Mineração de Dados
Aula 3

Especialização em Ciência de Dados e suas Aplicações

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Classificação

Classificação: Definição

- **Dado uma coleção de registros (treino)**
  - **Cada registro é caracterizado pela tupla (x,y), onde x é um conjunto de atributos e y é o rótulo da classe**

- **Tarefa:**
  - **Aprender um modelo que mapeia cada conjunto de atributo x em uma das classes predefinidas y**

Exemplos

Tarefa	Conjunto de atributos (x)	Rótulos das classes (y)
Categorização de emails	Características (features) extraídas das mensagens e cabeçalho	spam ou regular
Identificação de tumor	Features extraídas resonâncias magnéticas	Malígno ou benígno
Categorizar textos	Features extraídas do texto	Cultura, Lazer, Esportes e Educação

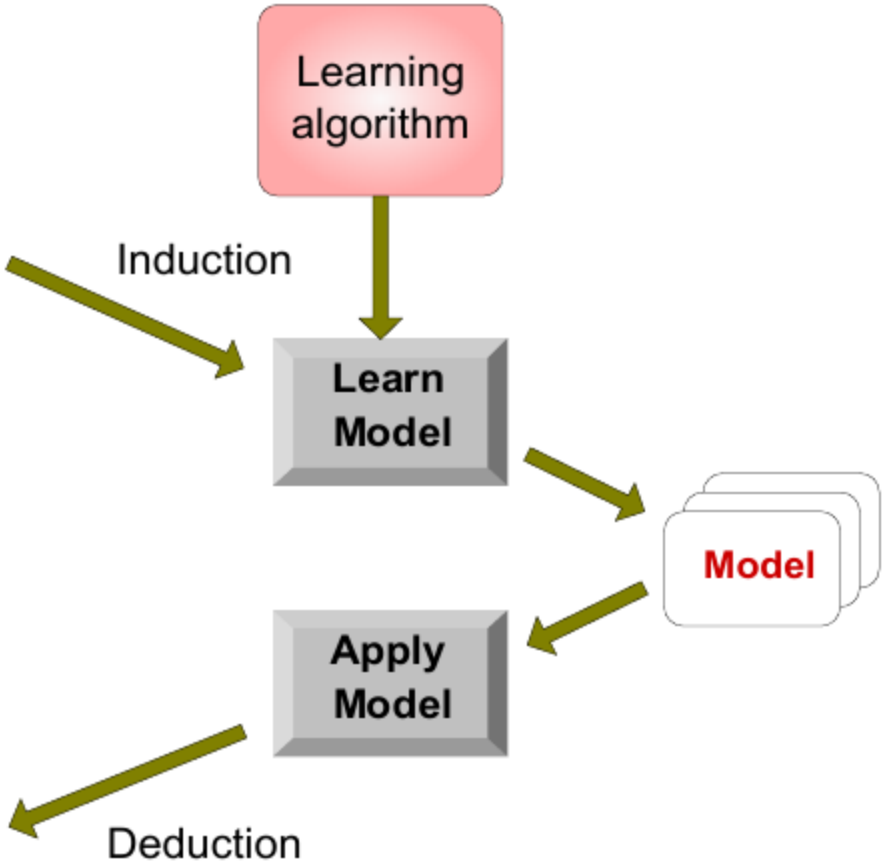
Abordagem geral

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



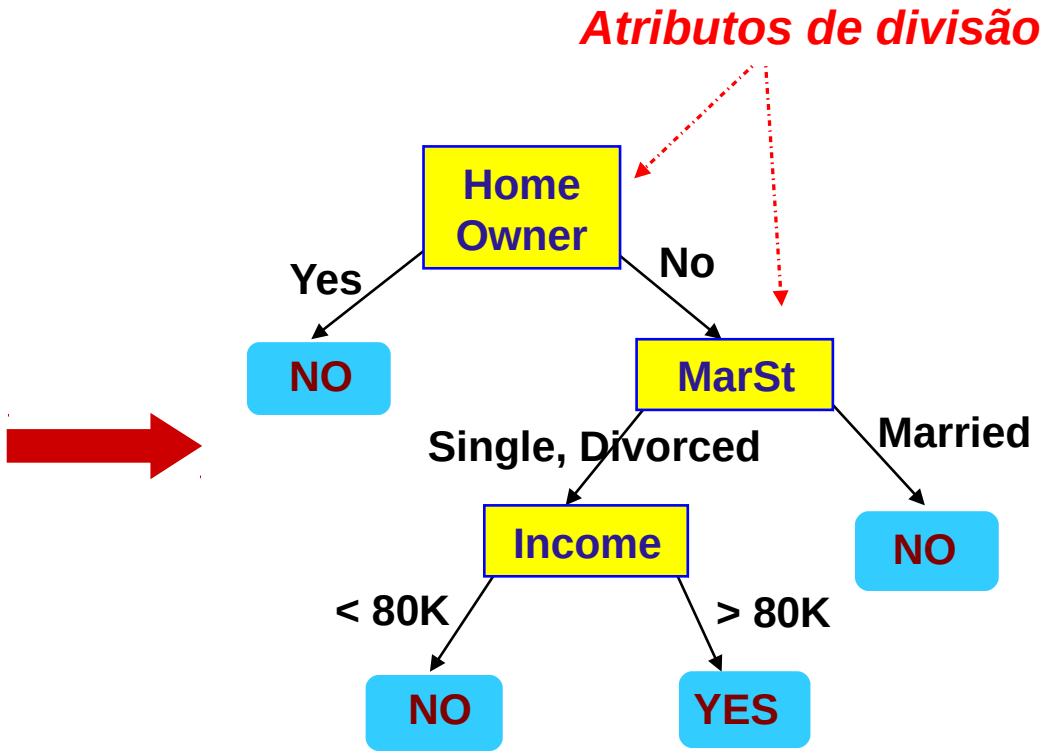
Técnicas de classificação

- Métodos baseados em árvore de decisão
- Naïve Bayes e Redes Bayesianas
- Redes Neurais Artificiais
- ...

Exemplo de árvore de decisão

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
	Categórico	Categórico	Contínuo	Classe
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Dados de treinamento

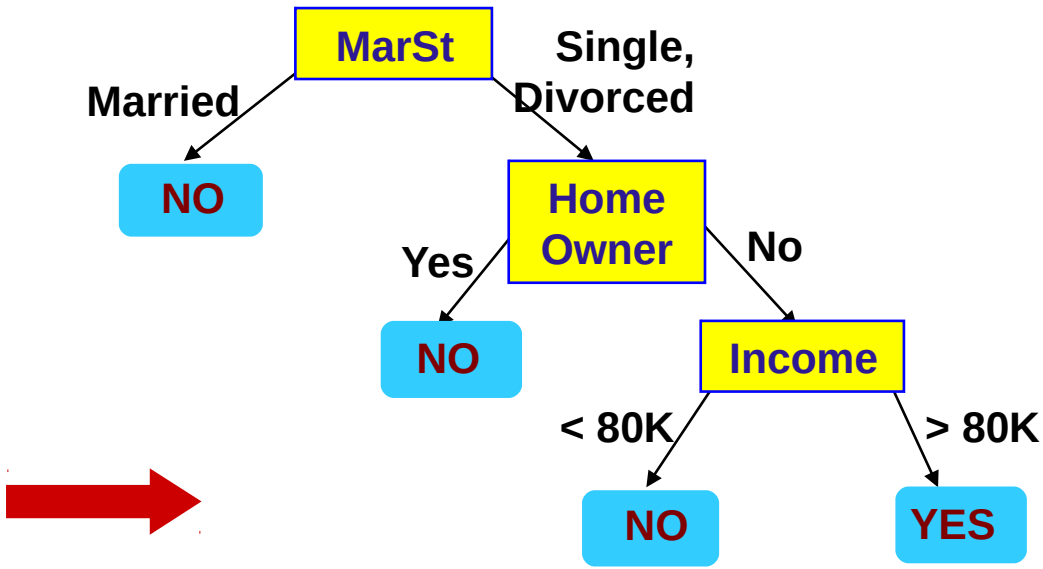


Modelo: Árvore de decisão

Outra árvore de decisão

ID	Home Owner (Categórico)	Marital Status (Categórico)	Annual Income (Contínuo)	Defaulted Borrower (Classe)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Dados de treinamento



