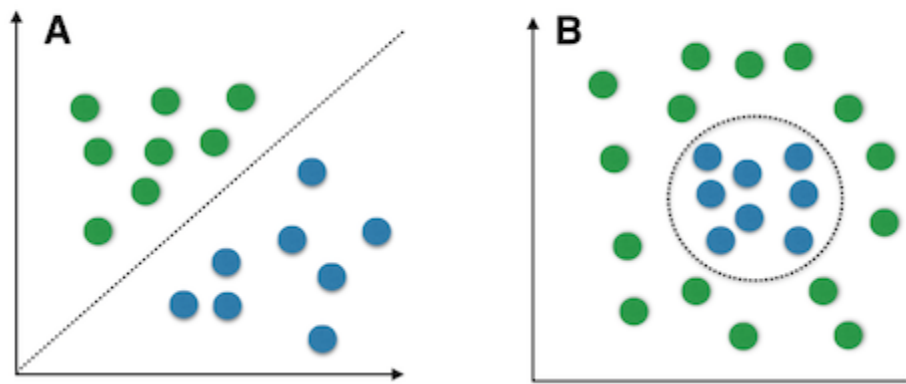


Modelos Não-Lineares

André E. Lazzaretti

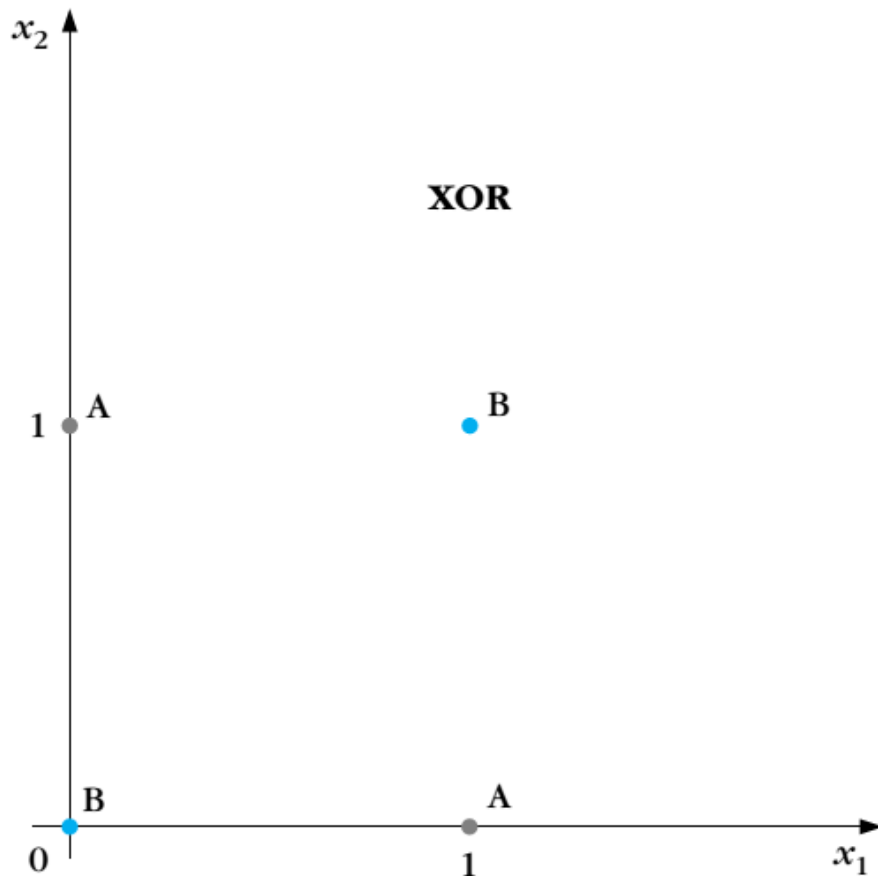
UTFPR/CPGEI

Linear vs. nonlinear problems



Objetivos

- Problema do OU-Exclusivo:



x_1	x_2	XOR	Class
0	0	0	B
0	1	1	A
1	0	1	A
1	1	0	B

É possível resolver esse problema com um classificador linear?

Máquina de Vetor Suporte Linear

- O problema de otimização tem como objetivo a maximização da margem, mantendo o número de pontos com $\xi > 0$, o menor possível:

$$\text{minimize } J(\mathbf{w}, w_0, \boldsymbol{\xi}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$$

$$\text{subject to } y_i [\mathbf{w}^T \mathbf{x}_i + w_0] \geq 1 - \xi_i, \quad i = 1, 2, \dots, N$$

$$\xi_i \geq 0, \quad i = 1, 2, \dots, N$$

Máquina de Vetor Suporte Linear

- Formulação:

$$\mathcal{L}(\mathbf{w}, w_0, \boldsymbol{\xi}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \mu_i \xi_i - \sum_{i=1}^N \lambda_i [y_i(\mathbf{w}^T \mathbf{x}_i + w_0) - 1 + \xi_i]$$



$$\max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right)$$

subject to $0 \leq \lambda_i \leq C, \quad i = 1, 2, \dots, N$

$$\sum_{i=1}^N \lambda_i y_i = 0$$

Máquina de Vetor Suporte Linear

- Uma vez que o hiperplano está determinado (\mathbf{w} e w_0), a classificação de um novo padrão é feita de acordo com o sinal (+ ou -) resultante de:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

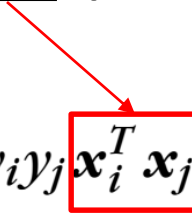
$$= \sum_{i=1}^{N_s} \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + w_0$$

Detalhe Importante

- A solução do problema de programação quadrática com desigualdades lineares como restrições tem a propriedade interessante de que os dados entram somente na forma de produto escalar ($\mathbf{x}^T \mathbf{x}$):

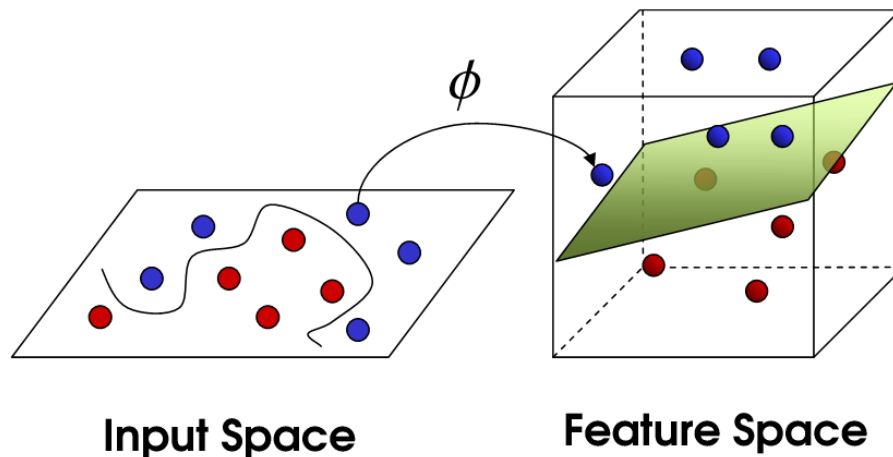
$$\max_{\boldsymbol{\lambda}} \left(\sum_{i=1}^N \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \right)$$

