Aula 11 - Seleção e Avaliação de Modelos I

João Florindo

Instituto de Matemática, Estatística e Computação Científica
Universidade Estadual de Campinas - Brasil
florindo@unicamp.br

Outline

Regressão linear para o preço de imóveis:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right].$$

E se o erro em novos dados for muito grande?

Controlar a **capacidade** do modelo:

- Obter mais exemplos de treinamento (complicado!)
- Usar menos atributos ou obter mais atributos
- Adicionar atributos polinomiais ($x_1^2, x_2^2, x_1 x_2$,etc.)
- Aumentar ou diminuir λ

- Diagnóstico do algoritmo.
- Toma tempo, mas vale a pena!

Regressão linear para o preço de imóveis:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right].$$

E se o erro em novos dados for muito grande?

Controlar a **capacidade** do modelo:

- Obter mais exemplos de treinamento (complicado!)
- Usar menos atributos ou obter mais atributos
- Adicionar atributos polinomiais (x_1^2, x_2^2, x_1x_2,etc.)
- Aumentar ou diminuir λ

- Diagnóstico do algoritmo.
- Toma tempo, mas vale a pena!

Regressão linear para o preço de imóveis:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right].$$

E se o erro em novos dados for muito grande?

Controlar a **capacidade** do modelo:

- Obter mais exemplos de treinamento (complicado!)
- Usar menos atributos ou obter mais atributos
- Adicionar atributos polinomiais ($x_1^2, x_2^2, x_1 x_2$,etc.)
- Aumentar ou diminuir λ

- Diagnóstico do algoritmo.
- Toma tempo, mas vale a pena!

Regressão linear para o preço de imóveis:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right].$$

E se o erro em novos dados for muito grande?

Controlar a **capacidade** do modelo:

- Obter mais exemplos de treinamento (complicado!)
- Usar menos atributos ou obter mais atributos
- Adicionar atributos polinomiais (x_1^2, x_2^2, x_1x_2,etc.)
- Aumentar ou diminuir λ

- Diagnóstico do algoritmo.
- Toma tempo, mas vale a pena!

Regressão linear para o preço de imóveis:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right].$$

E se o erro em novos dados for muito grande?

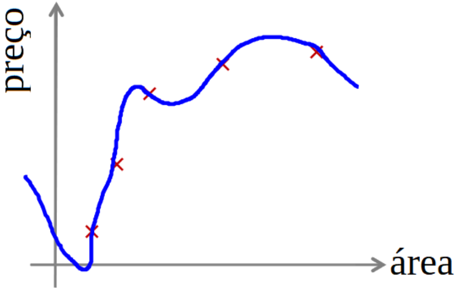
Controlar a **capacidade** do modelo:

- Obter mais exemplos de treinamento (complicado!)
- Usar menos atributos ou obter mais atributos
- Adicionar atributos polinomiais ($x_1^2, x_2^2, x_1 x_2$,etc.)
- Aumentar ou diminuir λ

- Diagnóstico do algoritmo.
- Toma tempo, mas vale a pena!

Outline

Overfitting:



$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

- Mas e se temos centenas/milhares de atributos?
- Como medir a generalização do modelo?
- Avaliar (validar) no próprio treino (**validação de ressubstituição**) gera viés otimista.
- *Holdout*: dividir aleatoriamente em 2 subconjuntos independentes (treino e teste). 70% para treino e 30% para teste é usual (usa-se até 90/10 para conjuntos muito grandes).

- Mas e se temos centenas/milhares de atributos?
- Como medir a generalização do modelo?
- Avaliar (validar) no próprio treino (**validação de ressubstituição**) gera viés otimista.
- *Holdout*: dividir aleatoriamente em 2 subconjuntos independentes (treino e teste). 70% para treino e 30% para teste é usual (usa-se até 90/10 para conjuntos muito grandes).

- Mas e se temos centenas/milhares de atributos?
- Como medir a generalização do modelo?
- Avaliar (validar) no próprio treino (**validação de ressubstituição**) gera viés otimista.
- *Holdout*: dividir aleatoriamente em 2 subconjuntos independentes (treino e teste). 70% para treino e 30% para teste é usual (usa-se até 90/10 para conjuntos muito grandes).