Pode haver mais de uma árvore para o mesmo conjunto de dados

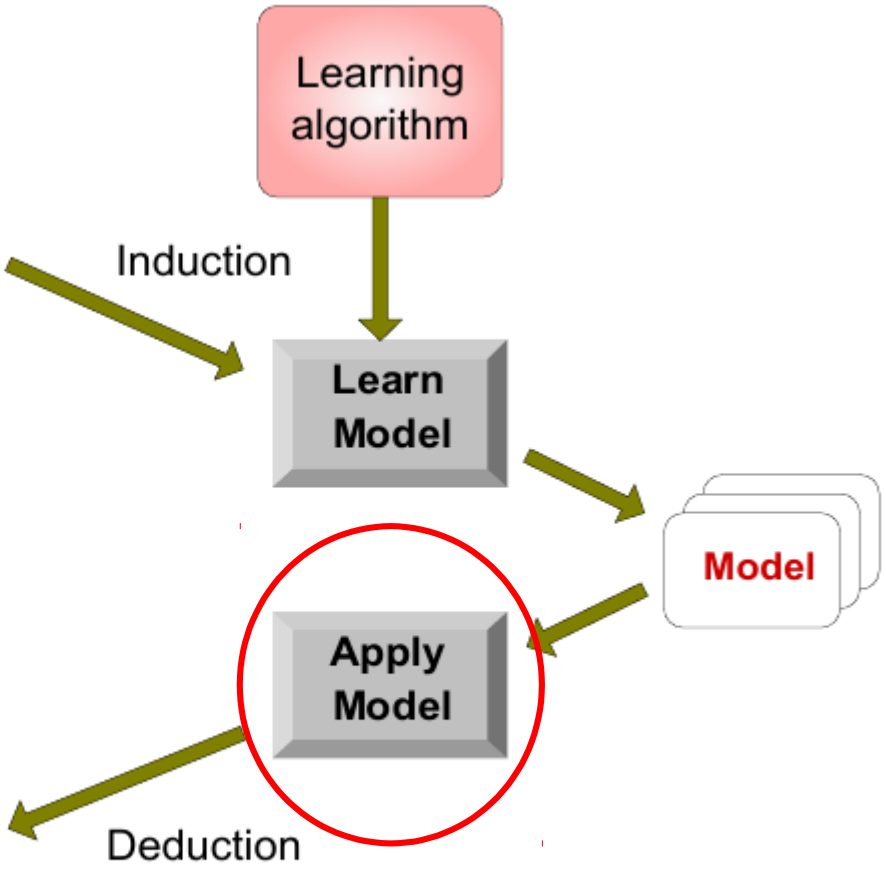
Abordagem geral

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

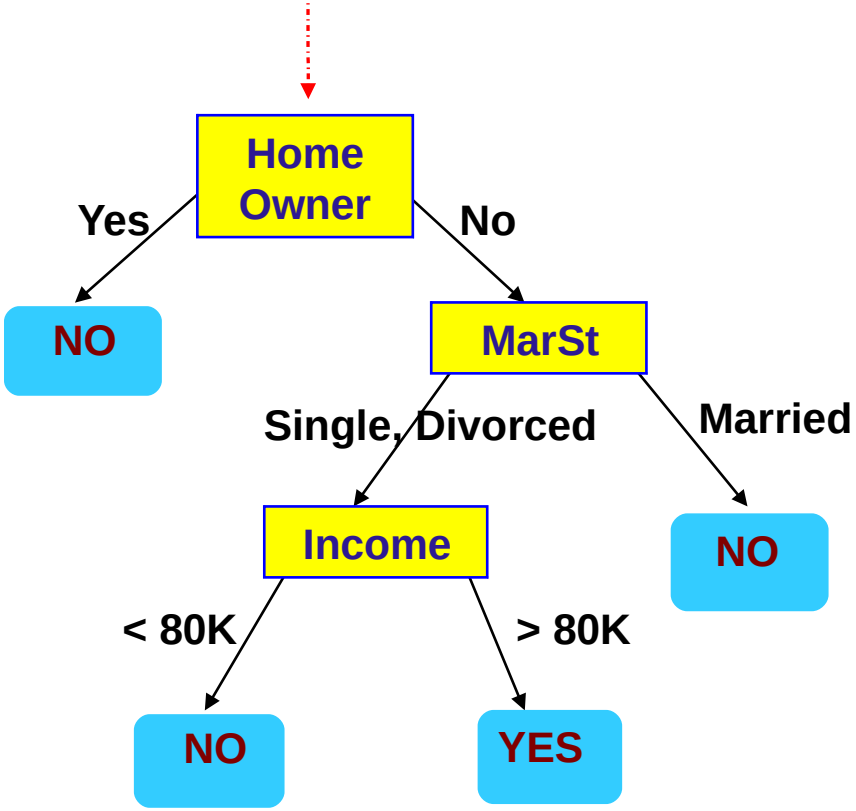
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Aplicar o modelo aos dados de teste

Iniciar da raiz da árvore



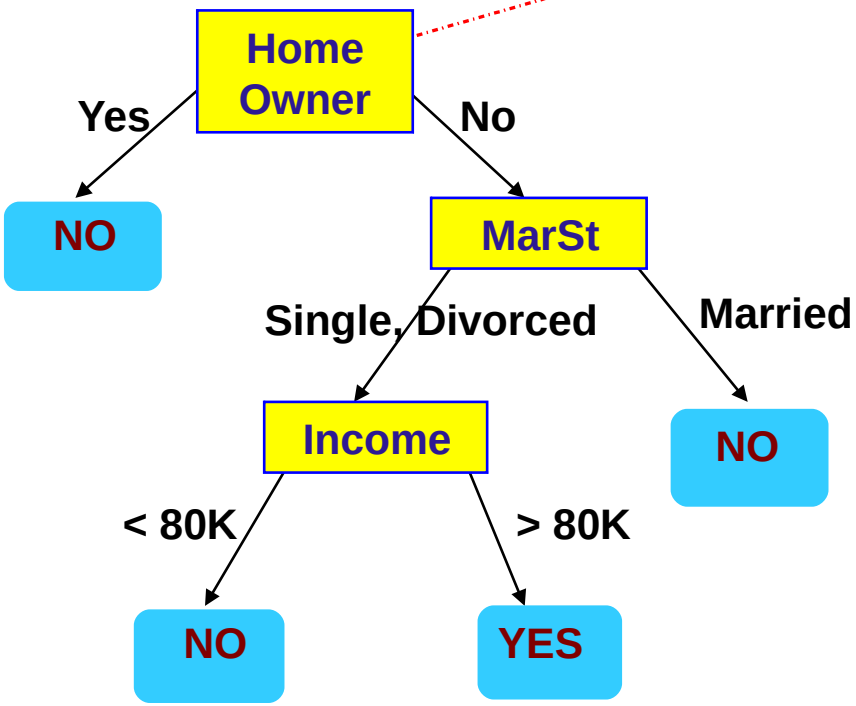
Dados de teste

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Aplicar o modelo aos dados de teste

Dados de teste

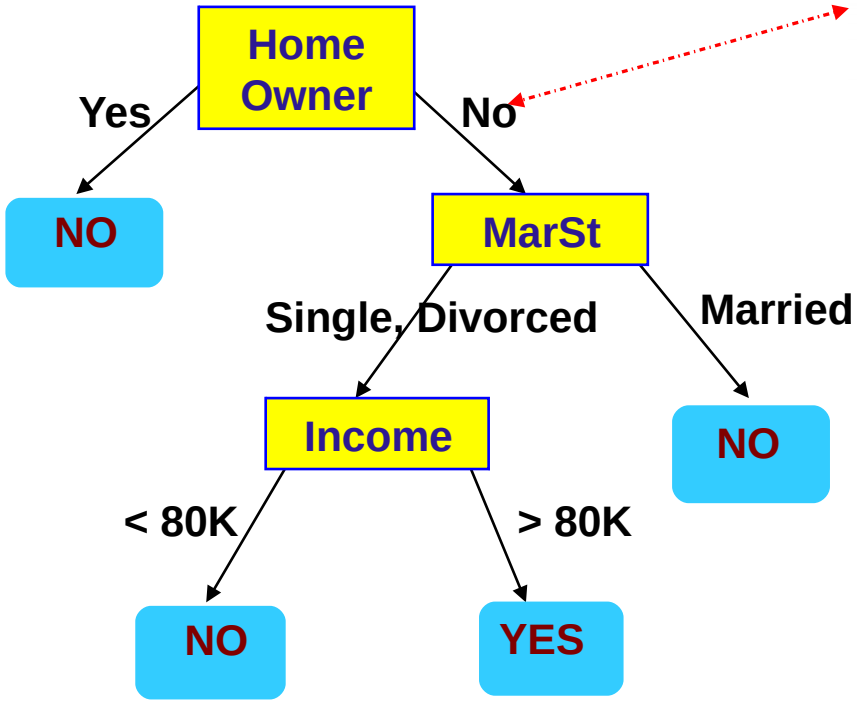
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Aplicar o modelo aos dados de teste

Dados de teste

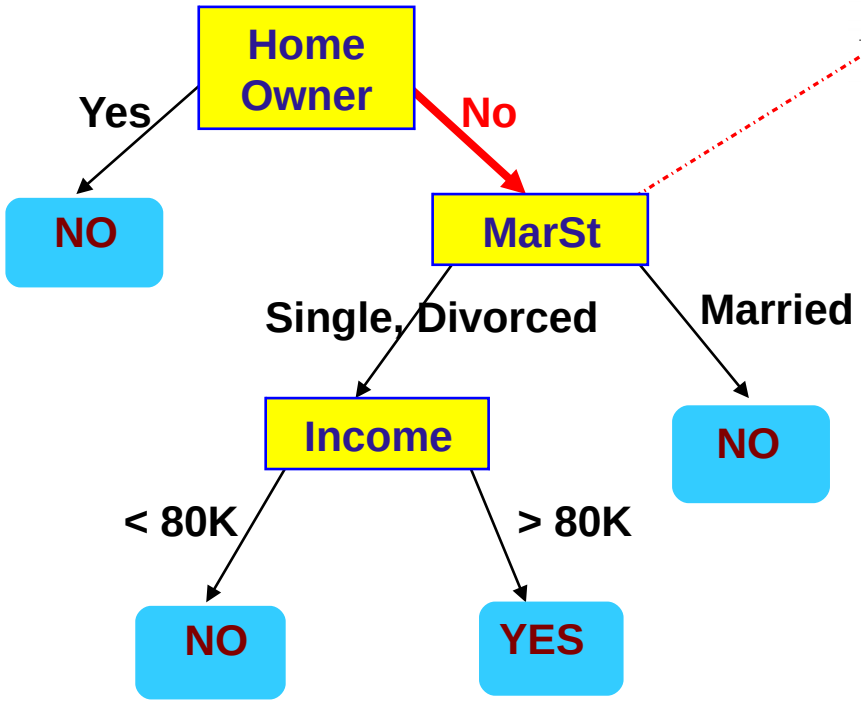
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Aplicar o modelo aos dados de teste

Dados de teste

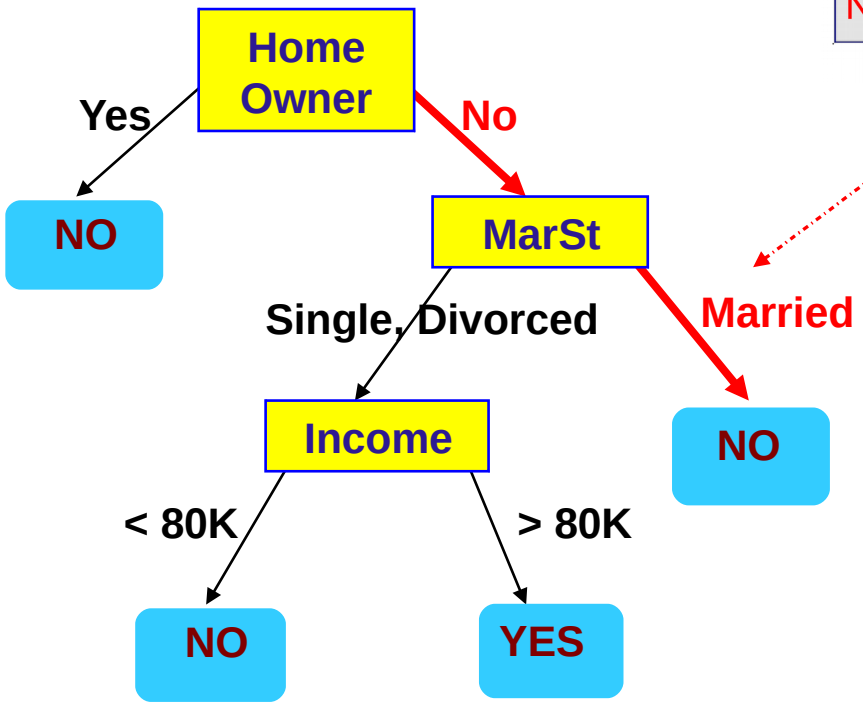
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Aplicar o modelo aos dados de teste