subject to $0 \leq \lambda_i \leq C, \quad i = 1, 2, \dots, N$

$$\sum_{i=1}^N \lambda_i y_i = 0$$


Kernels

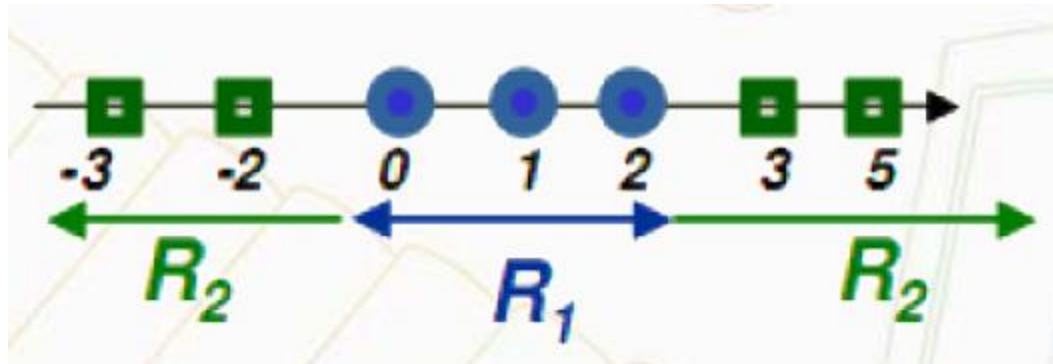
- O algoritmo linear depende somente de $\mathbf{x}^T \mathbf{x}$, portanto pode-se utilizar uma transformação $\phi(\cdot)$, tal que o algoritmo possa ser reescrito em termos de $\phi(\mathbf{x}^T) \phi(\mathbf{x})$
- Pode-se escolher uma transformação $\phi(\cdot)$, tal que:



Define-se como kernel, uma função $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \phi(\mathbf{y})$

Kernels

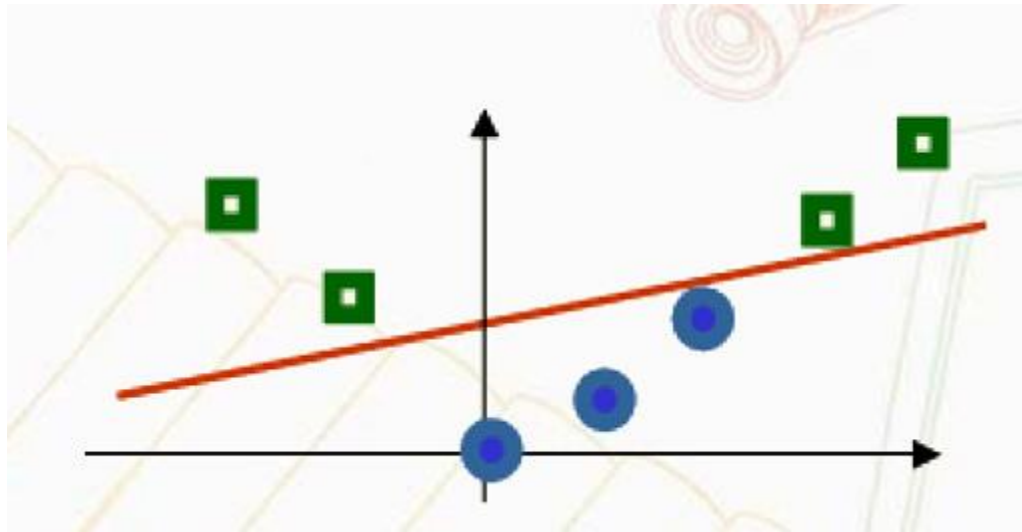
- Como separar linearmente estes exemplos de duas classes?



- Elevando para uma dimensão linearmente separável: $R^1 \rightarrow R^2$

Kernels

- $\phi(\mathbf{x}) = (\mathbf{x}, \mathbf{x}^2)$



- Elevando a dimensionalidade \rightarrow tornar o problema linearmente separável!

Teorema de Mercer

- Considerando o mapeamento para um novo espaço H , definido como um espaço de Hilbert: $\mathbf{x} \mapsto \phi(\mathbf{x}) \in H$
- Um espaço de Hilbert é definido como um espaço completamente linear, onde o produto interno é uma operação definida. Um espaço de Hilbert com dimensão finita é um espaço Euclideano.

Teorema de Mercer

- Para qualquer função contínua $K(\mathbf{x}, \mathbf{z})$ que respeite as condições do teorema de Mercer, existe um espaço no qual $K(\mathbf{x}, \mathbf{z})$ define um produto interno:

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle = K(\mathbf{x}, \mathbf{z})$$

- Onde $\langle ., . \rangle$ representa o produto interno.
- Prova do teorema para um dado kernel é complexa.

Teorema de Mercer

- Uma conclusão importante desse teorema é que não é necessário conhecer o mapeamento $\phi(\cdot)$, uma vez que somente o produto interno nesse espaço é necessário para a formulação do modelo;
- Além disso, não é possível saber a dimensionalidade desse novo espaço, que pode ser inclusive infinita em determinados casos!
 - Foge ao escopo da disciplina...
- Como checar Teorema de Mercer na prática: verificar se a matriz do kernel é **positiva semi-definida**.

Kernels mais Usuais

- Polinomial:

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + 1)^q, \quad q > 0$$

- **Gaussiano (Radial Basis Function):**

$$K(\mathbf{x}, \mathbf{z}) = \exp \left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{\sigma^2} \right)$$

- Tangente Hiperbólica:

$$K(\mathbf{x}, \mathbf{z}) = \tanh \left(\beta \mathbf{x}^T \mathbf{z} + \gamma \right)$$

