

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

Exemplo


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

Regressão Linear com Funções de Base

- De maneira geral, os atributos x_i podem ser escolhidos como uma transformação não-linear de uma ou mais variáveis

$$x_i = \varphi_i(x), \qquad \text{ou} \qquad x_i = \varphi_i(u_1, \ldots, u_N)$$

  onde $\varphi_i(\cdot)$ são chamadas de funções de base

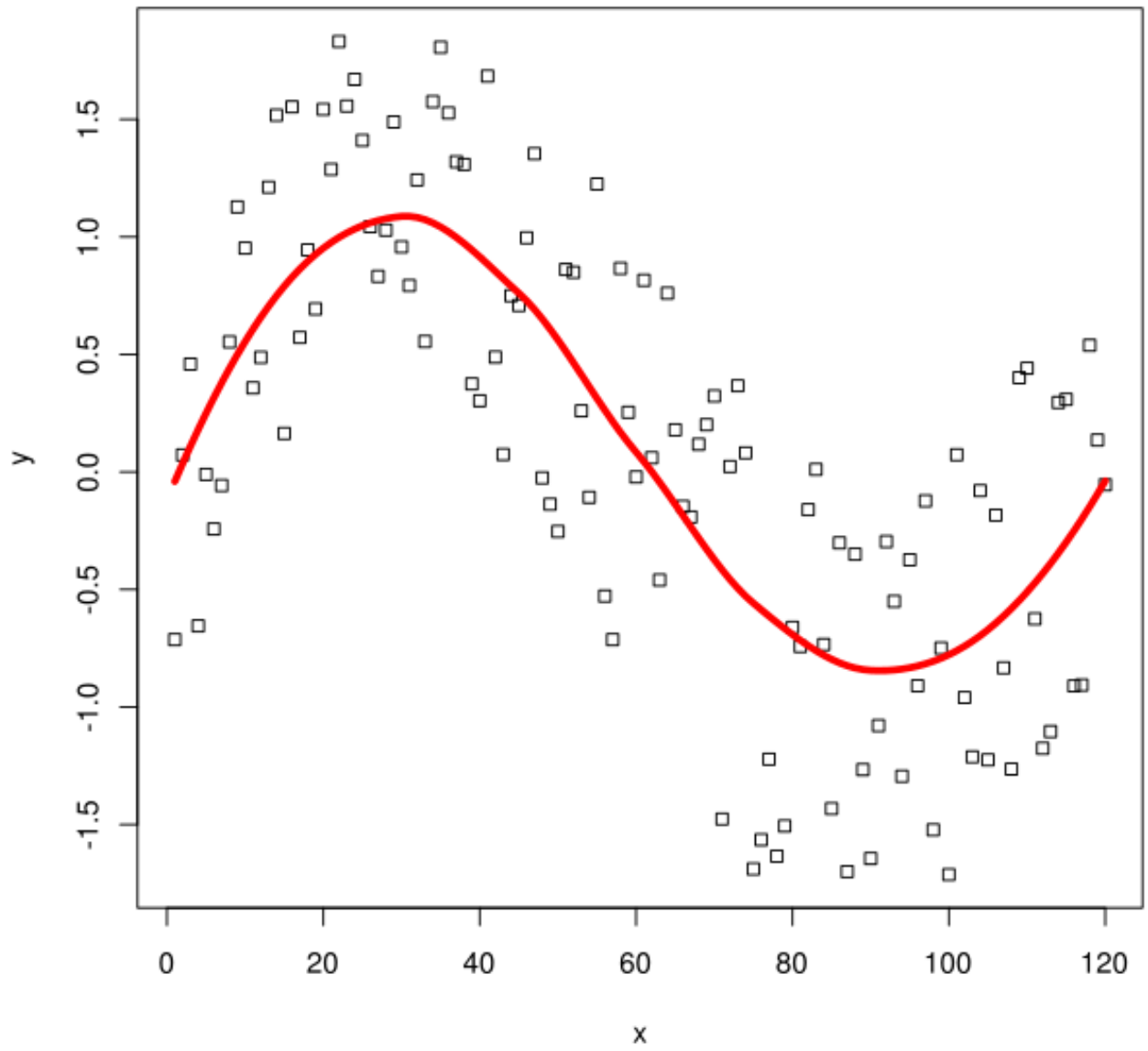
- Um exemplo é regressão polinomial de ordem n:

$$\hat{y} = w_0 + w_1 x + w_2 x^2 + \cdots + w_n x^n$$

  onde $\varphi_i(x) = x^i$.

- Nesse caso, embora o modelo continue linear em relação aos atributos x_i (e também em relação aos parâmetros w_i), um ajuste mais flexível pode ser feito com relação à variável original x

Exemplo



Treinamento

- Método dos mínimos quadrados (*ordinary least squares*): assume como função custo o erro quadrático médio

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

- Otimização via equação normal:

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

- Otimização via método do gradiente:

$$\mathbf{w}^{[t+1]} = \mathbf{w}^{[t]} - \alpha\frac{1}{m}\mathbf{X}^T(\mathbf{X}\mathbf{w}^{[t]} - \mathbf{y})$$

Desafios

- Equação normal:
  - Requer $m \geq n$
  - $\mathbf{X}^T\mathbf{X}$ não-inversível se houver atributos redundantes ($\star$)
  - Alta complexidade computacional se n é grande ($> 10^4$)
- Método do gradiente:
  - Requer escolha de α
  - Dificuldade de convergir se $\mathbf{X}^T\mathbf{X}$ for mal condicionada
    - Solução: normalização de atributos
  - Alta complexidade se m for muito grande
    - Solução: método do gradiente estocástico
- Ambos os métodos:
  - Aumento de n pode causar overfitting
    - Solução: regularização

Normalização de Atributos

Convergência do Método do Gradiente

- Se a matriz hessiana $\nabla^2 J(\mathbf{w}) = \frac{1}{m}\mathbf{X}^T\mathbf{X}$ for mal condicionada (isto é, com valor elevado da razão entre os autovalores máximo e mínimo), então o método do gradiente apresentará dificuldades de convergir (comportamento em "zig-zag")

- Exemplo:

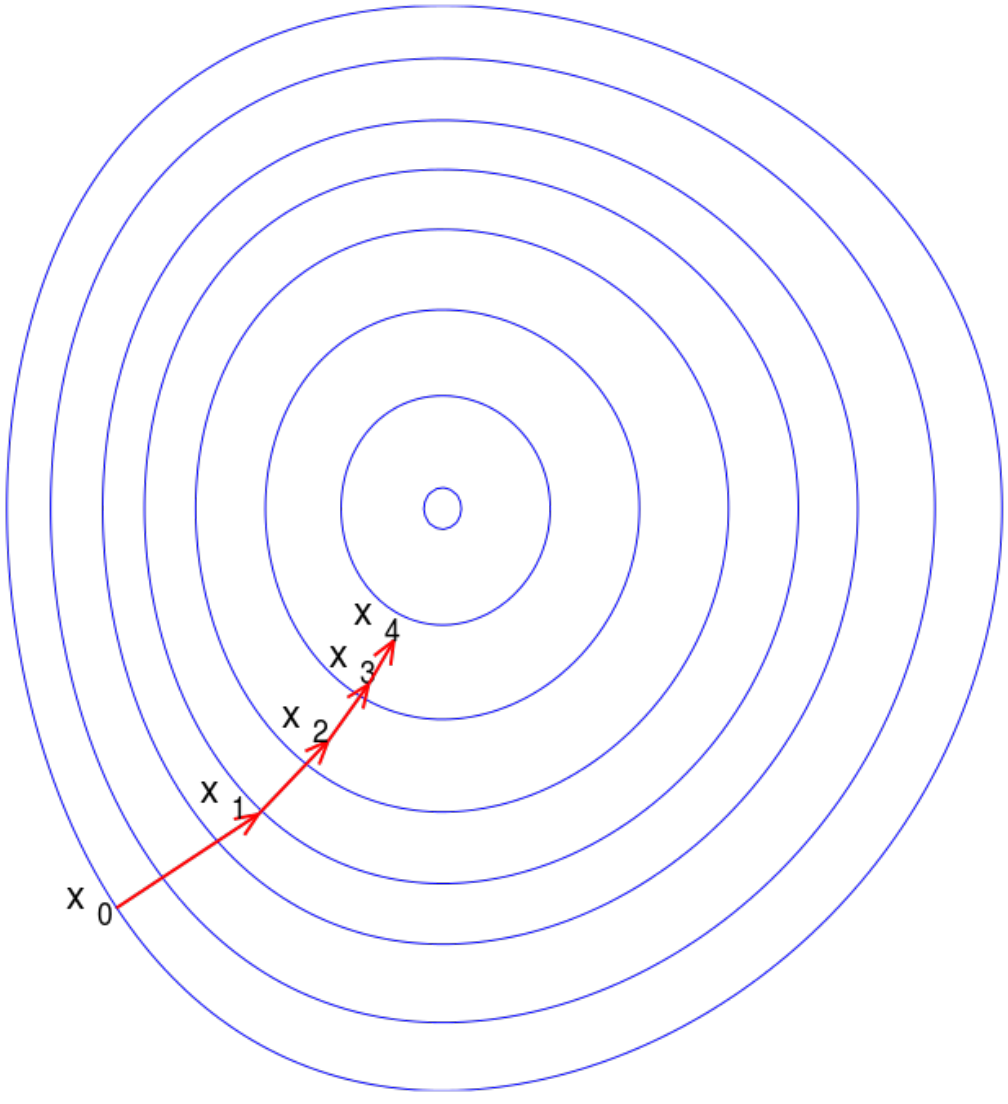
$$J(\mathbf{w}) = \lambda_0 w_0^2 + \lambda_1 w_1^2$$

$$\nabla J(\mathbf{w}) = \begin{bmatrix} \lambda_0 w_0 \\ \lambda_1 w_1 \end{bmatrix}, \quad \nabla^2 J(\mathbf{w}) = \begin{bmatrix} \lambda_0 & \\ & \lambda_1 \end{bmatrix}$$

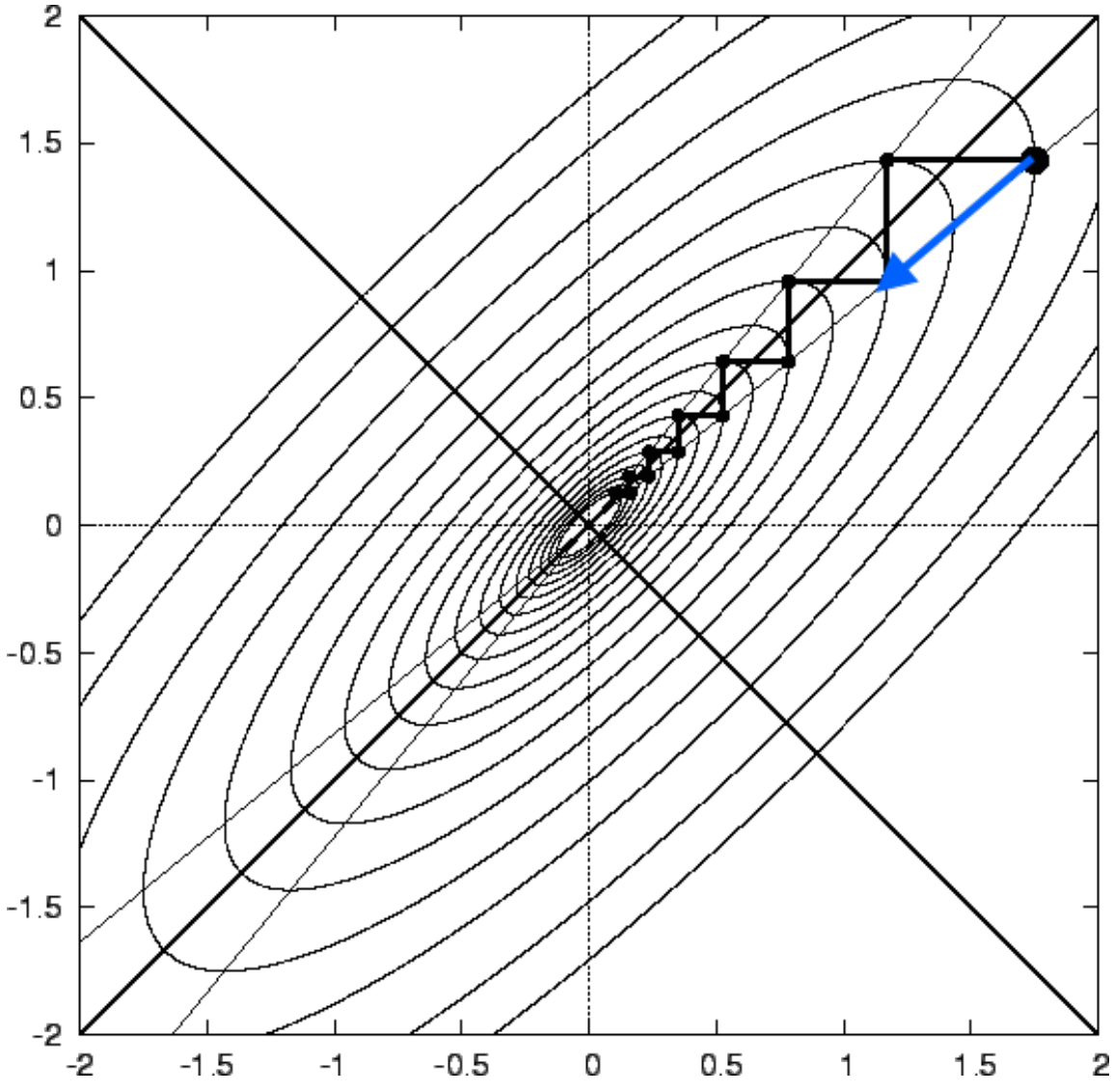
$$\mathbf{w}^{[t+1]} = \mathbf{w}^{[t]} - \alpha \begin{bmatrix} \lambda_0 w_0^{[t]} \\ \lambda_1 w_1^{[t]} \end{bmatrix}$$