Dados de teste

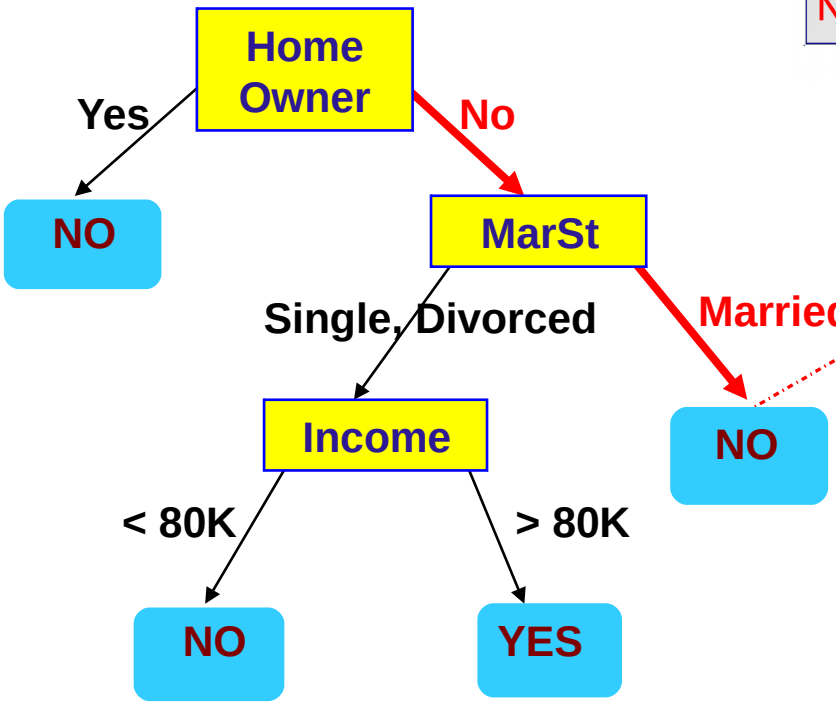
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Aplicar o modelo aos dados de teste

Dados de teste

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Assign Defaulted to "No"

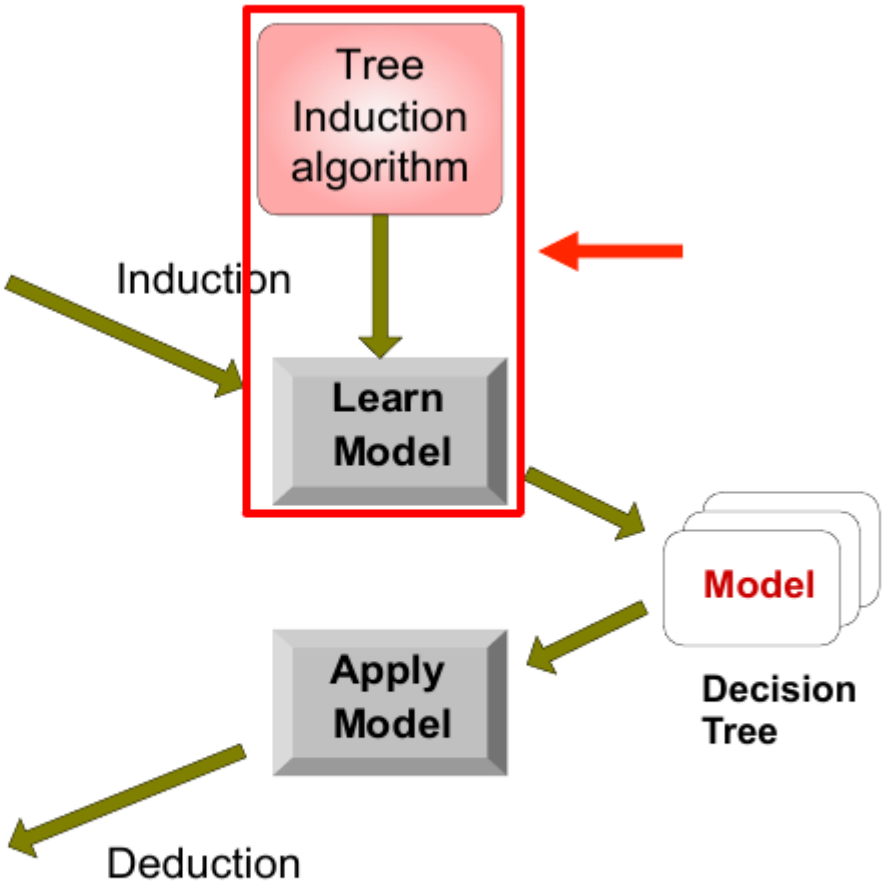
Árvore de decisão

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Indução da árvore de decisão

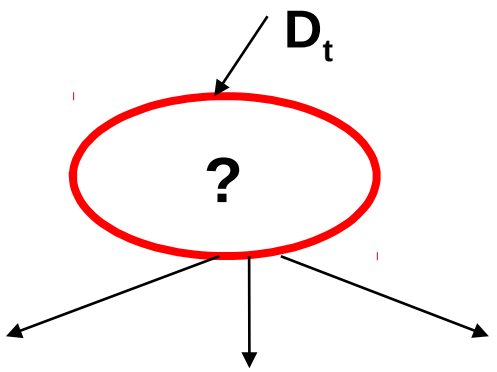
- Vários algoritmos :
  - Algoritmo de Hunt (um dos primeiros)
  - CART
  - ID3, C4.5, C.5
  - SLIQ, SPRINT

Algoritmo de Hunt

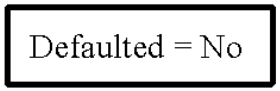
- Seja D_t o conjunto de treino que alcança o nó t

- Procedimento geral:
  - Se D_t contém registos que pertencem à mesma classe y_t, então t é um nó folha rotulado como y_t
  - Se D_t contém registros que pertencem a mais de uma classe, usar um atributo para dividir os dados em subconjuntos menores. Recursivamente aplicar o procedimento para cada subconjunto.

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Algoritmo de Hunt

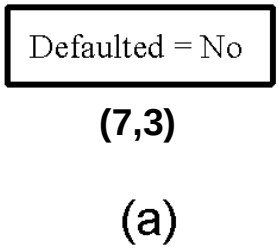


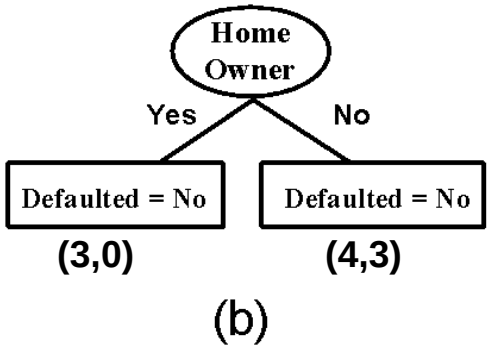
(7,3)

(a)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

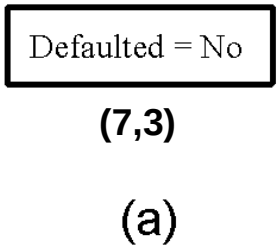
Algoritmo de Hunt

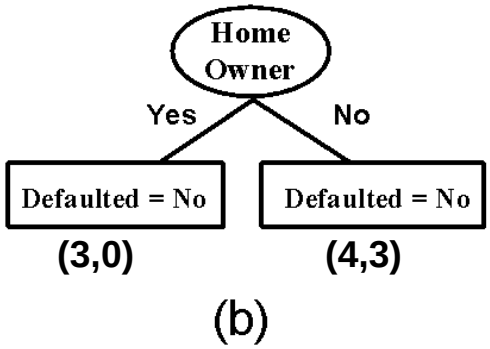




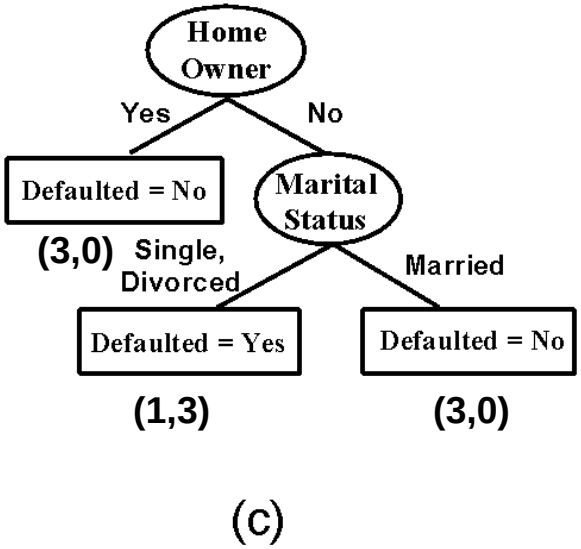
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Algoritmo de Hunt

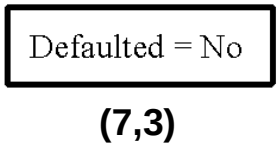




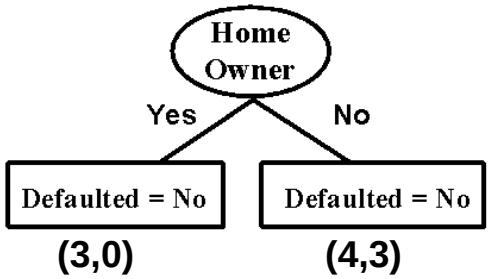
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