- Mas e se temos centenas/milhares de atributos?
- Como medir a generalização do modelo?
- Avaliar (validar) no próprio treino (**validação de ressubstituição**) gera viés otimista.
- *Holdout*: dividir aleatoriamente em 2 subconjuntos independentes (treino e teste). 70% para treino e 30% para teste é usual (usa-se até 90/10 para conjuntos muito grandes).

Holdout

Área	Preço
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

Conjunto de treino

$$\boxed{\begin{array}{c}(x^{(1)}, y^{(1)})\\(x^{(2)}, y^{(2)})\\\vdots\\(x^{(m)}, y^{(m)})\end{array}}$$

Conjunto de teste

$$\boxed{\begin{array}{c}(x_{test}^{(1)}, y_{test}^{(1)})\\(x_{test}^{(2)}, y_{test}^{(2)})\\\vdots\\(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})\end{array}}$$

Procedimento geral

REGRESSÃO LINEAR:

- Aprender θ minimizando $J(\theta)$ usando os dados do conjunto de treino
- Calcular o erro de teste:

$$J_{teste}(\theta) = \frac{1}{2m_{teste}} \sum_{i=1}^{m_{teste}} (h_\theta(x_{teste}^{(i)}) - y_{teste}^{(i)})^2.$$

Procedimento geral

REGRESSÃO LINEAR:

- Aprender θ minimizando $J(\theta)$ usando os dados do conjunto de treino
- Calcular o erro de teste:

$$J_{teste}(\theta) = \frac{1}{2m_{teste}} \sum_{i=1}^{m_{teste}} (h_\theta(x_{teste}^{(i)}) - y_{teste}^{(i)})^2.$$

Procedimento geral

REGRESSÃO LOGÍSTICA:

- Aprender o parâmetro θ a partir do treinamento
- Calcular o erro de teste:

$$J_{teste}(\theta) = -\frac{1}{m_{teste}} \sum_{i=1}^{m_{teste}} y_{teste}^{(i)} \log h_\theta(x_{teste}^{(i)}) + (1 - y_{teste}^{(i)}) \log h_\theta(x_{teste}^{(i)}).$$

- Uma alternativa popular é o erro de classificação (erro 0/1):

$$J_{teste}(\theta) = \frac{1}{m_{teste}} \sum_{i=1}^{m_{teste}} err(h_\theta(x_{teste}^{(i)}), y_{teste}^{(i)}),$$

em que

$$err(h_{teste}(x), y) = \begin{cases} 1 & \text{se } h_\theta(x) \geq 0.5 \text{ e } y = 0 \\ & \text{ou } h_\theta(x) < 0.5 \text{ e } y = 1 \\ 0 & \text{caso contrário.} \end{cases}$$

Procedimento geral

REGRESSÃO LOGÍSTICA:

- Aprender o parâmetro θ a partir do treinamento
- Calcular o erro de teste:

$$J_{teste}(\theta) = -\frac{1}{m_{teste}} \sum_{i=1}^{m_{teste}} y_{teste}^{(i)} \log h_\theta(x_{teste}^{(i)}) + (1 - y_{teste}^{(i)}) \log h_\theta(x_{teste}^{(i)}).$$

- Uma alternativa popular é o erro de classificação (erro 0/1):

$$J_{teste}(\theta) = \frac{1}{m_{teste}} \sum_{i=1}^{m_{teste}} err(h_\theta(x_{teste}^{(i)}), y_{teste}^{(i)}),$$

em que

$$err(h_{teste}(x), y) = \begin{cases} 1 & \text{se } h_\theta(x) \geq 0.5 \text{ e } y = 0 \\ & \text{ou } h_\theta(x) < 0.5 \text{ e } y = 1 \\ 0 & \text{caso contrário.} \end{cases}$$

Procedimento geral

REGRESSÃO LOGÍSTICA:

- Aprender o parâmetro θ a partir do treinamento
- Calcular o erro de teste:

$$J_{teste}(\theta) = -\frac{1}{m_{teste}} \sum_{i=1}^{m_{teste}} y_{teste}^{(i)} \log h_\theta(x_{teste}^{(i)}) + (1 - y_{teste}^{(i)}) \log h_\theta(x_{teste}^{(i)}).$$