  - Se $|\lambda_0/\lambda_1| \gg 1$ ou $|\lambda_1/\lambda_0| \gg 1$, então não existe uma taxa de aprendizado igualmente boa para os dois parâmetros

Exemplo (bem condicionado)



Exemplo (mal condicionado)



Normalização de Atributos

- O melhor condicionamento ocorre quando $\frac{1}{m}\mathbf{X}^T\mathbf{X} \propto \mathbf{I}$
- Uma solução é normalizar todos os atributos (exceto $x_0 = 1$) para que tenham média nula e variância unitária:

$$x'_j = \frac{x_j - \bar{x}_j}{\sigma_{x_j}}$$

onde $\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$ e $\sigma_{x_j}^2 = \frac{1}{m}\sum_{i=1}^{m}(x_j^{(i)} - \bar{x}_j)^2$

- Exemplo:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & \vdots \\ 1 & x_1^{(m)} \end{bmatrix} \implies \frac{1}{m}\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & \bar{x}_1 \\ \bar{x}_1 & \sigma_{x_1}^2 + \bar{x}_1^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

se $\bar{x}_1 = 0$ e $\sigma_{x_1} = 1$

Normalização de Atributos

- A normalização de atributos resulta no modelo linear:

$$\hat{y} = g(\mathbf{x}) = \mathbf{w}^T \mathbf{x}' = w_0 + w_1 \left(\frac{x_1 - \bar{x}_1}{\sigma_{x_1}} \right) + \cdots + w_n \left(\frac{x_n - \bar{x}_n}{\sigma_{x_n}} \right)$$

- Os parâmetros $\bar{x}_j$ e σ_{x_j} devem ser estimados exclusivamente a partir do conjunto de treinamento e guardados para serem usados na predição

- Alternativamente, o modelo pode ser reexpresso como:

$$\hat{y} = g(\mathbf{x}) = \mathbf{w'}^T \mathbf{x}$$

onde $w'_j = w_j / \sigma_{x_j}$ e $w'_0 = w_0 - \sum_j w_j \bar{x}_j / \sigma_{x_j}$

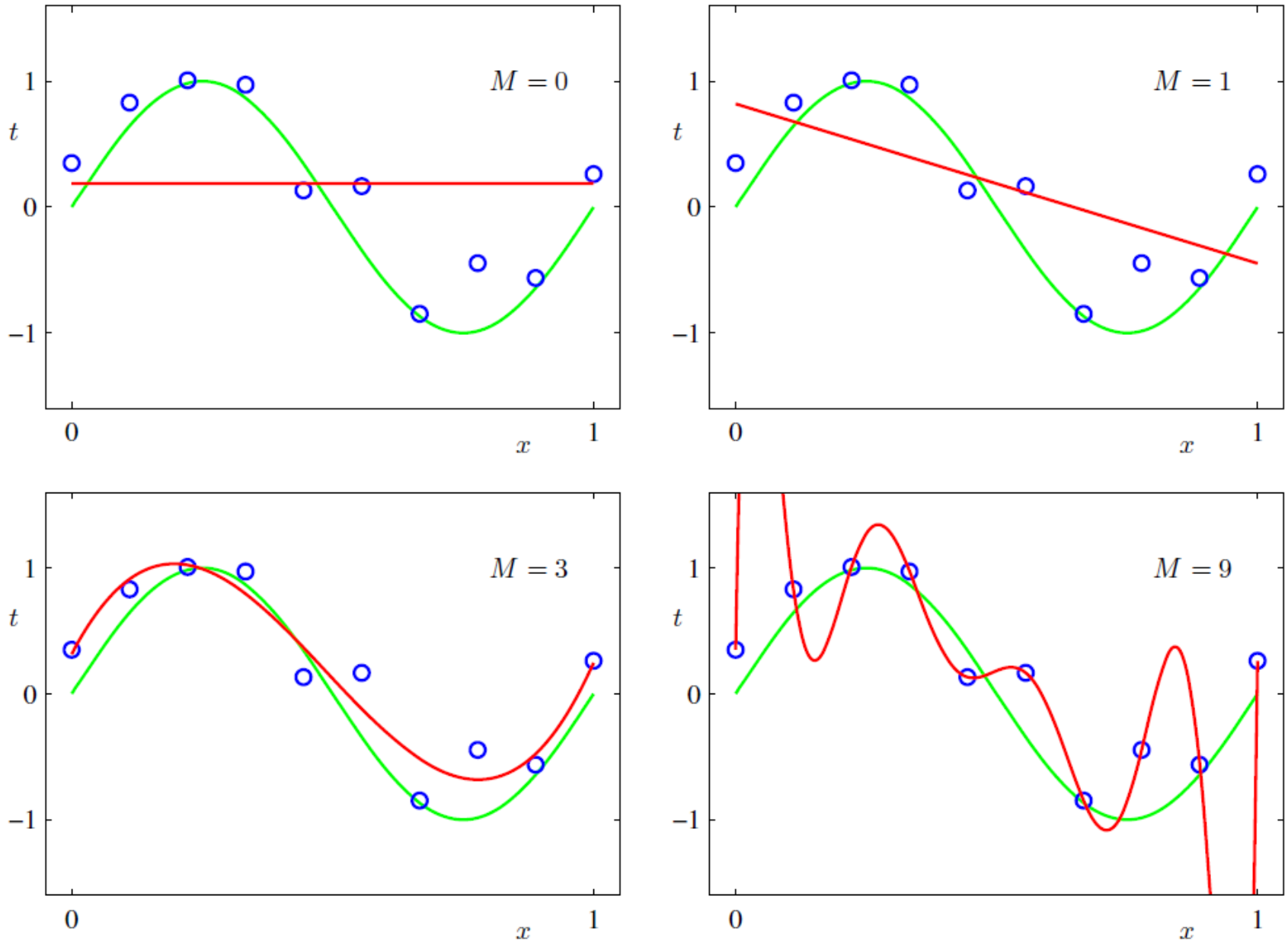
- Obs: normalização é essencial quando os atributos possuem faixas de valores bastante diferentes (ex: regressão polinomial)