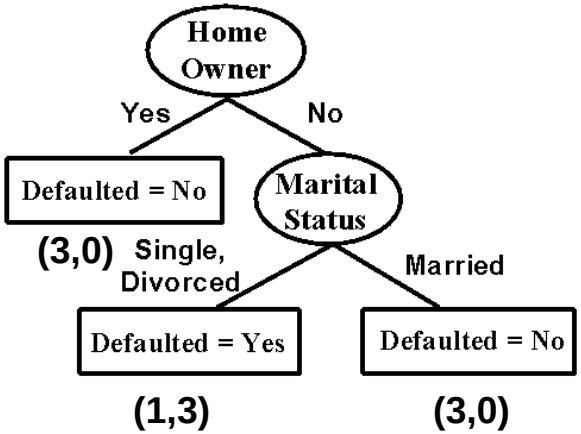
Algoritmo de Hunt



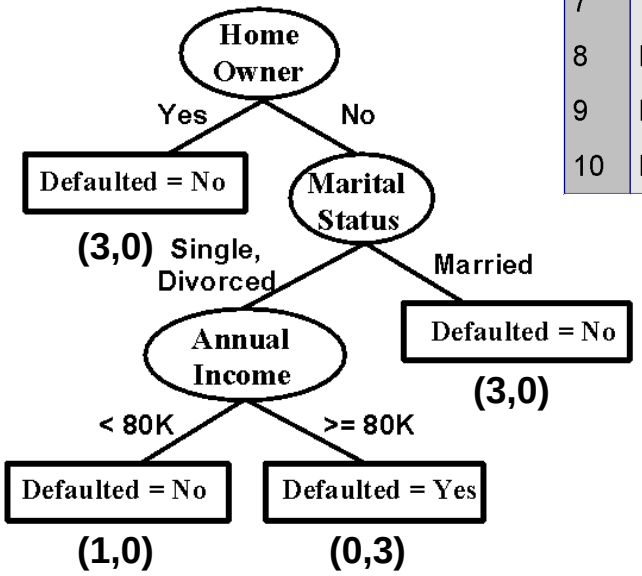
(a)



(b)



(c)



(d)

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Como determinar a melhor divisão

- **Estratégia gulosa:**
  - **Nós com distribuição de classes mais puras são preferidos**
- **Precisa de uma medida de impureza:**

C0: 5 C1: 5

Alto grau de impureza

C0: 9 C1: 1

Baixo grau de impureza

Medidas de impureza

- Índice de Gini
- Entropia
- Erro de classificação

Encontrando a melhor divisão

1. Computar a impureza (P) antes de dividir
2. Computar a impureza (M) depois de dividir
    - Computar a impureza para cada nó filho
    - M é a impureza ponderada para os filhos
3. Escolha o atributo que produz o maior ganho

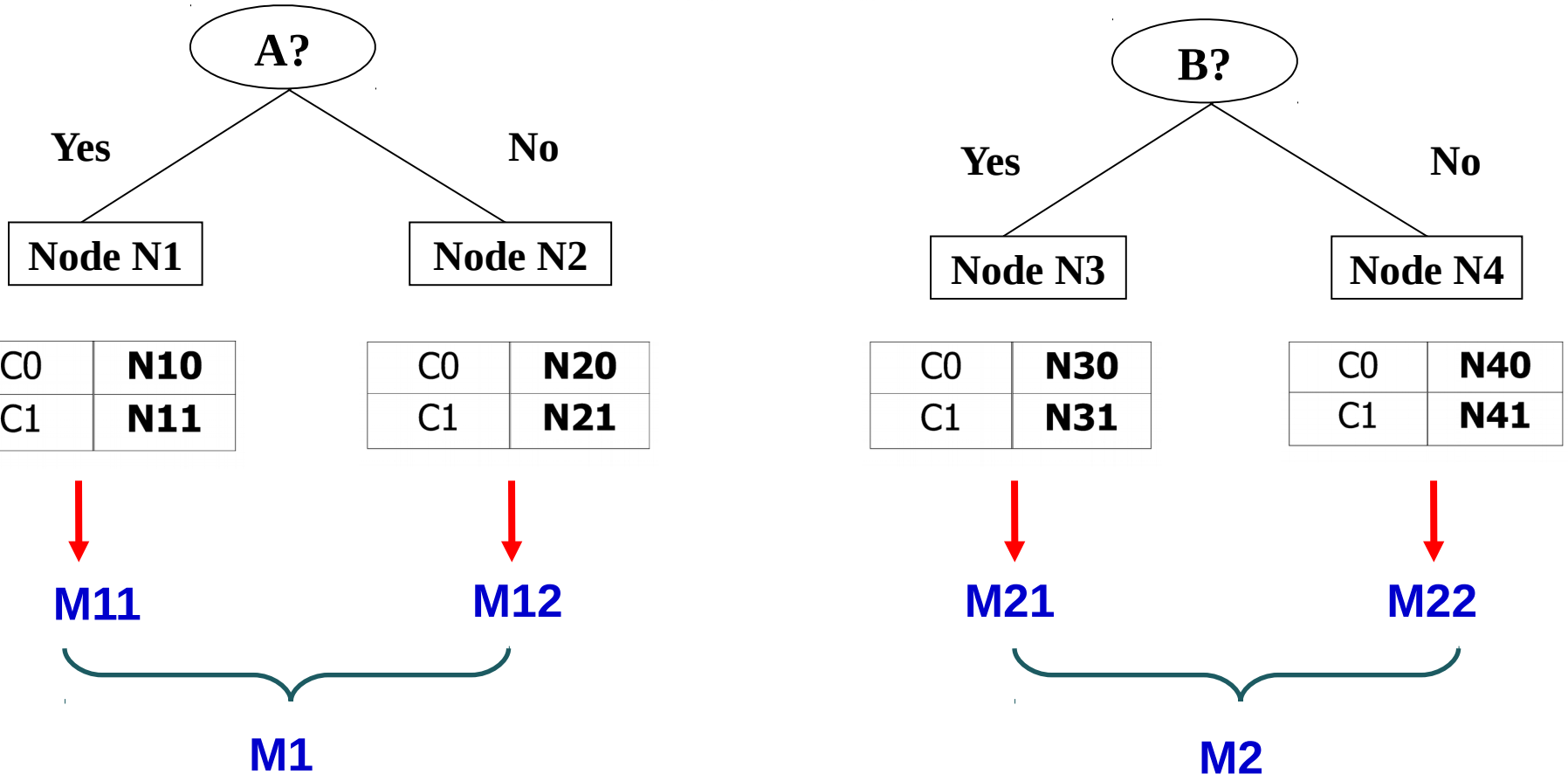
Ganho = P – M

Encontrando a melhor divisão

Antes da divisão:

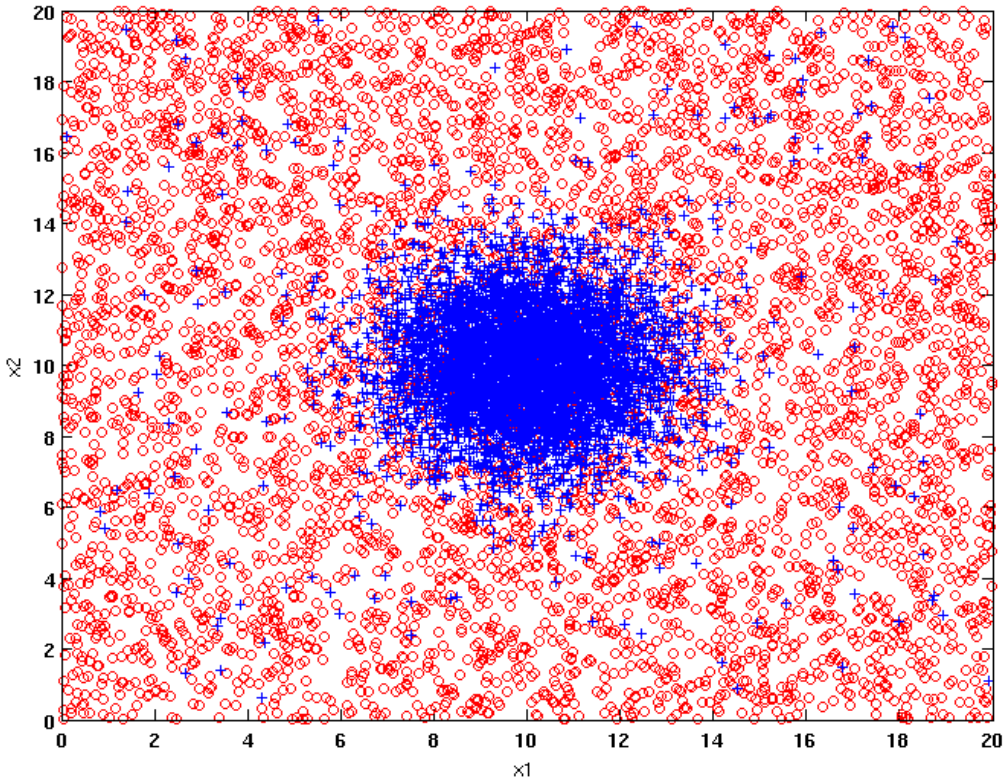
C0	**N00**
C1	**N01**

→ **P**



Ganho = P – M1 vs P – M2

Exemplo



Problema com duas classes:

+ : 5200 instances

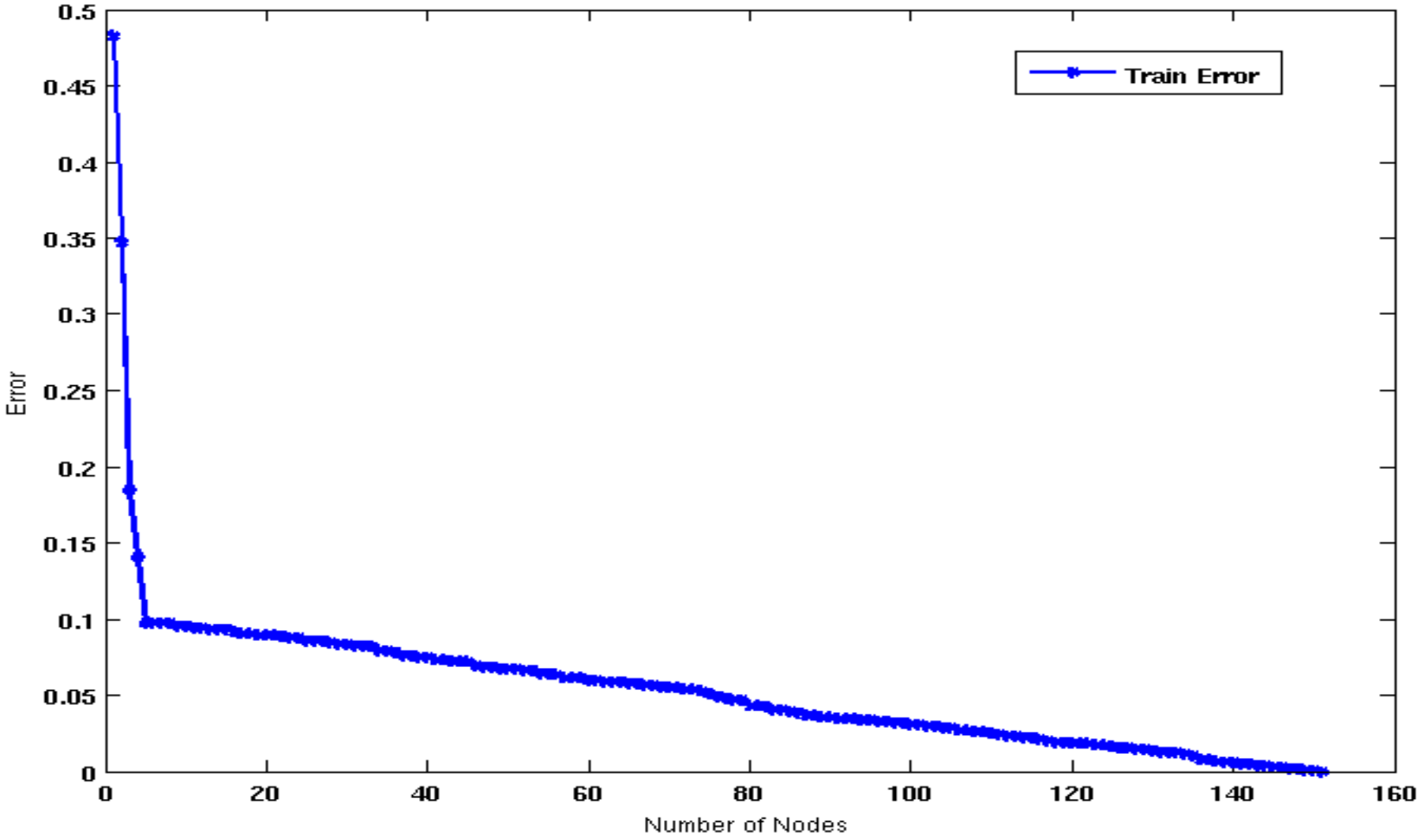
- **5000 instances generated from a Gaussian centered at (10,10)**
- **200 noisy instances added**