- Uma alternativa popular é o erro de classificação (erro 0/1):

$$J_{teste}(\theta) = \frac{1}{m_{teste}} \sum_{i=1}^{m_{teste}} err(h_\theta(x_{teste}^{(i)}), y_{teste}^{(i)}),$$

em que

$$err(h_{teste}(x), y) = \begin{cases} 1 & \text{se } h_\theta(x) \geq 0.5 \text{ e } y = 0 \\ & \text{ou } h_\theta(x) < 0.5 \text{ e } y = 1 \\ 0 & \text{caso contrário.} \end{cases}$$

Notas

- **Estratificação**: os conjuntos de treino e teste devem preservar a proporção de amostras por classe do conjunto original (crítico especialmente em conjuntos pequenos e desbalanceados).
- *Holdout* pode ser repetido k vezes, medindo-se o erro médio.
- O conjunto de teste é reincorporado ao treinamento para o modelo final (aplicado no mundo real).

Notas

- **Estratificação**: os conjuntos de treino e teste devem preservar a proporção de amostras por classe do conjunto original (crítico especialmente em conjuntos pequenos e desbalanceados).
- *Holdout* pode ser repetido k vezes, medindo-se o erro médio.
- O conjunto de teste é reincorporado ao treinamento para o modelo final (aplicado no mundo real).

Notas

- **Estratificação**: os conjuntos de treino e teste devem preservar a proporção de amostras por classe do conjunto original (crítico especialmente em conjuntos pequenos e desbalanceados).
- *Holdout* pode ser repetido k vezes, medindo-se o erro médio.
- O conjunto de teste é reincorporado ao treinamento para o modelo final (aplicado no mundo real).

Outline

- **Seleção de Modelo**: Qual o grau ideal para o polinômio de atributos ou qual melhor λ na regularização? Estes são **hiperparâmetros** do modelo.
- Erro de teste é otimista demais para este propósito.
- SOLUÇÃO: Separar um terceiro conjunto de **validação** (ou **validação cruzada**).
- 60/20/20 é uma razão usual para treino/validação/teste.

- **Seleção de Modelo**: Qual o grau ideal para o polinômio de atributos ou qual melhor λ na regularização? Estes são **hiperparâmetros** do modelo.
- Erro de teste é otimista demais para este propósito.
- SOLUÇÃO: Separar um terceiro conjunto de **validação** (ou **validação cruzada**).
- 60/20/20 é uma razão usual para treino/validação/teste.

- **Seleção de Modelo**: Qual o grau ideal para o polinômio de atributos ou qual melhor λ na regularização? Estes são **hiperparâmetros** do modelo.
- Erro de teste é otimista demais para este propósito.
- SOLUÇÃO: Separar um terceiro conjunto de **validação** (ou **validação cruzada**).
- 60/20/20 é uma razão usual para treino/validação/teste.

- **Seleção de Modelo**: Qual o grau ideal para o polinômio de atributos ou qual melhor λ na regularização? Estes são **hiperparâmetros** do modelo.
- Erro de teste é otimista demais para este propósito.
- SOLUÇÃO: Separar um terceiro conjunto de **validação** (ou **validação cruzada**).
- 60/20/20 é uma razão usual para treino/validação/teste.

Área	Preço
2104	400
1600	330
2400	369
1416	232
3000	540
1985	300
1534	315
1427	199
1380	212
1494	243

Treino

$$
\boxed{\begin{array}{c}
(x^{(1)}, y^{(1)}) \\
(x^{(2)}, y^{(2)}) \\
\vdots \\
(x^{(m)}, y^{(m)})
\end{array}}
$$

Validação Cruzada (cv)

$$
\boxed{\begin{array}{c}
(x_{cv}^{(1)}, y_{cv}^{(1)}) \\
(x_{cv}^{(2)}, y_{cv}^{(2)}) \\
\vdots \\
(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})
\end{array}}
$$

Teste

$$
\boxed{\begin{array}{c}
(x_{test}^{(1)}, y_{test}^{(1)}) \\
(x_{test}^{(2)}, y_{test}^{(2)}) \\
\vdots \\
(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})
\end{array}}
$$