Overfitting e Regularização

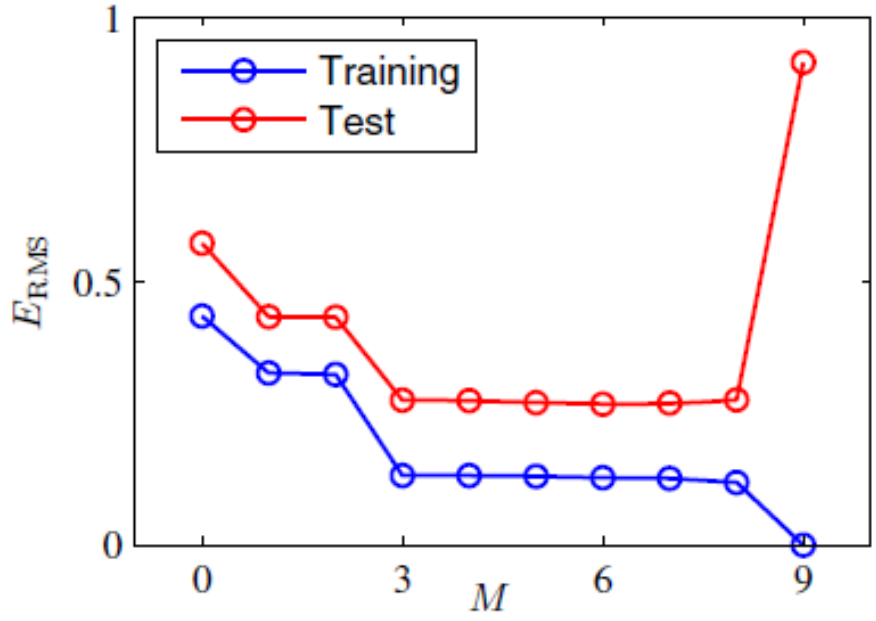
Overfitting

- 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

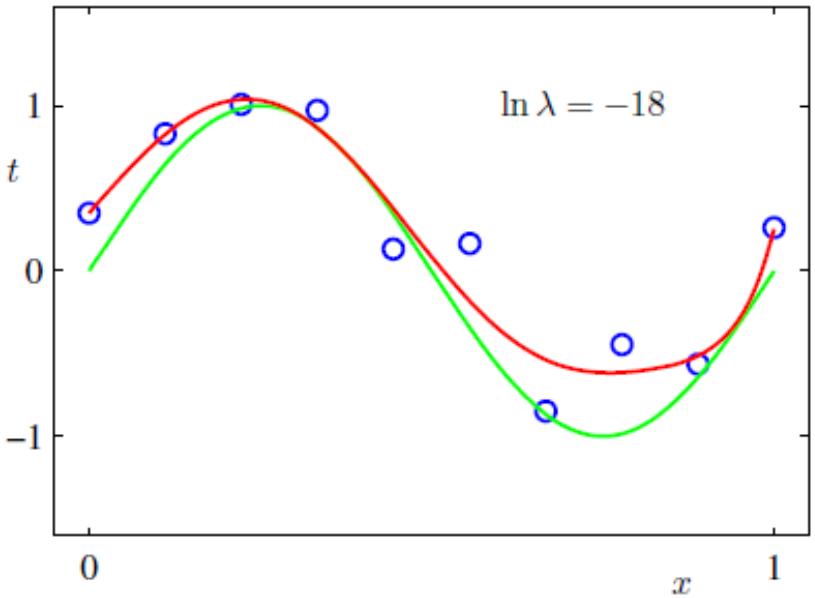
Regularização

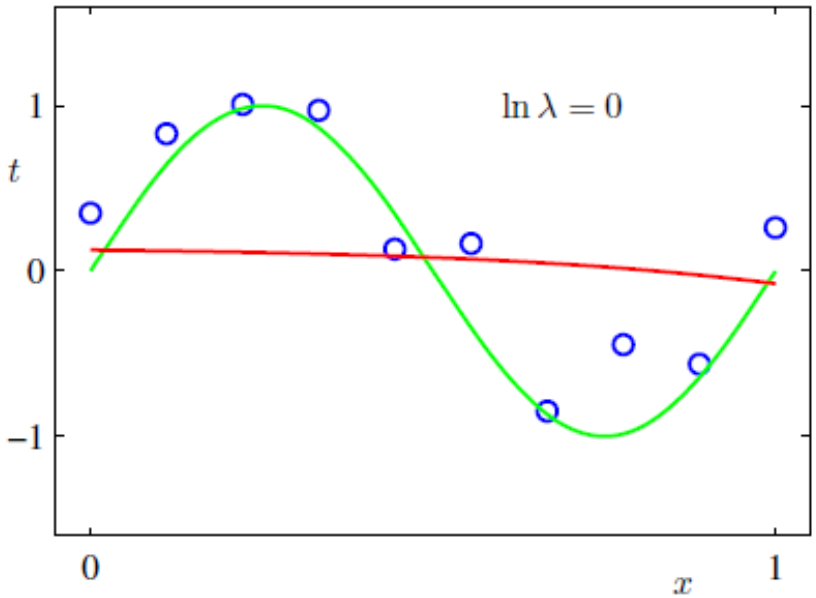
- A ideia é penalizar parâmetros que assumem valores muito elevados:

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

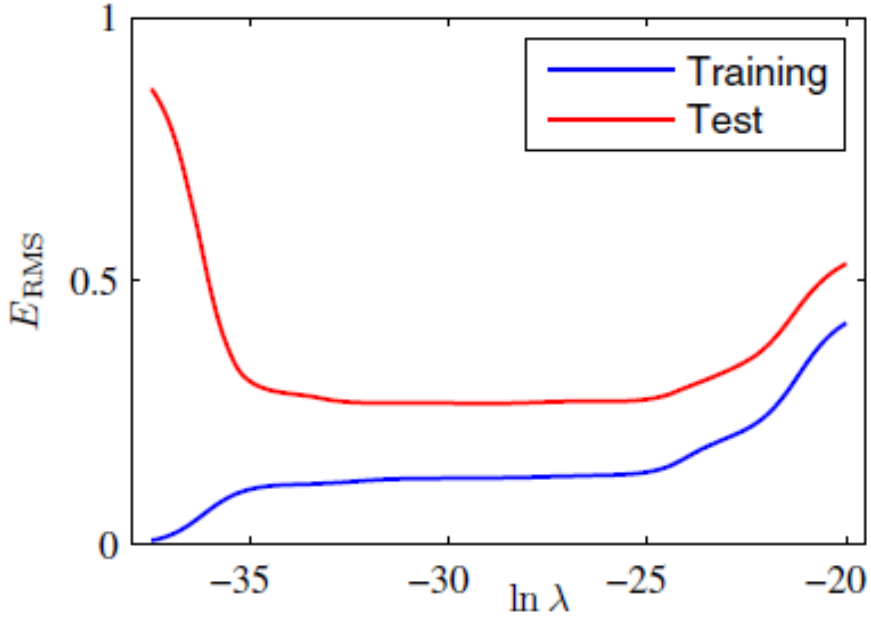
  onde
  - $J_{\text{train}}(\mathbf{w})$ é a função custo sobre o conjunto de treinamento
  - $\Omega(\mathbf{w})$ é a função de penalidade, chamada de regularizador
  - λ é o parâmetro de regularização (controla nossa preferência por parâmetros "menores", i.e., que sejam menos penalizados)
  - $J(\mathbf{w})$ é a função objetivo da otimização
- Exemplo: $\Omega(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\|\mathbf{w}\|^2$
  - Conhecida como regularização ℓ_2 ou regressão *ridge* ou *weight-decay*
- Navalha de Occam (*Occam's razor*): dentre todas as explicações consistentes com os dados, a mais simples é a mais plausível
  - Menor $\Omega(\mathbf{w})$ = "mais simples"

Exemplo





Exemplo



	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	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

- Função custo:

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

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

- Otimização via equação normal:

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

  - Obs: resolve o problema da invertibilidade

- Otimização via método do gradiente:

$$\mathbf{w}^{[t+1]} = \left(1 - \alpha\frac{\lambda}{m}\right)\mathbf{w}^{[t]} - \alpha\frac{1}{m}\mathbf{X}^T(\mathbf{X}\mathbf{w}^{[t]} - \mathbf{y})$$

Mínimos Quadrados com Regularização

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

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

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

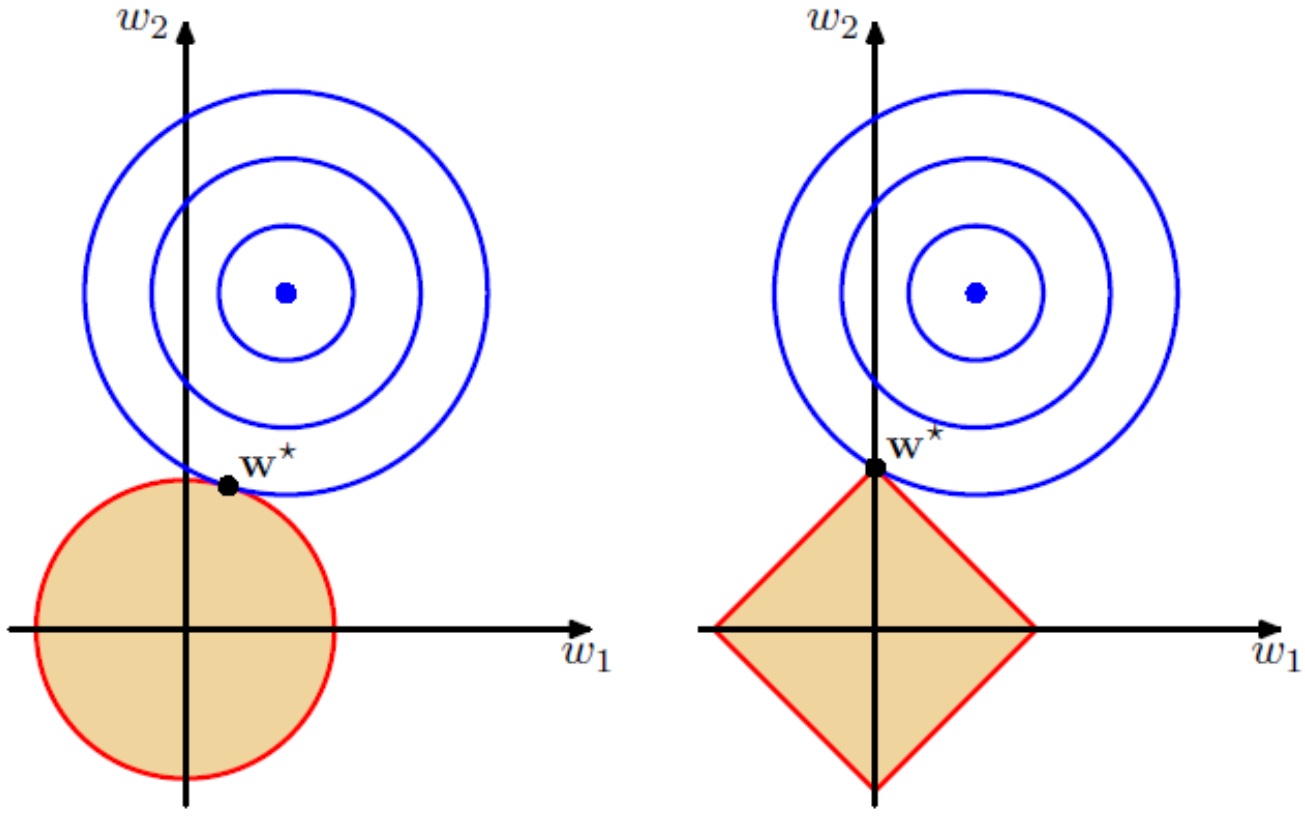
- Otimização via equação normal:

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

- Otimização via método do gradiente:

$$\mathbf{w}^{[t+1]} = \left(\mathbf{I} - \alpha\frac{\lambda}{m}\mathbf{L}\right)\mathbf{w}^{[t]} - \alpha\frac{1}{m}\mathbf{X}^T(\mathbf{X}\mathbf{w}^{[t]} - \mathbf{y})$$

Outros Tipos de Regularização



- Regularização ℓ_1 (*lasso regression*): $\Omega(\mathbf{w}) = \|\mathbf{w}\|$
  - Favorece modelos com pouco coeficientes não-nulos

Avaliação do Modelo

- A função custo regularizada $J(\mathbf{w})$ é usada como função objetivo do problema de otimização—portanto, deve ser usada para diagnosticar o comportamento do método do gradiente ao longo das iterações
  - Ex: Analisar convergência, escolher a taxa de aprendizado

- No entanto, o desempenho do modelo deve continuar sendo avaliado pela função custo sem regularização

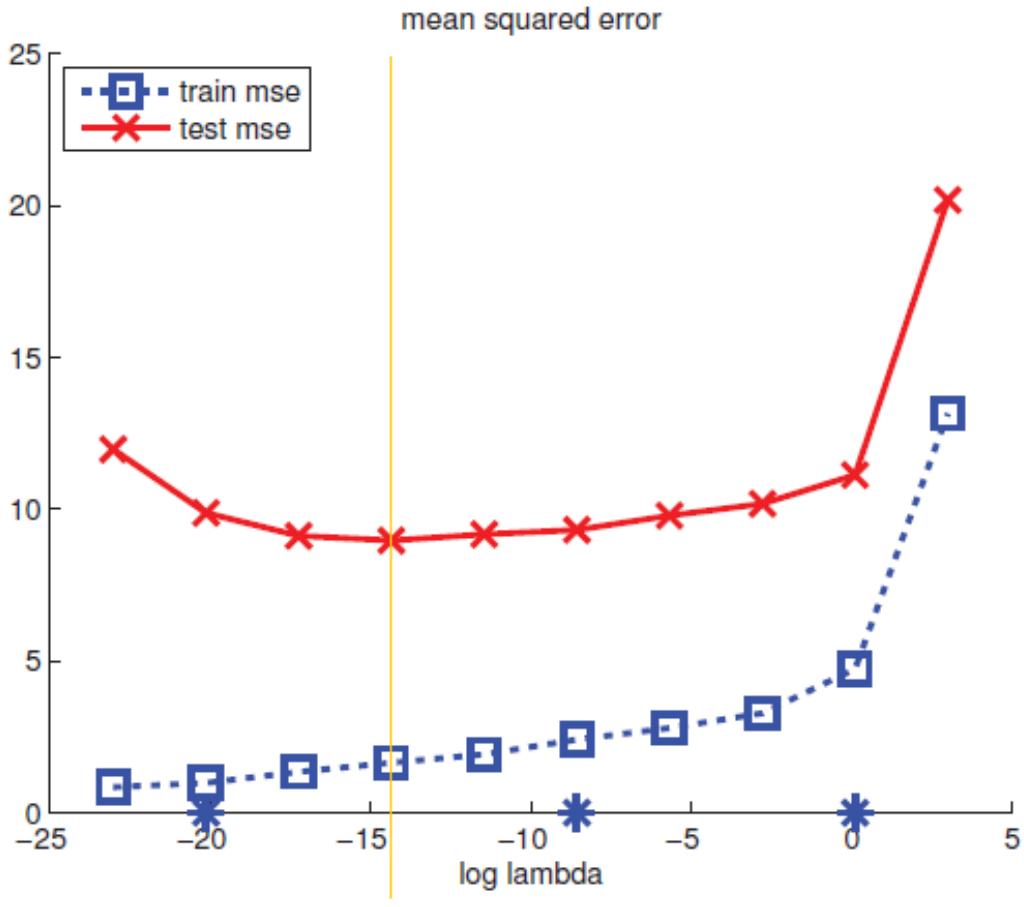
$$J_{\text{test}}(\mathbf{w}) = \frac{1}{2m}\|\mathbf{X}_{\text{test}}\mathbf{w} - \mathbf{y}_{\text{test}}\|^2$$
$$J_{\text{train}}(\mathbf{w}) = \frac{1}{2m}\|\mathbf{X}_{\text{train}}\mathbf{w} - \mathbf{y}_{\text{train}}\|^2$$