o : 5200 instances

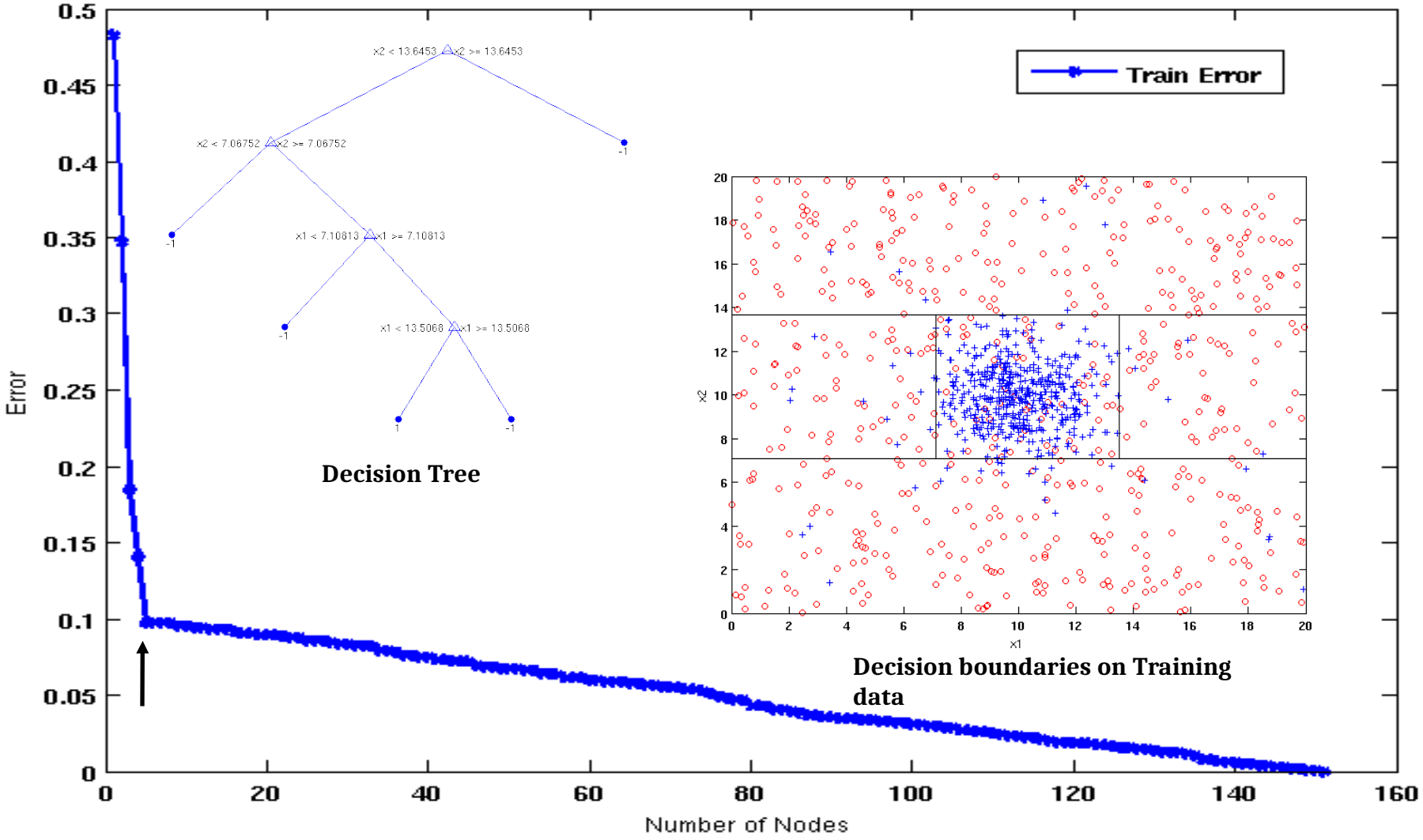
- **Generated from a uniform distribution**