Máquina de Vetor Suporte Não-Linear

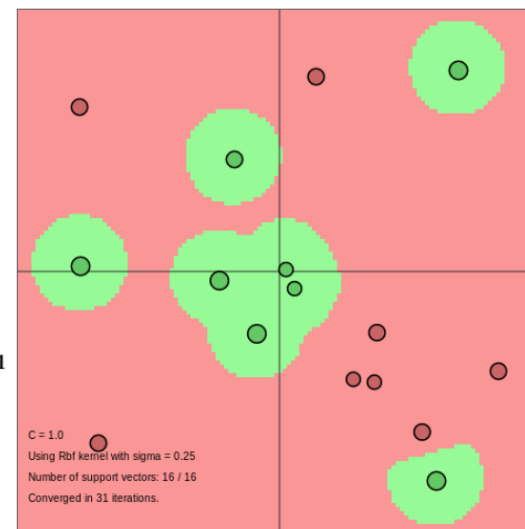
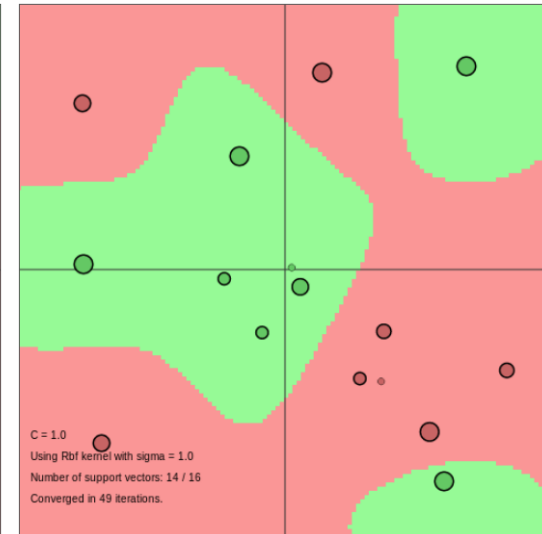
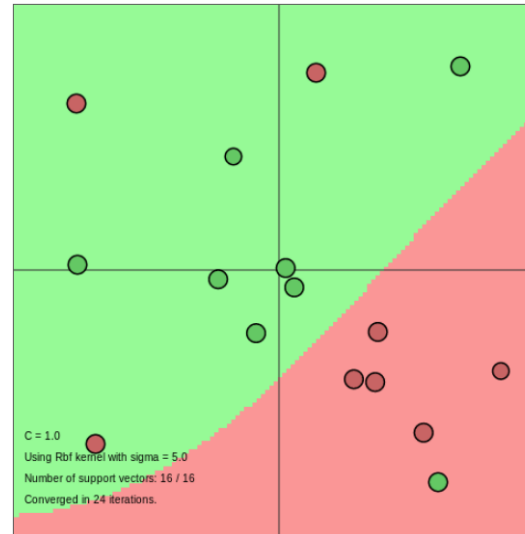
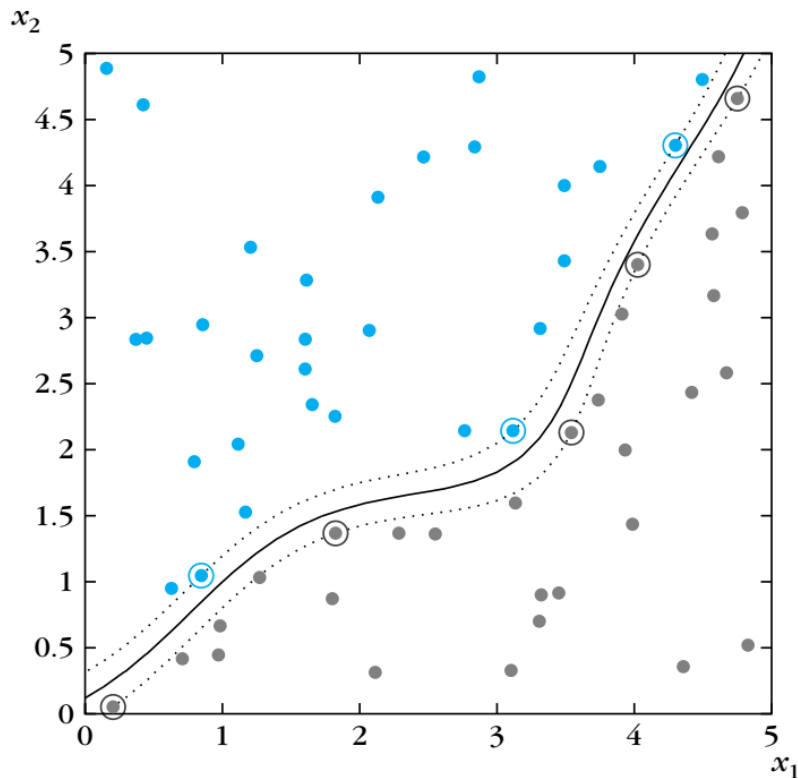
$$\max_{\boldsymbol{\lambda}} \left(\sum_i \lambda_i - \frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \right)$$

$$\text{subject to } 0 \leq \lambda_i \leq C, \quad i = 1, 2, \dots, N$$

$$\sum_i \lambda_i y_i = 0$$

$$\text{assign } \mathbf{x} \text{ in } \omega_1(\omega_2) \text{ if } g(\mathbf{x}) = \sum_{i=1}^{N_s} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0 > (<) 0$$

Máquina de Vetor Suporte Não-Linear



- Maior o sigma \rightarrow mais "suave" o classificador \rightarrow reduz overfitting \rightarrow classificador global
- Menor o sigma \rightarrow mais "sharp" o classificador \rightarrow overfitting \rightarrow classificador local

Generalização da Ideia do Kernel

- O uso do kernel pode ser generalizado para outros algoritmos, como por exemplo kernel perceptron, kernel least squares, etc...
- Isso é possível quando pode-se escrever os produtos em termos de produtos internos.
- Exemplo: Reescrever um classificador baseado na distância Euclideana, utilizando as definições e propriedades do kernel. Ideia:

classifies it to
the ω_2 class if

$$|| \mathbf{x} - \boldsymbol{\mu}_1 ||^2 > || \mathbf{x} - \boldsymbol{\mu}_2 ||^2$$

Centroide da classe
para facilitar

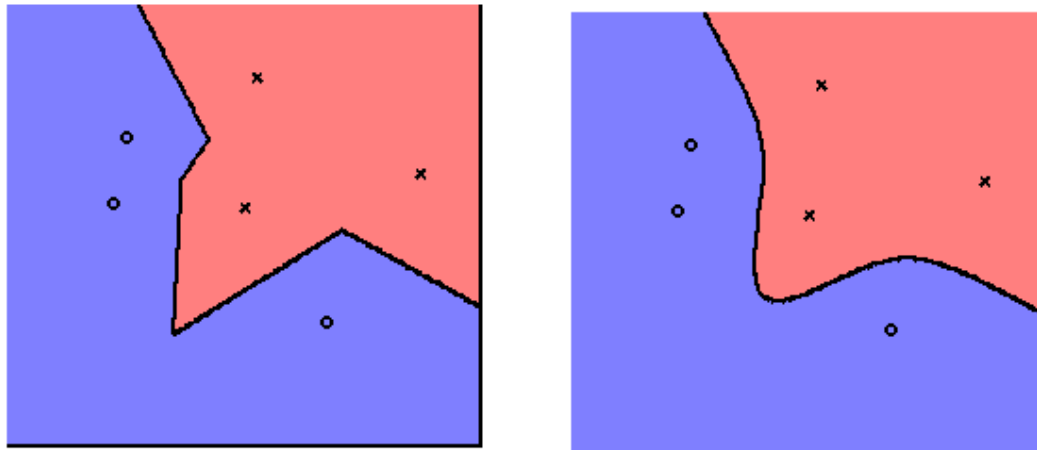
Exercício

- Dicas:

$$\|\phi(\mathbf{x}) - \mu_1\|^2 > \|\phi(\mathbf{x}) - \mu_2\|^2$$

$$\mu_1 = \frac{1}{N_1} \sum_{i: y_i = +1} \phi(\mathbf{x}_i) \quad \text{and} \quad \mu_2 = \frac{1}{N_2} \sum_{i: y_i = -1} \phi(\mathbf{x}_i)$$

- Resultado:

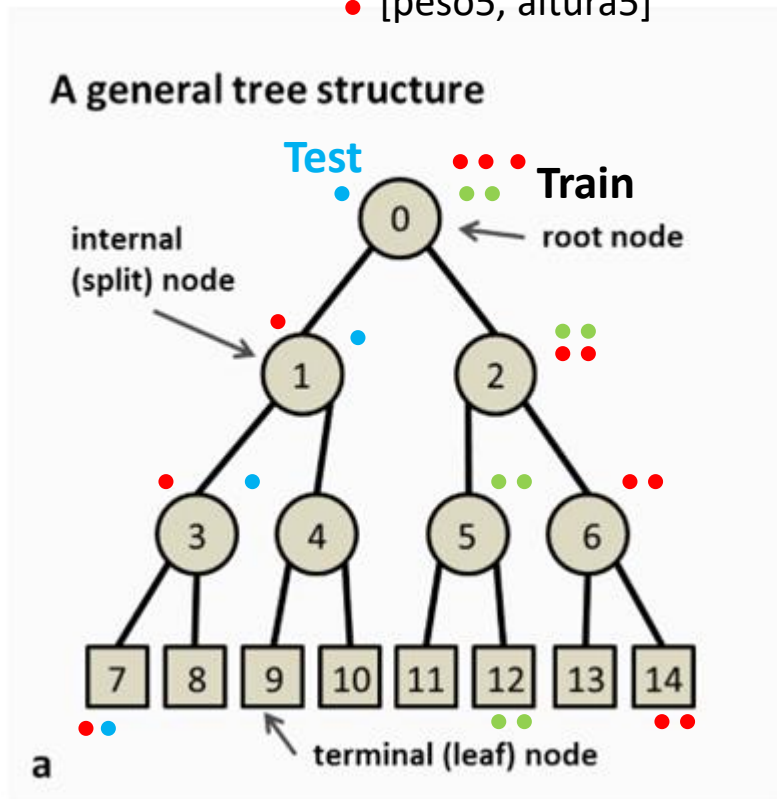


Exemplo
Matlab!