- Erro de treino:

$$J_{treino}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- Erro de validação:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

- Erro de teste:

$$J_{teste}(\theta) = \frac{1}{2m_{teste}}\sum_{i=1}^{m_{teste}}(h_\theta(x_{teste}^{(i)}) - y_{teste}^{(i)})^2$$

- Erro de treino:

$$J_{treino}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- Erro de validação:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

- Erro de teste:

$$J_{teste}(\theta) = \frac{1}{2m_{teste}}\sum_{i=1}^{m_{teste}}(h_\theta(x_{teste}^{(i)}) - y_{teste}^{(i)})^2$$

- Erro de treino:

$$J_{treino}(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

- Erro de validação:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}}\sum_{i=1}^{m_{cv}}(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

- Erro de teste:

$$J_{teste}(\theta) = \frac{1}{2m_{teste}}\sum_{i=1}^{m_{teste}}(h_\theta(x_{teste}^{(i)}) - y_{teste}^{(i)})^2$$

Procedimento Geral (grau do polinômio)

1. Otimizar parâmetros Θ no conjunto de treino para cada grau.
2. Escolhemos então o grau d que propicia o menor erro de validação.
3. Retreinamos o modelo para o grau d agora reincluindo os dados de validação no treino.
4. Estimar o erro de generalização do modelo escolhido usando $J_{teste}(\Theta)$.
5. Retreinamos usando todos os dados disponíveis (treino, validação e teste) para o modelo final.

Procedimento Geral (grau do polinômio)

1. Otimizar parâmetros Θ no conjunto de treino para cada grau.
2. Escolhemos então o grau d que propicia o menor erro de validação.
3. Retreinamos o modelo para o grau d agora reincluindo os dados de validação no treino.
4. Estimar o erro de generalização do modelo escolhido usando $J_{teste}(\Theta)$.
5. Retreinamos usando todos os dados disponíveis (treino, validação e teste) para o modelo final.

Procedimento Geral (grau do polinômio)

1. Otimizar parâmetros Θ no conjunto de treino para cada grau.
2. Escolhemos então o grau d que propicia o menor erro de validação.
3. Retreinamos o modelo para o grau d agora reincluindo os dados de validação no treino.
4. Estimar o erro de generalização do modelo escolhido usando $J_{teste}(\Theta)$.
5. Retreinamos usando todos os dados disponíveis (treino, validação e teste) para o modelo final.

Procedimento Geral (grau do polinômio)

1. Otimizar parâmetros Θ no conjunto de treino para cada grau.
2. Escolhemos então o grau d que propicia o menor erro de validação.
3. Retreinamos o modelo para o grau d agora reincluindo os dados de validação no treino.
4. Estimar o erro de generalização do modelo escolhido usando $J_{teste}(\Theta)$.
5. Retreinamos usando todos os dados disponíveis (treino, validação e teste) para o modelo final.

Procedimento Geral (grau do polinômio)

1. Otimizar parâmetros Θ no conjunto de treino para cada grau.
2. Escolhemos então o grau d que propicia o menor erro de validação.
3. Retreinamos o modelo para o grau d agora reincluindo os dados de validação no treino.
4. Estimar o erro de generalização do modelo escolhido usando $J_{teste}(\Theta)$.
5. Retreinamos usando todos os dados disponíveis (treino, validação e teste) para o modelo final.

K-Fold

- A separação de parte significativa dos exemplos para teste/validação gera um viés pessimista no modelo.
- Mais crítico em conjuntos pequenos.
- **K-Fold**: Divide o conjunto de dados em K partes. Usa 1 parte para teste/validação e as $K - 1$ para treino. Itera K vezes de modo a percorrer o conjunto todo.
- $K = 5$ e $K = 10$ são os mais frequentes.

K-Fold

- A separação de parte significativa dos exemplos para teste/validação gera um viés pessimista no modelo.
- Mais crítico em conjuntos pequenos.
- **K-Fold**: Divide o conjunto de dados em K partes. Usa 1 parte para teste/validação e as $K-1$ para treino. Itera K vezes de modo a percorrer o conjunto todo.
- $K = 5$ e $K = 10$ são os mais frequentes.