10 % para treino e 90% para teste

Aumentando o número de nós



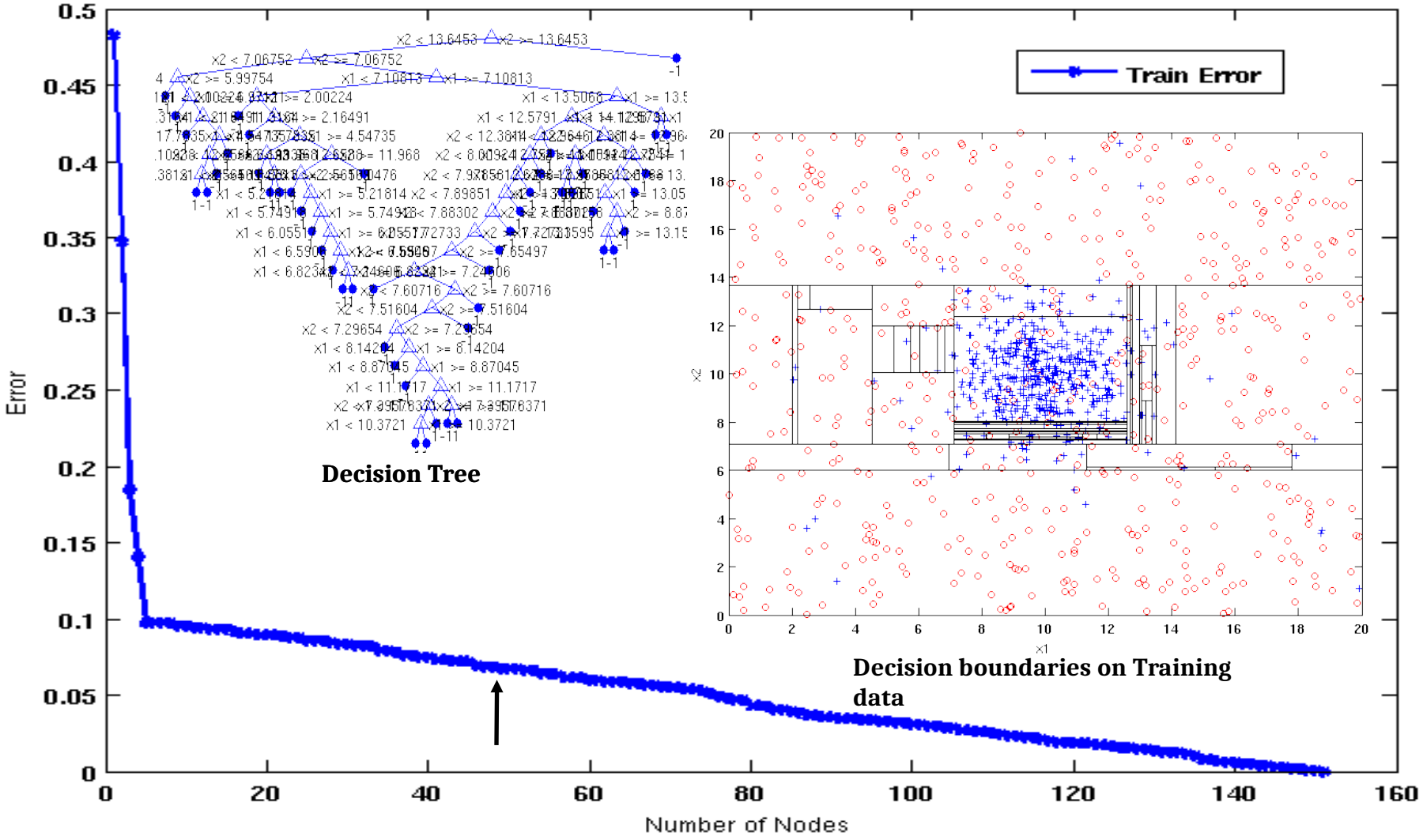
Árvore com 4 nós



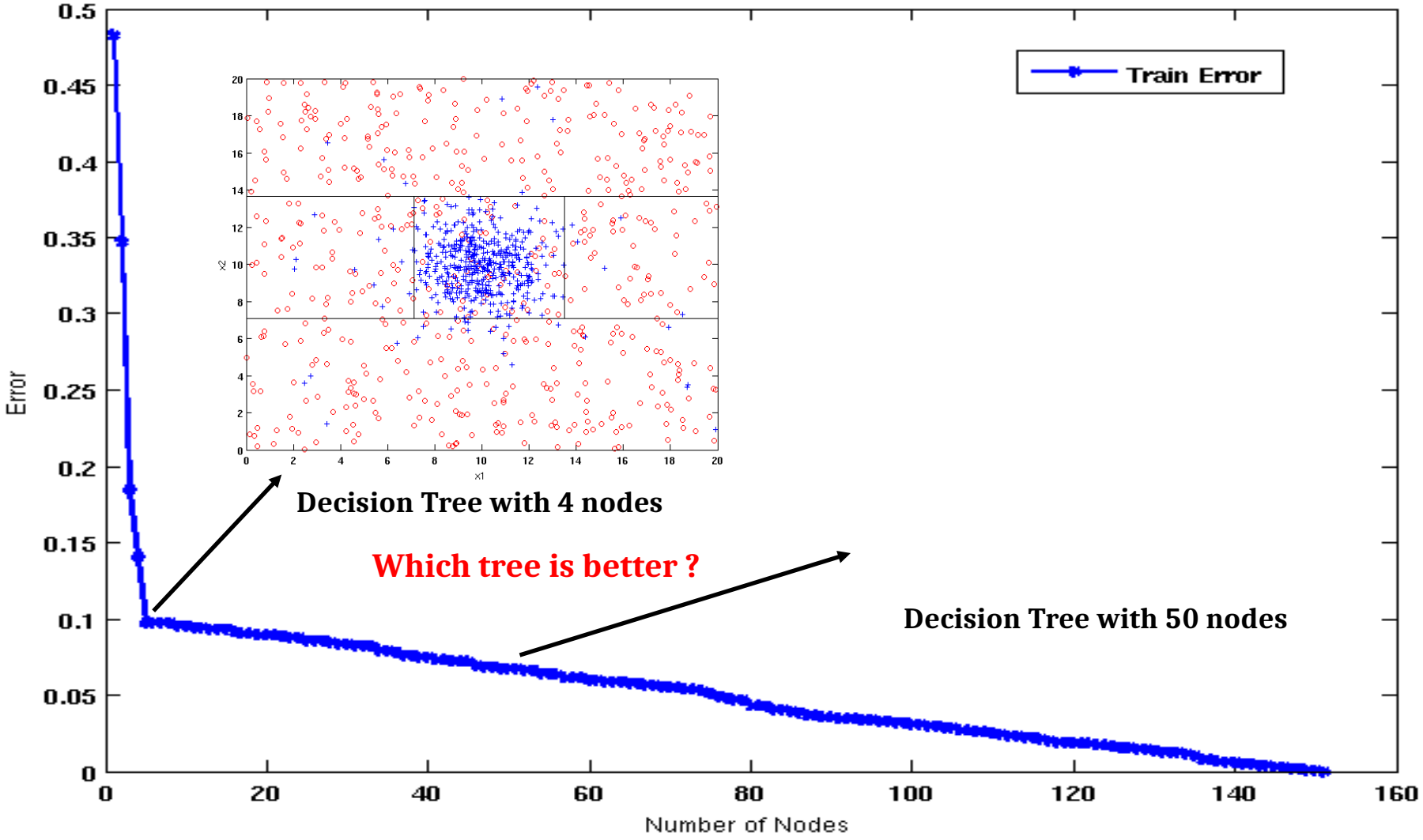
Decision Tree

Decision boundaries on Training data

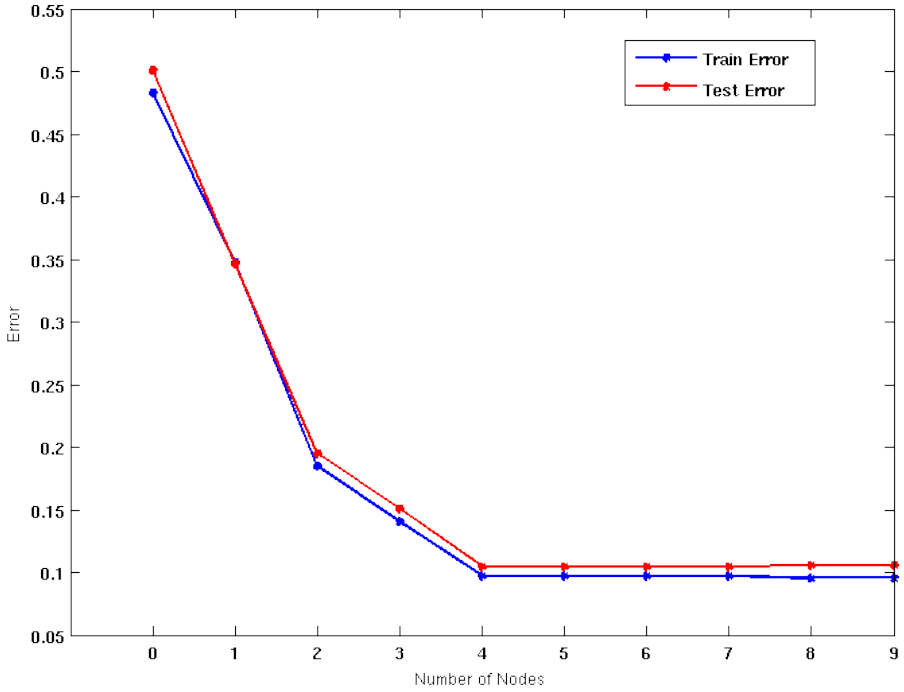
Árvore com 50 nós

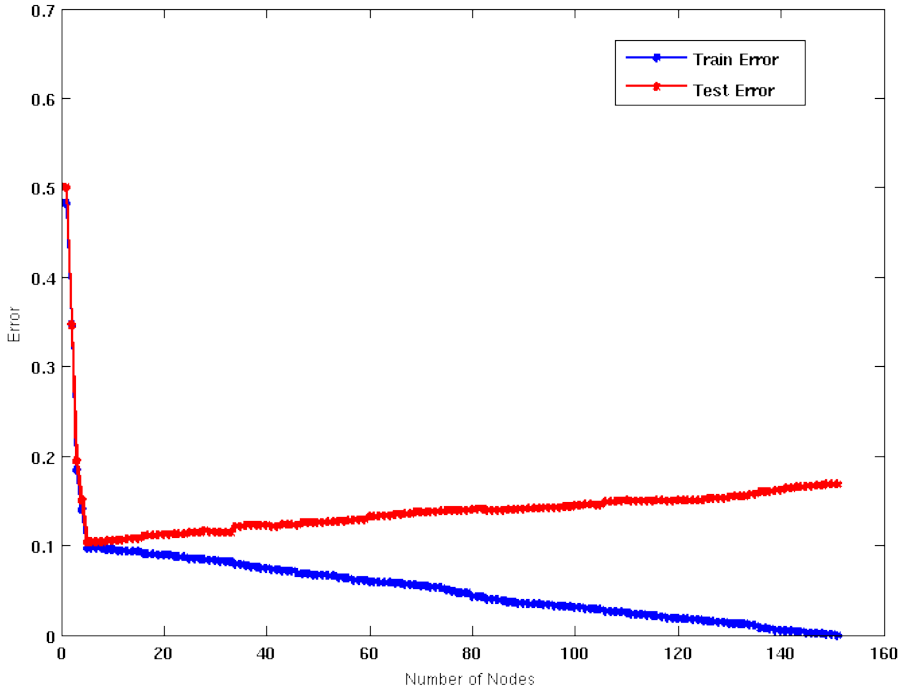


Qual a melhor?



Model Overfitting

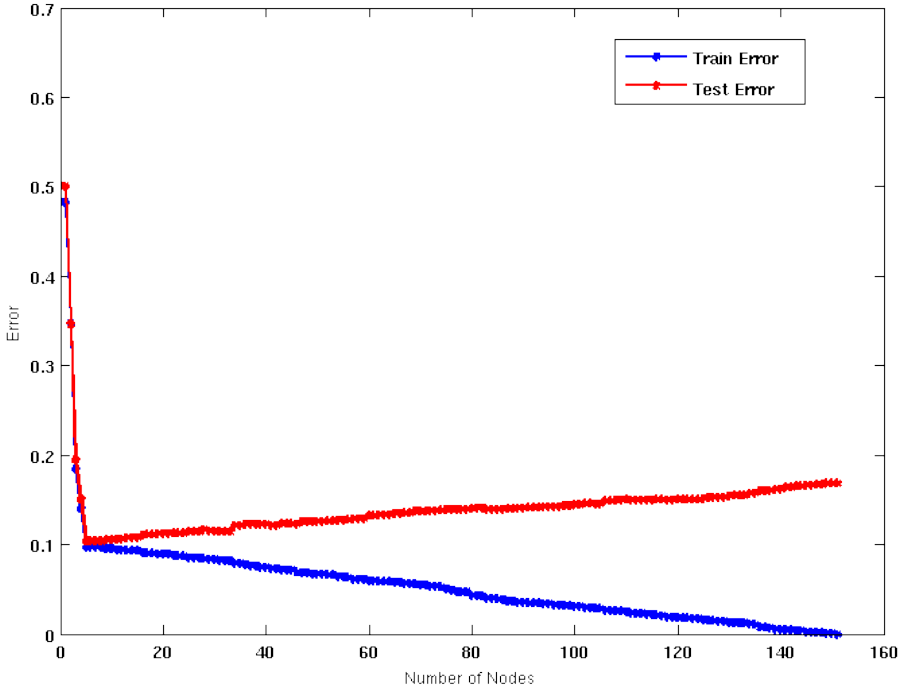


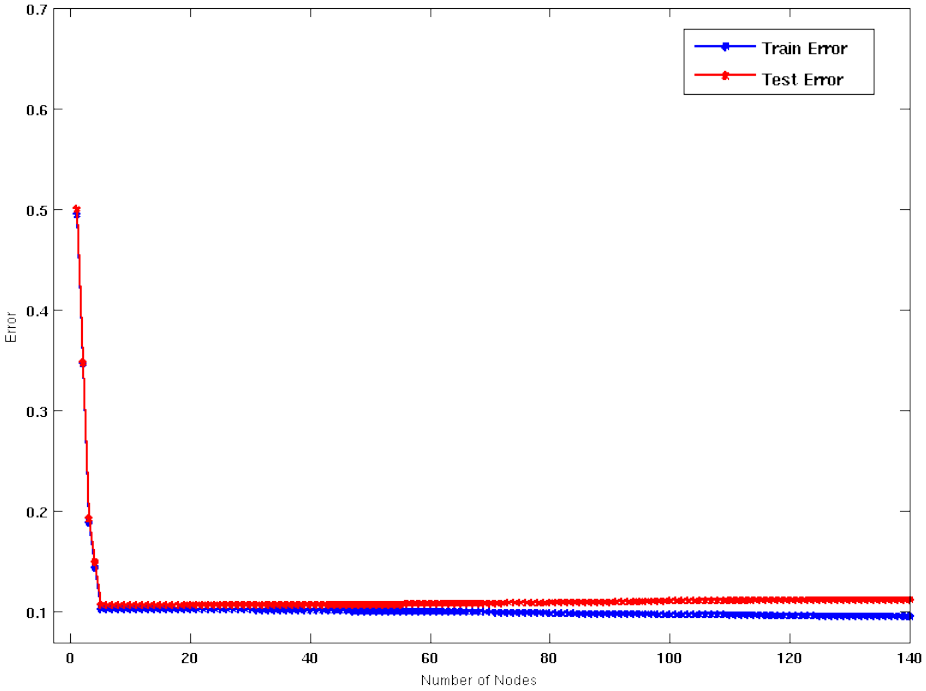


Underfitting: o modelo é muito simples, os erros de treino e teste são altos

Overfitting: o modelo é muito complexo, o erro de treino é pequeno, mas o erro de teste é alto