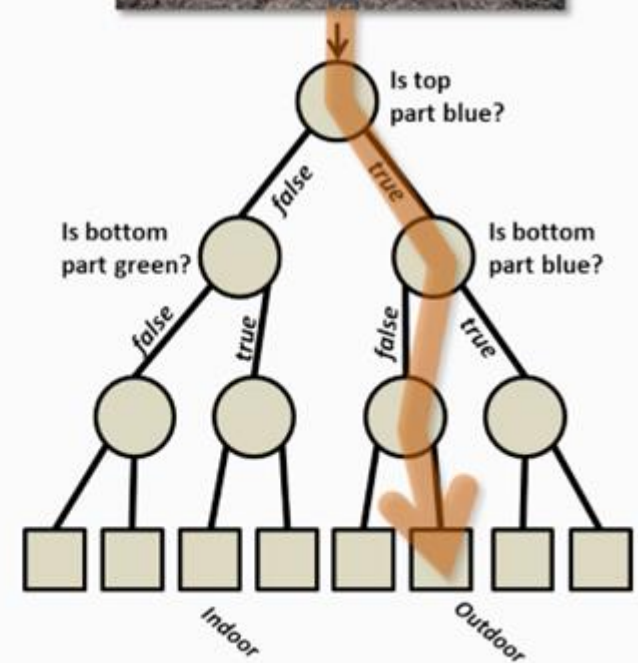
Árvores de Decisão

- Visão Geral:

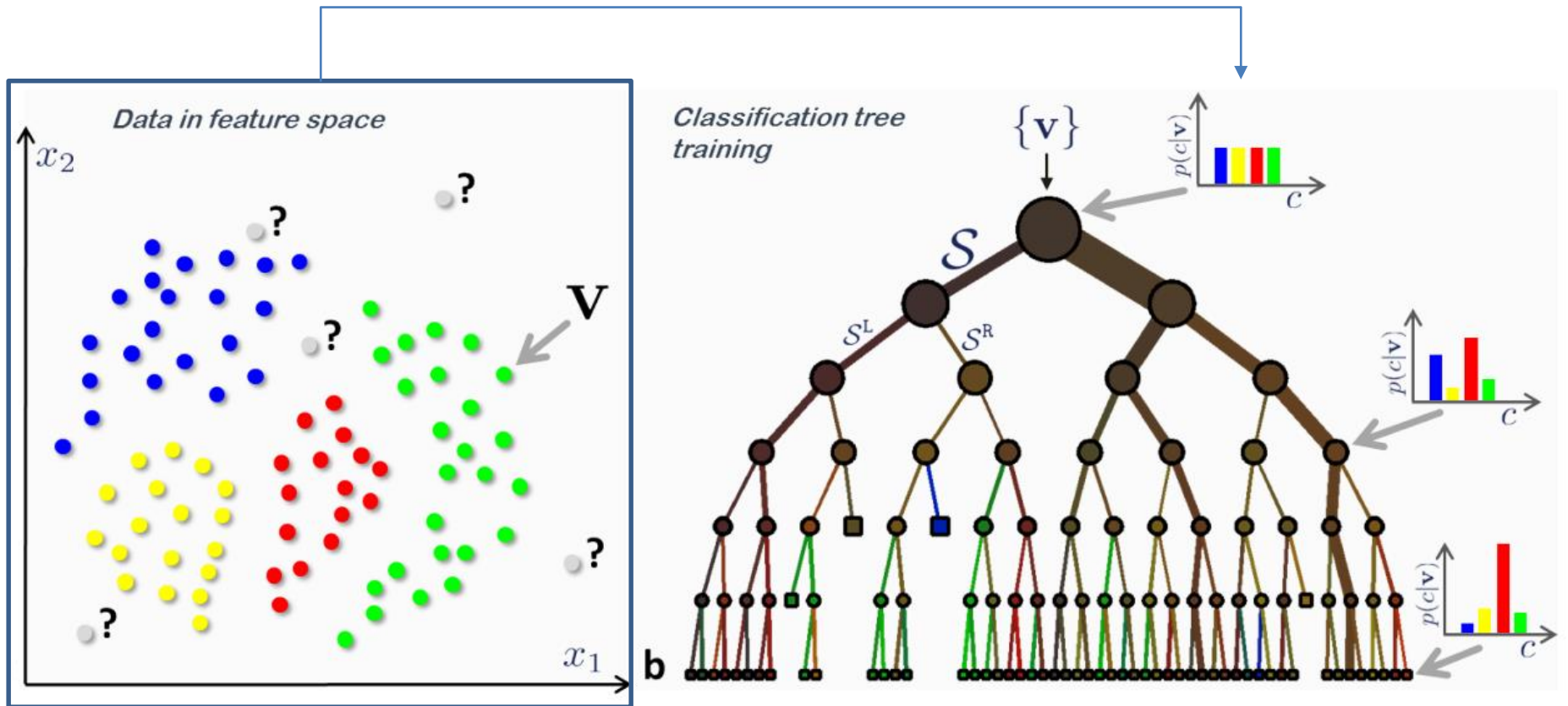
- [peso1, altura1]
- [peso2, altura2]
- [peso3, altura3]
- [peso4, altura4]
- [peso5, altura5]



A decision tree



Árvores de Decisão

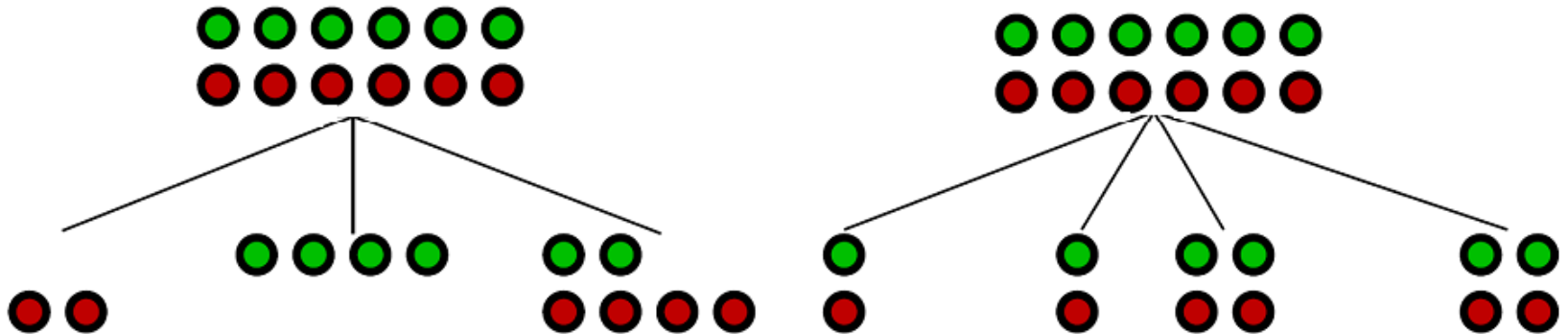


$$\mathbf{v} = (x_1, \dots, x_d) \in \mathbb{R}^d$$

$$\mathcal{S}_j = \mathcal{S}_j^L \cup \mathcal{S}_j^R$$

Árvores de Decisão

- Como construir a árvore (as divisões)?
 - Um bom atributo separa os exemplos em subconjuntos que, idealmente, são todos positivos e todos negativos, por exemplo:

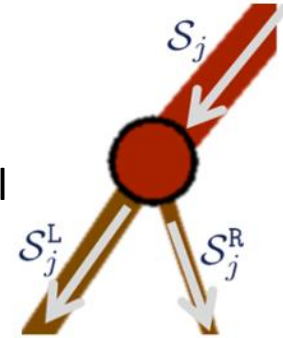


- Para isso, pode-se utilizar o conceito de Ganho de Informação (Information Gain) ou redução de entropia;

Árvores de Decisão

- Treinamento:

$|S|$ - número total
de exemplos



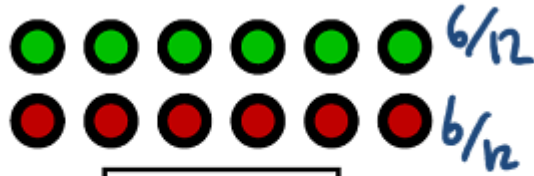
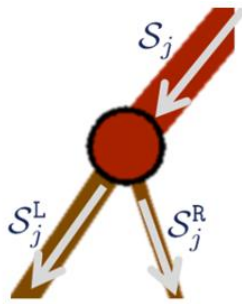
$$I_j = H(\mathcal{S}_j) - \sum_{i \in \{L, R\}} \frac{|\mathcal{S}_j^i|}{|\mathcal{S}_j|} H(\mathcal{S}_j^i) \longrightarrow \text{Information Gain}$$

$$H(\mathcal{S}) = - \sum_{c \in \mathcal{C}} p(c) \log p(c) \longrightarrow \text{Entropy}$$

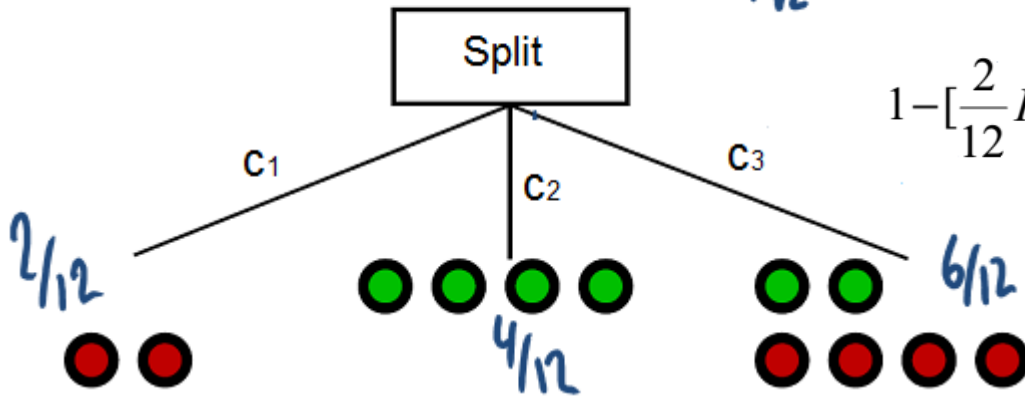
$c \in \{c_k\}$ indexing the class

Ex.: S possui 10 exemplos da classe c_1
e 10 exemplos da classe c_2 :

$$H(S) = H(10/20) + H(10/20) = H(10/20, 10/20)$$



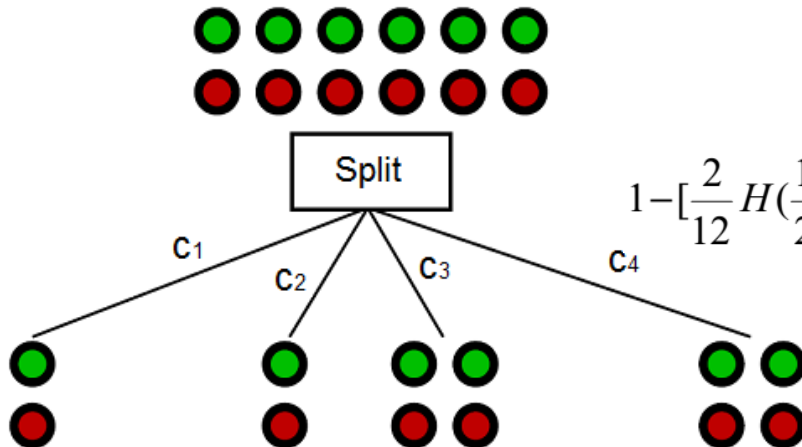
$$p = n = 6, H(6/12, 6/12) = 1 \text{ bit}$$



$$1 - \left[\frac{2}{12} H(0,1) + \frac{4}{12} H(1,0) + \frac{6}{12} H\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .541 \text{ bits}$$

$$I_j = H(S_j) - \sum_{i \in \{L,R\}} \frac{|S_j^i|}{|S_j|} H(S_j^i)$$

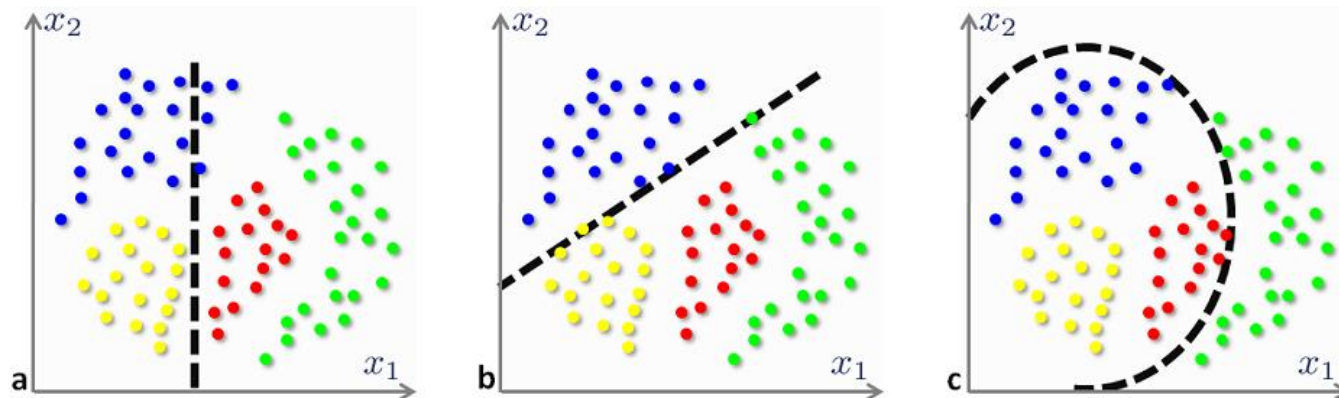
$$H(S) = - \sum_{c \in C} p(c) \log p(c)$$