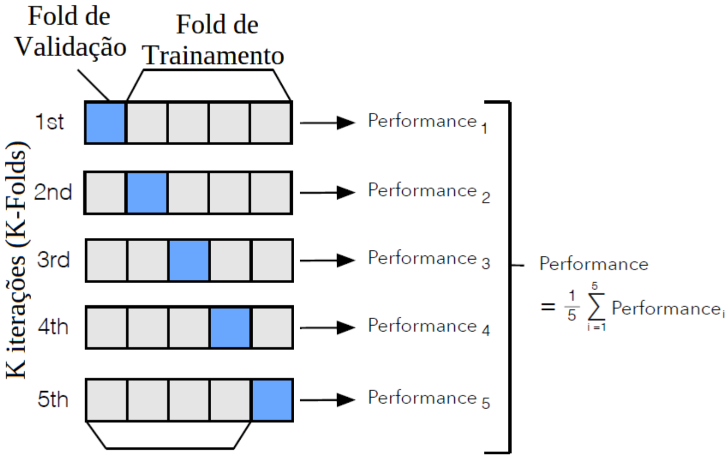
K-Fold

- A separação de parte significativa dos exemplos para teste/validação gera um viés pessimista no modelo.
- Mais crítico em conjuntos pequenos.
- **K-Fold**: Divide o conjunto de dados em K partes. Usa 1 parte para teste/validação e as $K-1$ para treino. Itera K vezes de modo a percorrer o conjunto todo.
- $K=5$ e $K=10$ são os mais frequentes.

K-Fold

- A separação de parte significativa dos exemplos para teste/validação gera um viés pessimista no modelo.
- Mais crítico em conjuntos pequenos.
- **K-Fold**: Divide o conjunto de dados em K partes. Usa 1 parte para teste/validação e as $K-1$ para treino. Itera K vezes de modo a percorrer o conjunto todo.
- $K = 5$ e $K = 10$ são os mais frequentes.

K-Fold


Fold de Validação
Fold de Trainamento
K iterações (K-Folds)
1st
2nd
3rd
4th
5th
Performance 1
Performance 2
Performance 3
Performance 4
Performance 5
Performance
$= \frac{1}{5} \sum_{i=1}^{5} \text{Performance}_i$

Leave-One-Out

- *Leave-One-Out*: Caso extremo de K-Fold (K=m).
- m iterações: em cada uma ajusta os parâmetros do modelo em $m-1$ exemplos e valida no exemplo restante.
- Reduz viés pessimista (preserva treino quase inteiro).
- Alta variância da estimativa de erro e caro computacionalmente!

SELEÇÃO DE MODELO

Tanto K-Fold quanto LOO podem ser usados para seleção de modelo (hiperparâmetros).

Leave-One-Out

- *Leave-One-Out*: Caso extremo de K-Fold (K=m).
- m iterações: em cada uma ajusta os parâmetros do modelo em $m - 1$ exemplos e valida no exemplo restante.
- Reduz viés pessimista (preserva treino quase inteiro).
- Alta variância da estimativa de erro e caro computacionalmente!

SELEÇÃO DE MODELO

Tanto K-Fold quanto LOO podem ser usados para seleção de modelo (hiperparâmetros).

Leave-One-Out

- *Leave-One-Out*: Caso extremo de K-Fold (K=m).
- m iterações: em cada uma ajusta os parâmetros do modelo em $m - 1$ exemplos e valida no exemplo restante.
- Reduz viés pessimista (preserva treino quase inteiro).
- Alta variância da estimativa de erro e caro computacionalmente!

SELEÇÃO DE MODELO

Tanto K-Fold quanto LOO podem ser usados para seleção de modelo (hiperparâmetros).

Leave-One-Out

- *Leave-One-Out*: Caso extremo de K-Fold (K=m).
- m iterações: em cada uma ajusta os parâmetros do modelo em $m - 1$ exemplos e valida no exemplo restante.
- Reduz viés pessimista (preserva treino quase inteiro).
- Alta variância da estimativa de erro e caro computacionalmente!

SELEÇÃO DE MODELO

Tanto K-Fold quanto LOO podem ser usados para seleção de modelo (hiperparâmetros).