Model Overfitting





Usando o dobro de instâncias de treino

- **Se o conjunto de treino é subrepresentado, erros de teste aumentam e erros de treino diminuem ao aumentar os nós**
- **Aumentando o tamanho do conjunto de treino reduz a diferença entre os erros de teste e treino em um determinado número de nós**

Métricas de avaliação

Exemplo : acc(M) = 90%

YES = tem-câncer (4 pacientes)
NO = não-tem-câncer (500 pacientes)

Classes “não-balanceadas”

- Classificou corretamente **454** pacientes que não tem câncer
- Não acertou nenhum dos que tem câncer

Bom classificador?

acc = # classificação correta / total de classificações

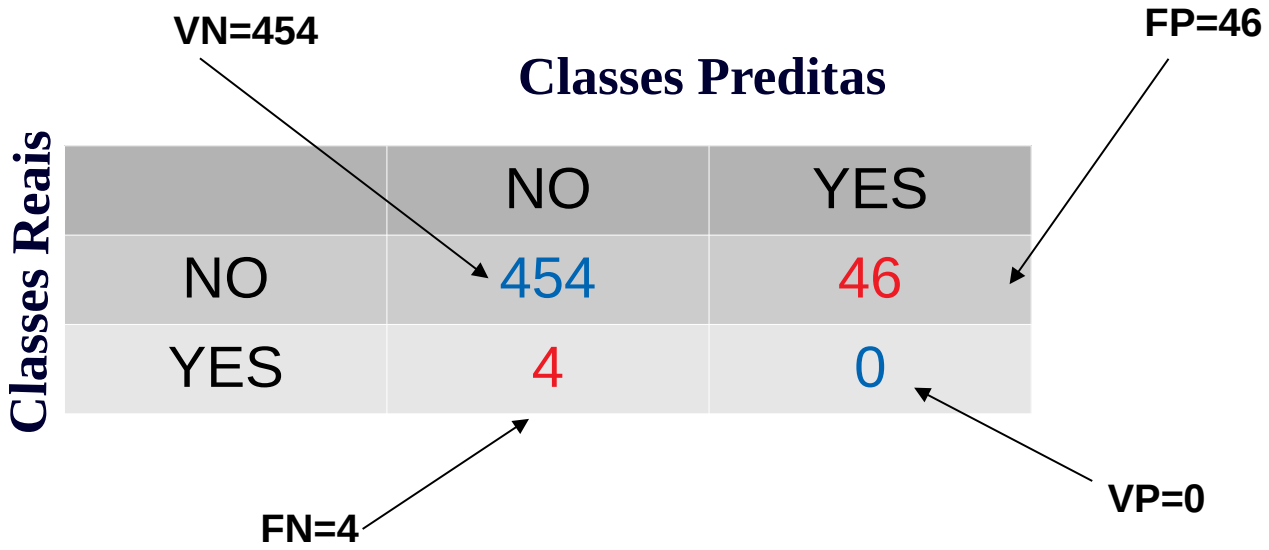
Métricas de avaliação

Exemplo : acc(M) = 90%

Classes "não-balanceadas"

YES = tem-câncer (4 pacientes)
NO = não-tem-câncer (500 pacientes)

- Classificou corretamente **454** pacientes que não tem câncer
- Não acertou nenhum dos que tem câncer



Métricas de avaliação

(VN=454)
(VP=0)
FP=4
FN=6

Sensitividade/Recall $= \frac{VP}{VP + FN}$

% pacientes classificados corretamente como positivos dentre todos os que realmente são positivos

Precisão $= \frac{VP}{VP + FP}$

% pacientes classificados corretamente com câncer dentre todos os que foram classificados com câncer

Precisão e Recall: medidas originadas em *Recuperação de Informação* utilizadas em classificação, quando se lida com "classes não-balanceadas"

Exemplo prático 1

- Dataset Iris
  - Três tipos de plantas (classes):
    - Setosa
    - Versicolour
    - Virginica
  - Quatro atributos
    - Sépala: largura e comprimento
    - Pétalas: largura e comprimento

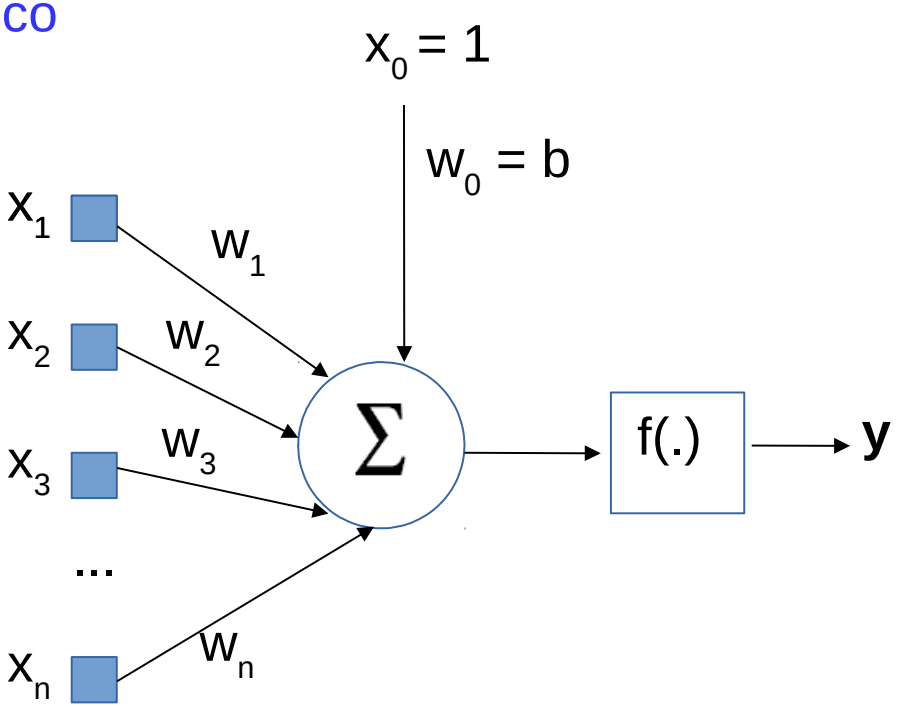
Courtesy of USDA NRCS Wetland Science Institute.

Exemplo prático 1

arvoreSimples.r

Redes Neurais Artificiais

Simula o modelo biológico



Neurônio artificial do tipo perceptron

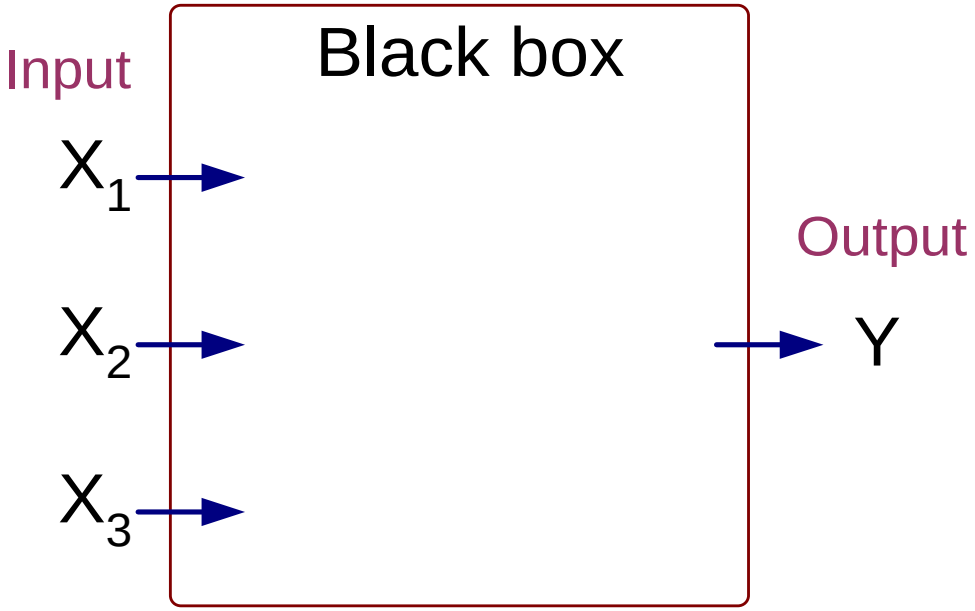
$\vec{\mathbf{x}}$ =entrada (dendrito) e $\vec{\mathbf{w}}$ = peso sináptico

função somatória = corpo celular

f(.) = função de ativação que gera a saída no axônio **y**

Artificial Neural Networks (ANN)

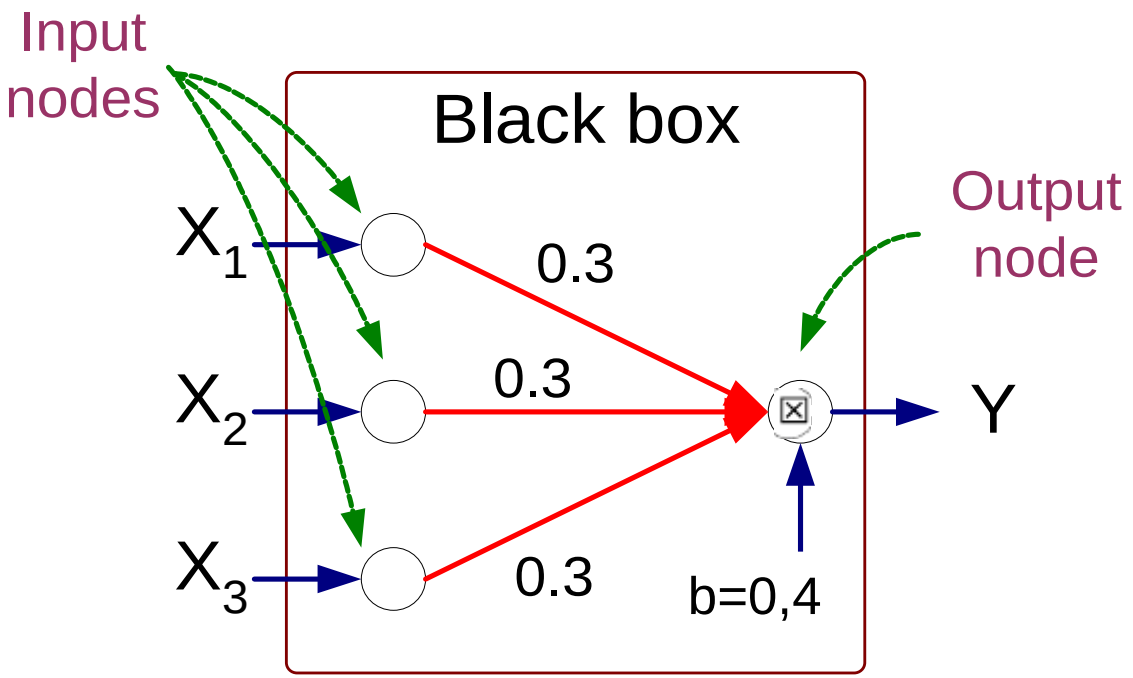
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Saída Y é 1 se pelo menos duas das três entradas são iguais a 1