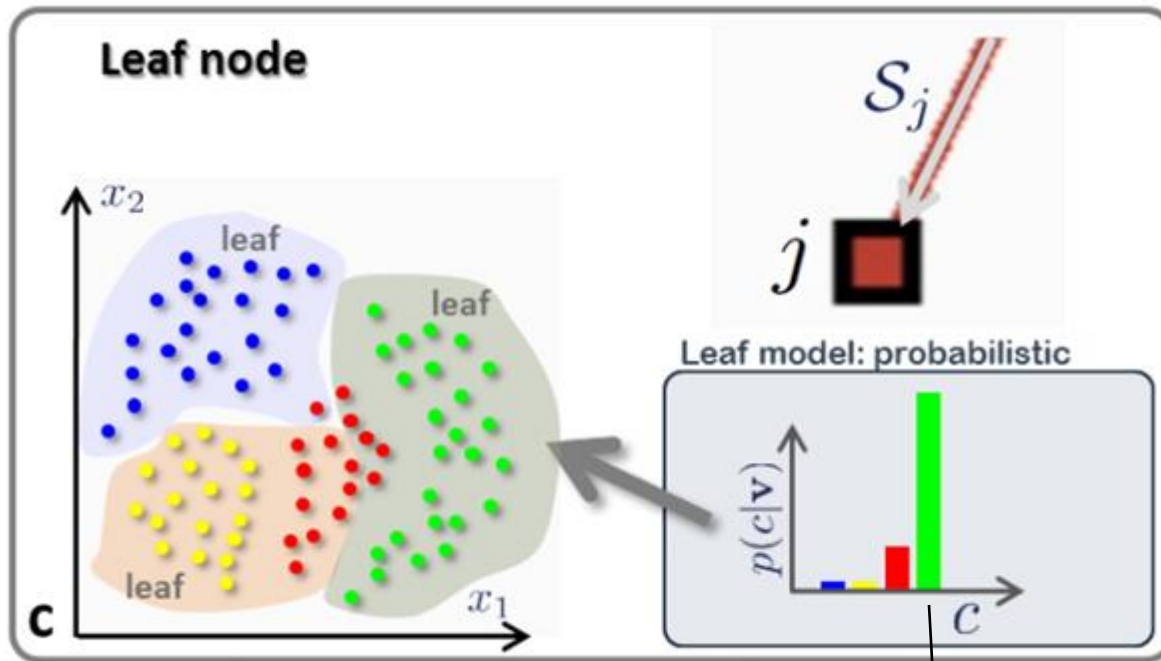
$$1 - \left[\frac{2}{12} H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} H\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} H\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} H\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0 \text{ bits}$$

Árvores de Decisão

- Na prática (versão mais simples):
 - Calcular a entropia do *dataset* como um todo
 - Para cada *feature*:
 - Dividir em intervalos (*grid* linear ou baseado no histograma)
 - Calcular o ganho médio de informação considerando os intervalos
 - Selecione a *feature* com maior ganho médio de informação
 - Repetir o processo (crescer a árvore) até critério de parada ser atingido (tamanho máximo, probabilidade máxima, etc).
- Diversas outras abordagens, cada uma delas um algoritmo diferente
- Por exemplo, outras formas de separar dados de entrada:



Na fase de teste

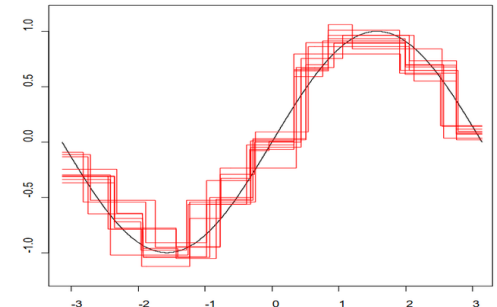


Assinala a classe majoritária!

Random Forests

- Algoritmo geral:
 - Para $b=1$ até B (cada árvore):
 1. Retire um subconjunto Z^* dos dados de treinamento de forma aleatória.
 2. Para cada subconjunto, gere uma árvore de tamanho pré-determinado, repetindo de maneira recursiva os seguintes passos (*random tree*):
 - I. Selecione aleatoriamente um subconjunto de m atributos, dos p disponíveis (apenas uma parte dos atributos são utilizados).
 - II. Determine a melhor divisão (split) dos dados com base no ganho de informação.
 - III. Divida o nó em dois nós-filhos.
- O resultado será dada pela combinação das várias árvores:

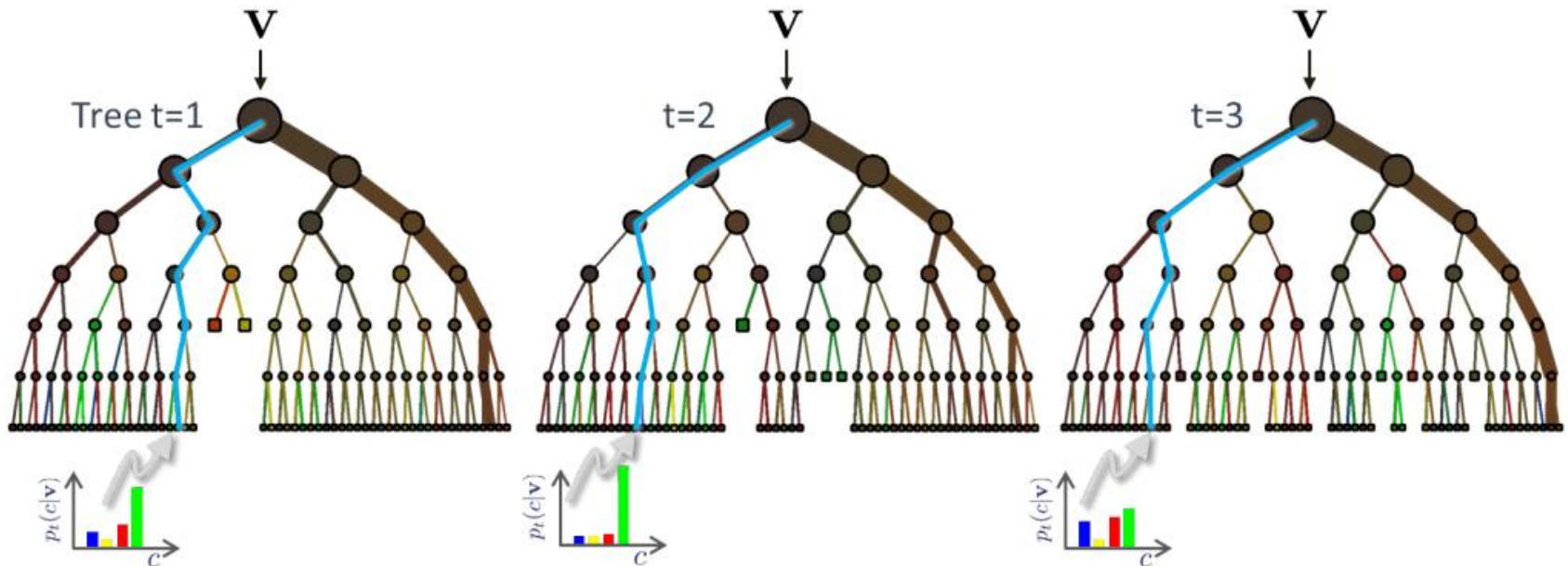
Por que funciona?



Random Forests

- **Treinamento:** as árvores são treinadas separadamente e individualmente (normalmente de forma distribuída).