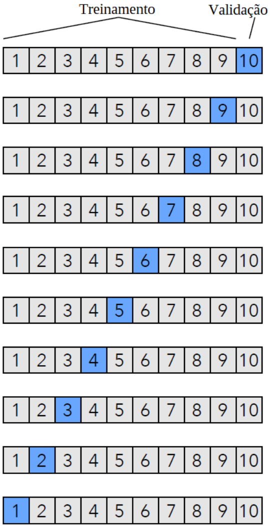
Leave-One-Out

- *Leave-One-Out*: Caso extremo de K-Fold (K=m).
- m iterações: em cada uma ajusta os parâmetros do modelo em $m-1$ exemplos e valida no exemplo restante.
- Reduz viés pessimista (preserva treino quase inteiro).
- Alta variância da estimativa de erro e caro computacionalmente!

SELEÇÃO DE MODELO

Tanto K-Fold quanto LOO podem ser usados para seleção de modelo (hiperparâmetros).

Leave-One-Out



Bootstrap

- Técnica estatística para calcular a confiança de um estimador obtido de uma amostra cuja distribuição é desconhecida e quando obter mais amostras independentes não é viável.
- A partir de um conjunto de tamanho m, extrair b amostras também de tamanho m COM REPETIÇÃO.
- Usar estas amostras como treino e exemplos que não aparecem (*out-of-bag*) nela como teste.
- Mais popular em *ensembles* (veremos mais à frente).

Bootstrap

- Técnica estatística para calcular a confiança de um estimador obtido de uma amostra cuja distribuição é desconhecida e quando obter mais amostras independentes não é viável.
- A partir de um conjunto de tamanho m, extrair b amostras também de tamanho m COM REPETIÇÃO.
- Usar estas amostras como treino e exemplos que não aparecem (*out-of-bag*) nela como teste.
- Mais popular em *ensembles* (veremos mais à frente).

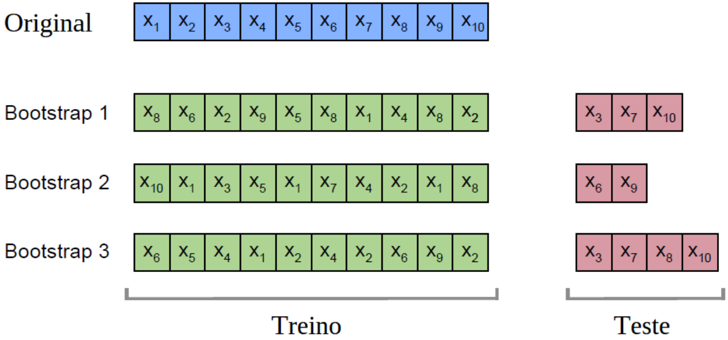
Bootstrap

- Técnica estatística para calcular a confiança de um estimador obtido de uma amostra cuja distribuição é desconhecida e quando obter mais amostras independentes não é viável.
- A partir de um conjunto de tamanho m, extrair b amostras também de tamanho m COM REPETIÇÃO.
- Usar estas amostras como treino e exemplos que não aparecem (*out-of-bag*) nela como teste.
- Mais popular em *ensembles* (veremos mais à frente).

Bootstrap

- Técnica estatística para calcular a confiança de um estimador obtido de uma amostra cuja distribuição é desconhecida e quando obter mais amostras independentes não é viável.
- A partir de um conjunto de tamanho m, extrair b amostras também de tamanho m COM REPETIÇÃO.
- Usar estas amostras como treino e exemplos que não aparecem (*out-of-bag*) nela como teste.
- Mais popular em *ensembles* (veremos mais à frente).

Bootstrap


Original
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}
Bootstrap 1
x_8 x_6 x_2 x_9 x_5 x_8 x_1 x_4 x_8 x_2
x_3 x_7 x_{10}
Bootstrap 2
x_{10} x_1 x_3 x_5 x_1 x_7 x_4 x_2 x_1 x_8
x_6 x_9
Bootstrap 3
x_6 x_5 x_4 x_1 x_2 x_4 x_2 x_6 x_9 x_2
x_3 x_7 x_8 x_{10}
Treino
Teste