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

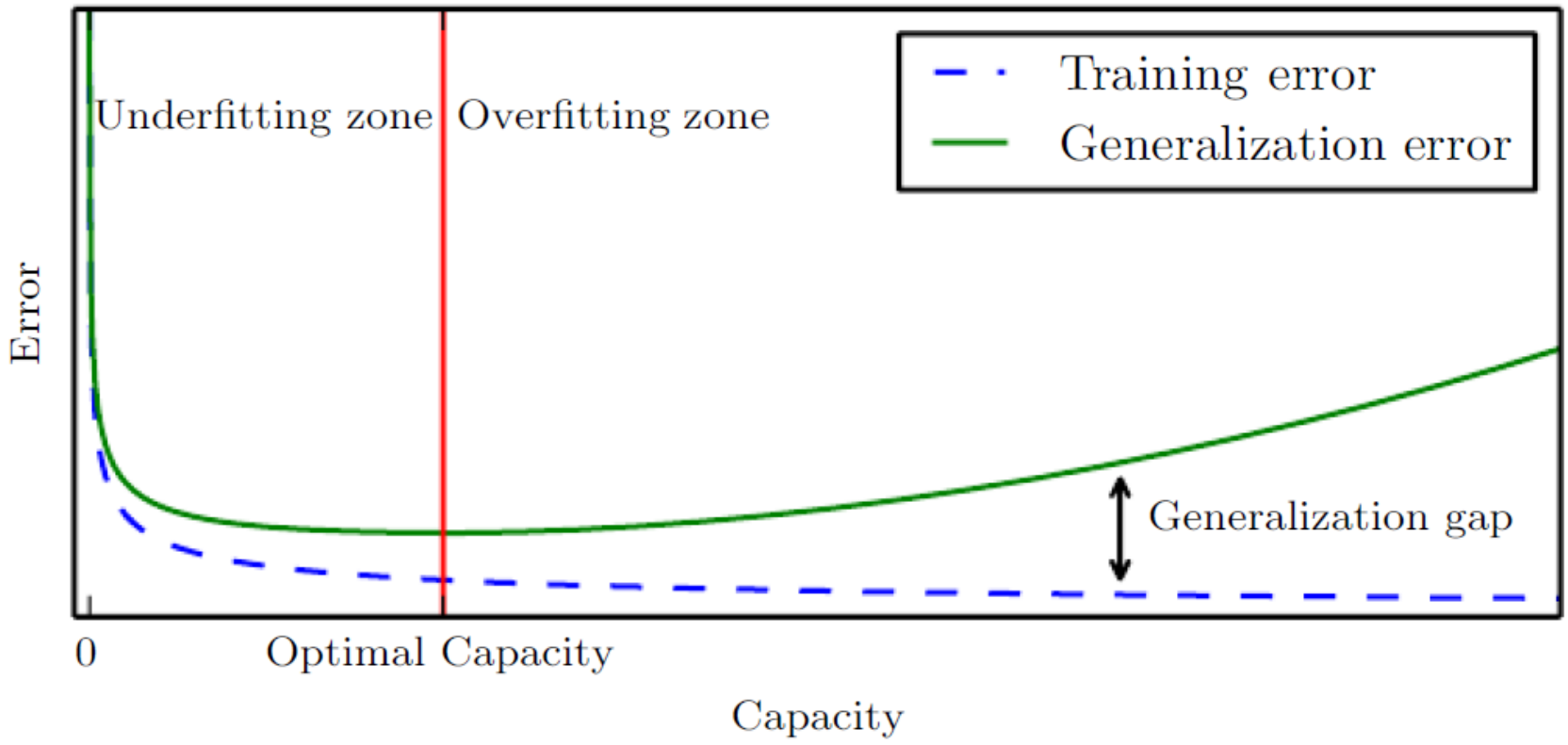
Escolha de Hiperparâmetros

- 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 projeto (por quê?)

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

Escolha de Hiperparâmetros

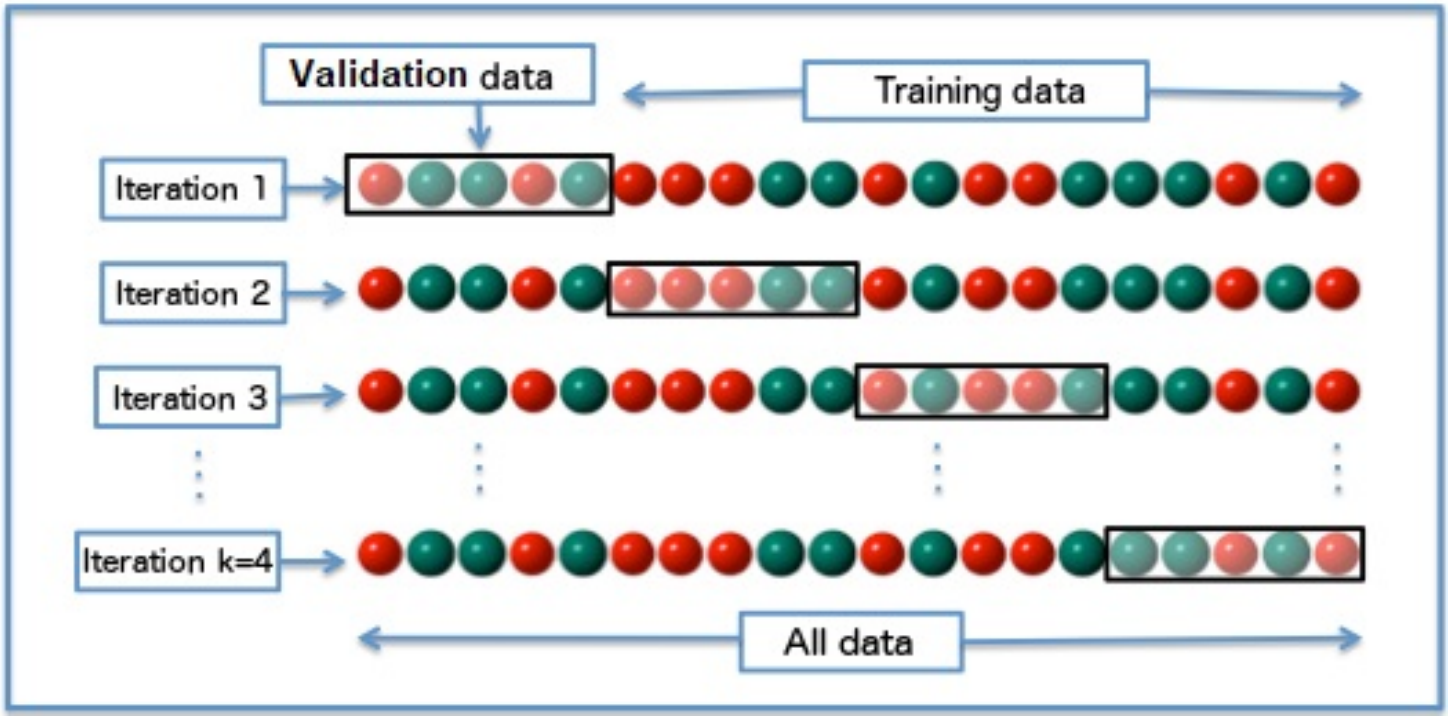
- A escolha de hiperparâmetros deve ser feita usando um conjunto separado (diferente dos conjuntos de treinamento e teste), chamado de conjunto de validação (ou conjunto de desenvolvimento)

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

  - Exemplo: 60% treinamento, 20% validação, 20% teste

- Validação cruzada (*cross-validation*): consiste em calcular a média de J_{val} ao longo de k rodadas de particionamento dos dados disponíveis em conjuntos de treinamento e validação
  - Evita "desperdiçar" amostras de treinamento, mas é mais complexo que utilizar $k = 1$ (validação convencional com conjuntos fixos)
  - Método k-fold: os k conjuntos de validação usados nas k rodadas formam uma partição dos dados disponíveis
  - Método de subamostragem aleatória: cada conjunto de validação é escolhido aleatoriamente a cada rodada