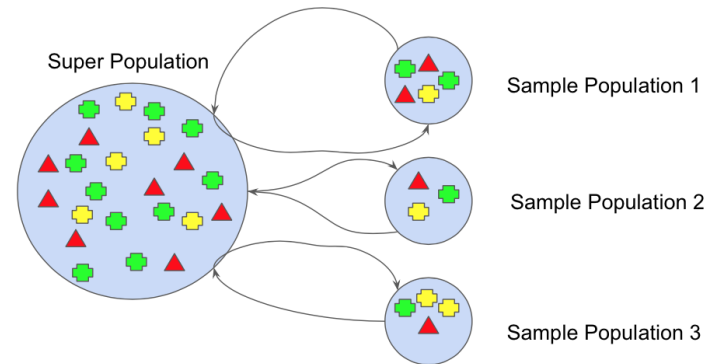
- **Teste:**
$$p(c|\mathbf{v}) = \frac{1}{T} \sum_t p_t(c|\mathbf{v})$$



Bootstrapping, Bagging e Boosting

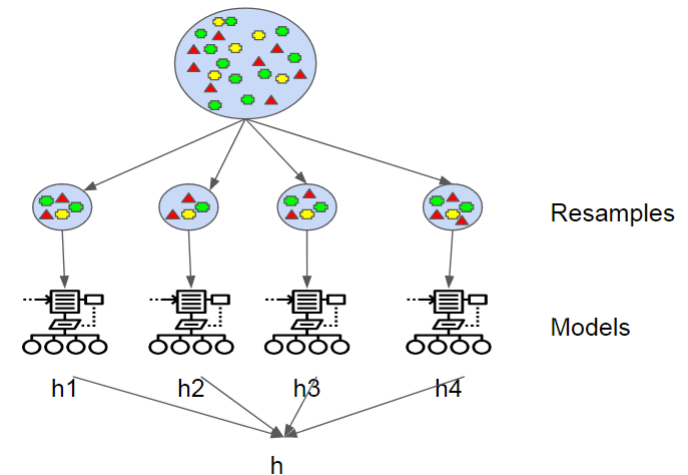
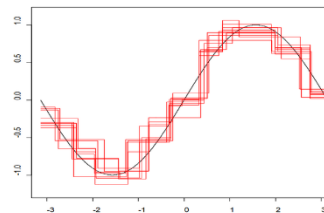
- **Bootstrapping:**

- Amostragem aleatória com substituição.
- Avaliar questões de overfitting, bias, variância, etc.



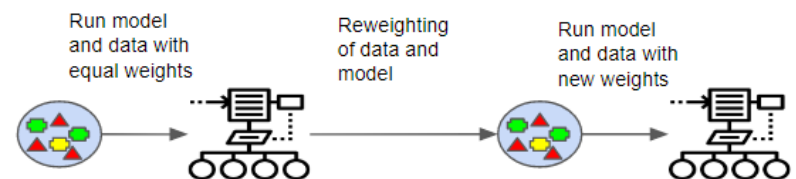
- **Bagging:**

- Cada modelo possui o mesmo peso na decisão final
- Reduz variância
- Random Forest

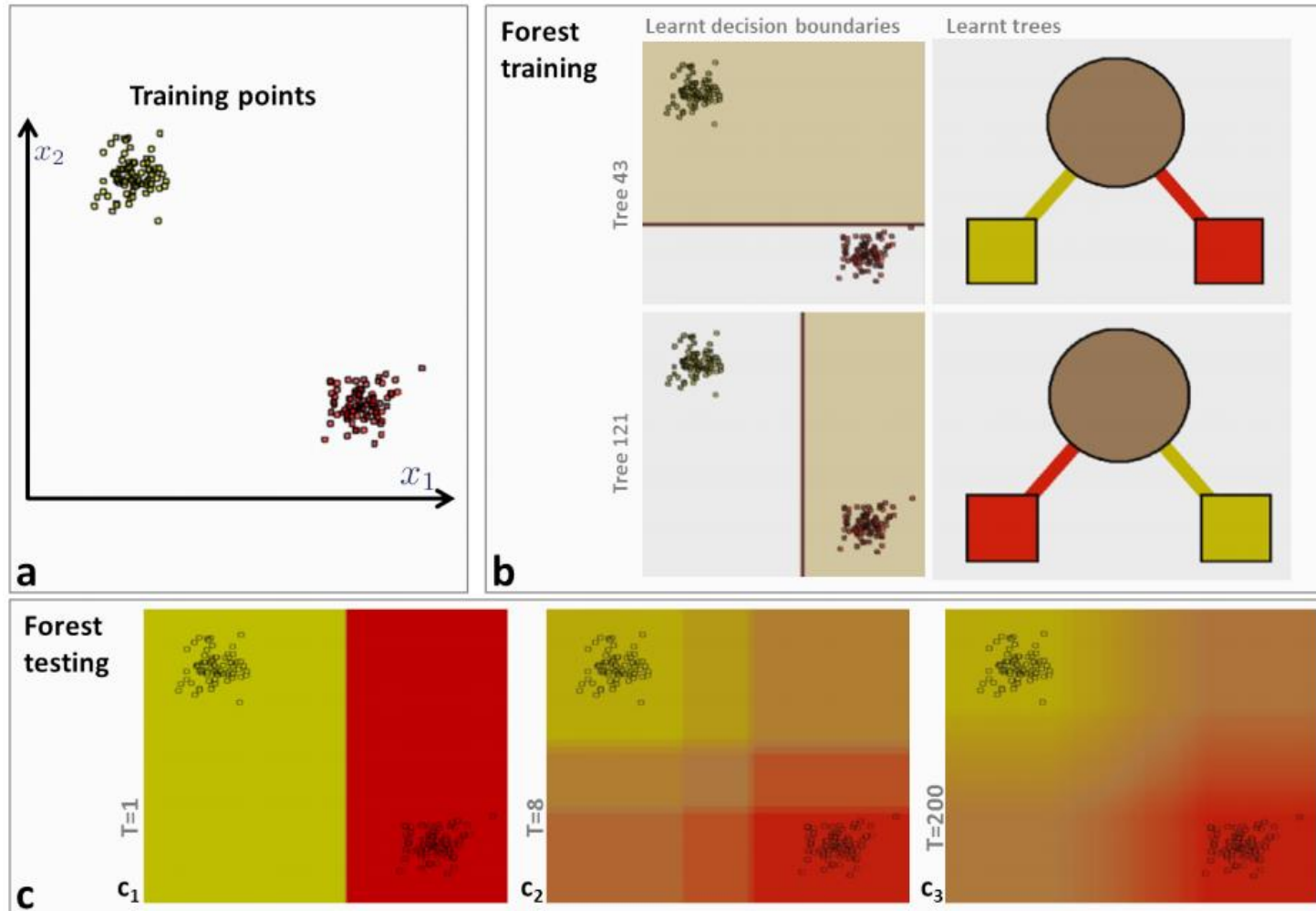


- **Boosting:**

- Cada modelo sequencial enfatiza instâncias que os modelos anteriores classificaram incorretamente.

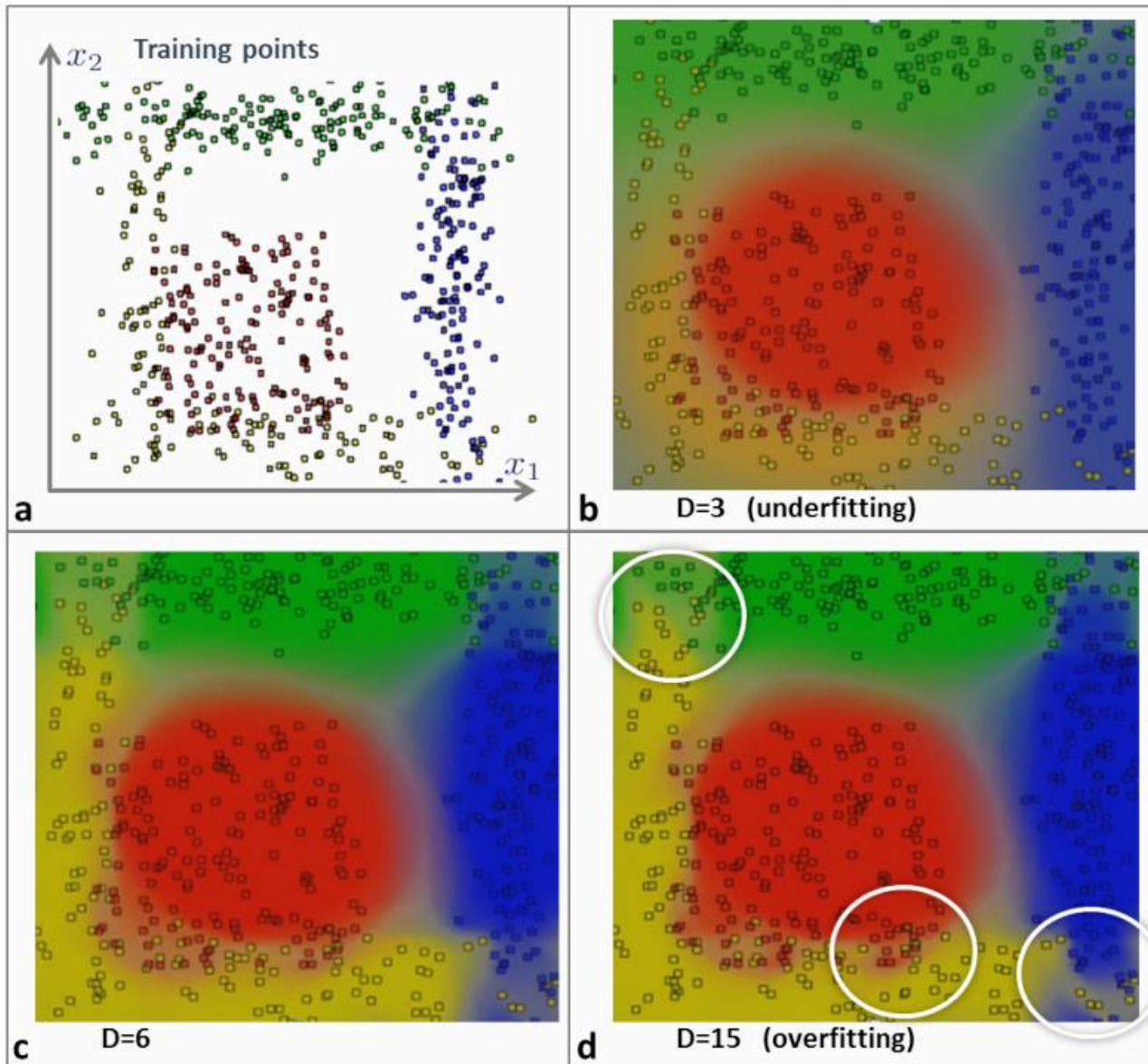


Efeito do tamanho da Forest



Smoother separation \longrightarrow

Efeito da profundidade da árvore

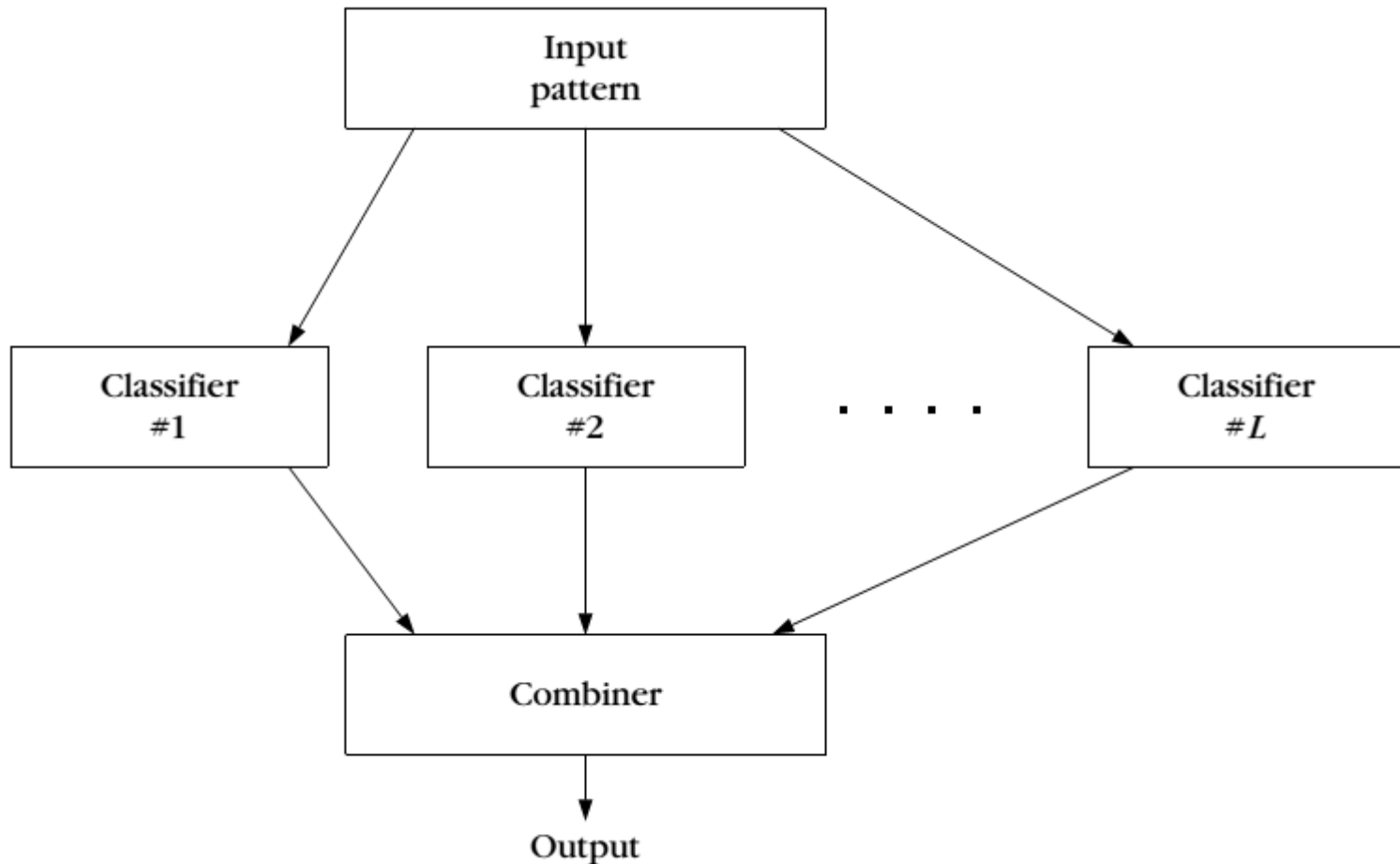


Solução:
Cross-validation

Exemplo
Matlab

Combinação de Classificadores

- Ideia Geral:



Combinação de Classificadores

- Regra da Maioria dos Votos:
Decide-se em favor da classe que apresenta um consenso entre os classificadores ou quando pelo menos l_c classificadores concordam no rótulo atribuído, sendo:

$$l_c = \begin{cases} \frac{L}{2} + 1, & L \text{ even} \\ \frac{L+1}{2}, & L \text{ odd} \end{cases}$$



PERGAMON

Pattern Recognition 33 (2000) 1475–1485

PATTERN
RECOGNITION

THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Combining multiple classifiers by averaging or by multiplying?

David M.J. Tax^{a,*}, Martijn van Breukelen^{a,1}, Robert P.W. Duin^a, Josef Kittler^b

Referências

- Capt. 4 - Livro Theodoridis (Pattern Recognition Fourth Edition e Pattern Recognition Matlab) – SVM e combinação de classificadores;
- Tutorial Regression Trees e Random Forests – Criminisi (Microsoft);
- Aulas Professor Nando de Freitas (UBC/Oxford) – decision trees and random forests;