Validação Cruzada: Método k-fold



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

Método do Gradiente Estocástico

Método do Gradiente Estocástico

- Cada iteração do método do gradiente requer o cálculo de $\nabla J(\mathbf{w})$:

$$\mathbf{w}^{[t+1]} = \mathbf{w}^{[t]} - \alpha \nabla J(\mathbf{w}^{[t]})$$

  o qual por sua vez depende de todo o conjunto de treinamento:

$$J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} J_i(\mathbf{w}), \qquad \nabla J(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \nabla J_i(\mathbf{w})$$

- Se m é muito grande, o número total de operações até a convergência pode ser muito elevado
- Uma solução é aproximar $\nabla J(\mathbf{w})$ por $\nabla J_i(\mathbf{w})$, realizando a atualização dos pesos a cada novo exemplo de treinamento:

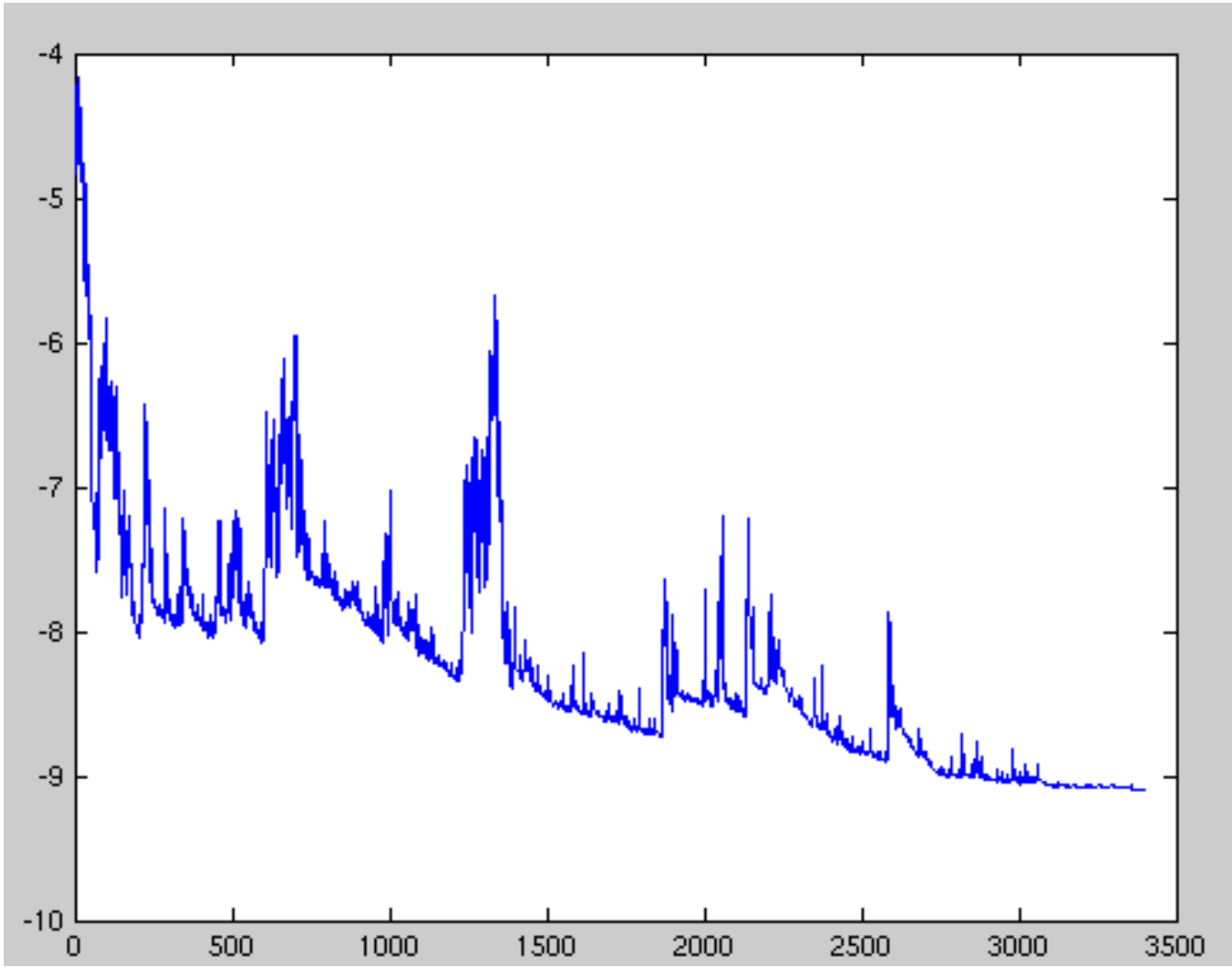
$$\mathbf{w}^{[i+1]} = \mathbf{w}^{[i]} - \alpha \nabla J_i(\mathbf{w}^{[i]})$$

  - Chamado de método do gradiente estocástico pois $J_i(\mathbf{w})$ é uma variável aleatória que depende de exemplo escolhido dentre $\mathcal{D}$

Método do Gradiente Estocástico

- Normalmente são necessárias múltiplas passagens por todo o conjunto de treinamento, chamadas de épocas (*epochs*)
  - Logo, m iterações são realizadas a cada época
  - É recomendável embaralhar o conjunto de treinamento a cada nova época
- Flutua significativamente, porém tem convergência garantida caso a taxa de aprendizagem seja reduzida apropriadamente a cada iteração

Exemplo



Método do Gradiente "*Mini-Batch*"

- Método do gradiente convencional (também chamado de "*batch*"): processa m exemplos a cada iteração
- Método do gradiente estocástico (também chamado de *on-line*): processa 1 exemplo a cada iteração
- Método do gradiente "*mini-batch*": processa b exemplos a cada iteração

$$\mathbf{w}^{[t+1]} = \mathbf{w}^{[t]} - \alpha \frac{1}{b} \sum_{i=\ell b+1}^{\ell b+b} \nabla J_i(\mathbf{w}^{[t]}), \quad \ell = 0, 1, 2, \ldots, m/b - 1$$

  - Aumento de b reduz as flutuações
  - Permite uso de bibliotecas vetorizadas
  - Menor complexidade que o método convencional

Observações dos Alunos

Observações

- Talvez cause menos confusão usar uma terminologia que faça distinção entre a função objetivo do treinamento e a função que fornece o erro médio do modelo sobre um conjunto de dados—isto é, não usar “função custo” para ambos