Artificial Neural Networks (ANN)

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



$$Y = \text{degrau}(0{,}3X_1 + 0{,}3X_2 + 0{,}3X_3 - 0.4)$$

$$\text{onde degrau}(v) = \begin{cases} 1 \text{ se } v >= 0 \\ -1 \text{ se } v < 0 \end{cases}$$

Degrau = função degrau

Função lógica AND

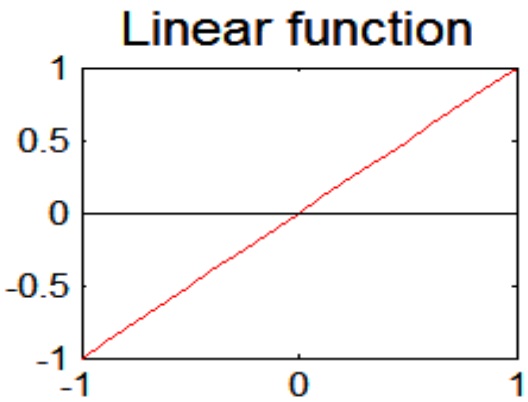
Tabela da verdade

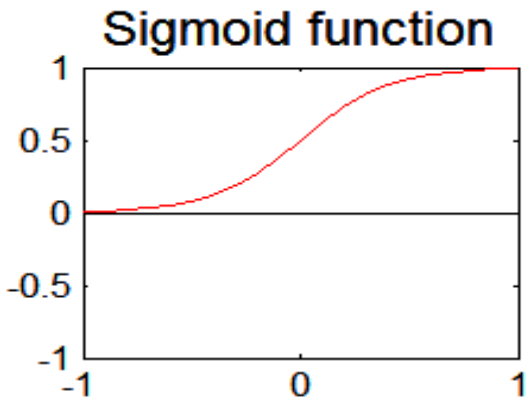
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

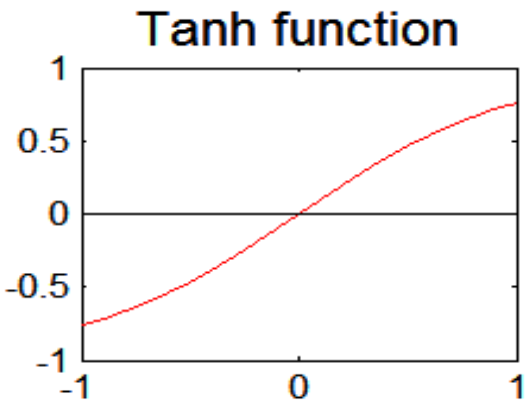
O neurônio projetado para resolver o problema deve:

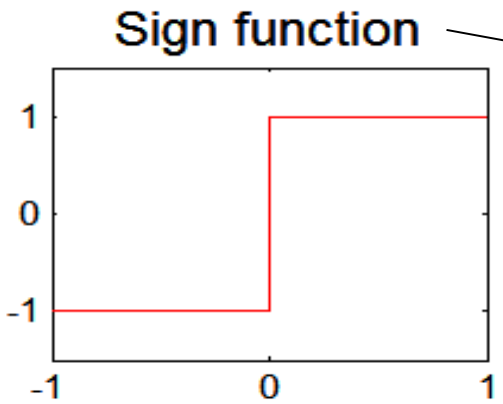
- Utilizar como entrada x_1 e x_2
- Ponderar as entradas com os pesos sinápticos
- Realizar um somatório
- Aplicar uma função de ativação para produzir uma saída $\hat{y}$ que deve ser igual à y

Vários tipos de funções de ativação









Função degrau

Função lógica AND

Para o exemplo definimos a função sign (degrau):

- Ela produzirá saída **1** quando o campo induzido for > 0 e **-1** caso contrário

A regra de processamento do neurônio pode ser definida como:

- cálculo do sinal que entra no neurônio *o é*

$$v_o = \sum_{i=1}^{n} (x_i * w_{oi}) + b_o$$

- x_i é o valor de entrada
- a saída do neurônio é $y_0 = f(v_o)$

Função lógica AND (aprendizado)

Inicializar os pesos (w_0, w_1, …, w_n)

Repetir

Para cada instância de treino (x_i, y_i)

Computar f(w, x_i)

Atualizar os pesos:

Taxa de aprendizado

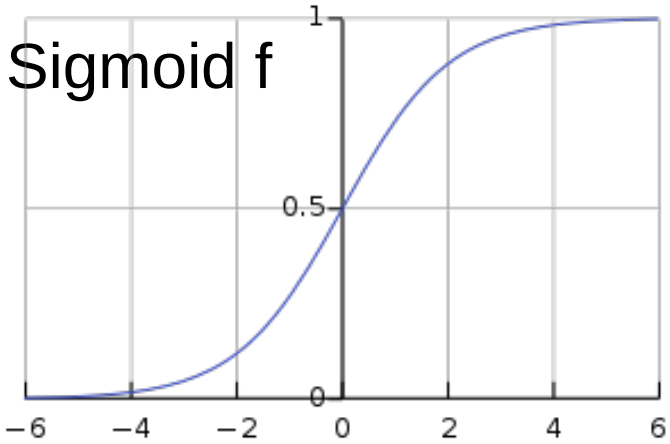
Erro

$$w^{(k+1)} = w^{(k)} + \lambda \left[y_i - f(w^{(k)}, x_i) \right] x_i$$

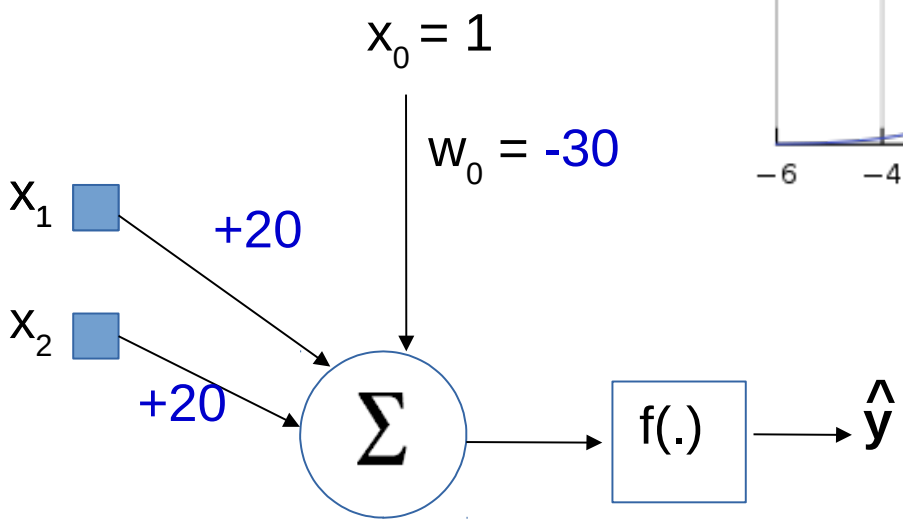
Até que a condição de parada seja atingida

Função lógica AND (aprendizado)

Exemplo com outra função de ativação:



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

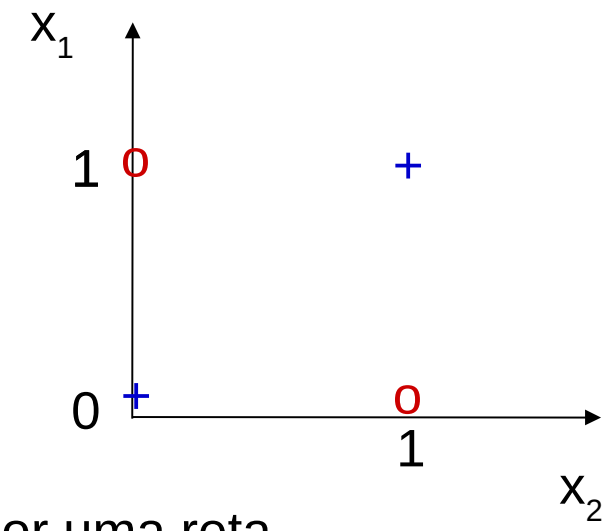


$f(20x_1 + 20x_2 - 30)$

x_1	x_2	$\hat{y}$
0	0	f(-30) ~0
0	1	f(-10) ~0
1	0	f(-10) ~0
1	1	f(10) ~ 1

Função lógica XOR

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

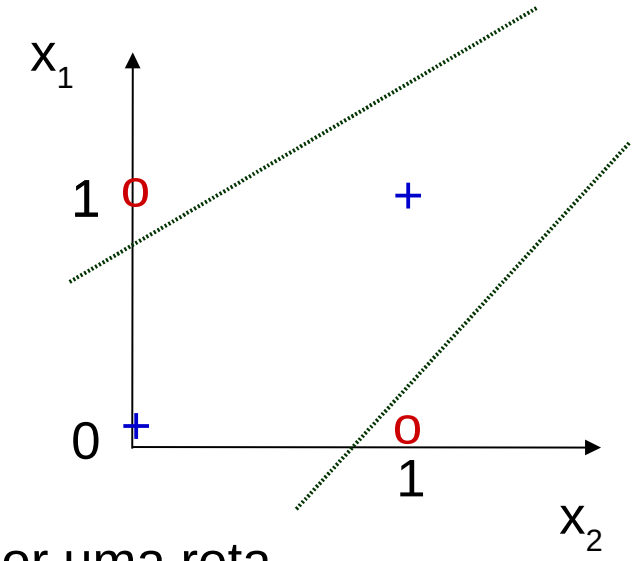


Não é linearmente separável

Ou seja, classes que podem ser separadas por uma reta (ou um hiperplano)

Função lógica XOR

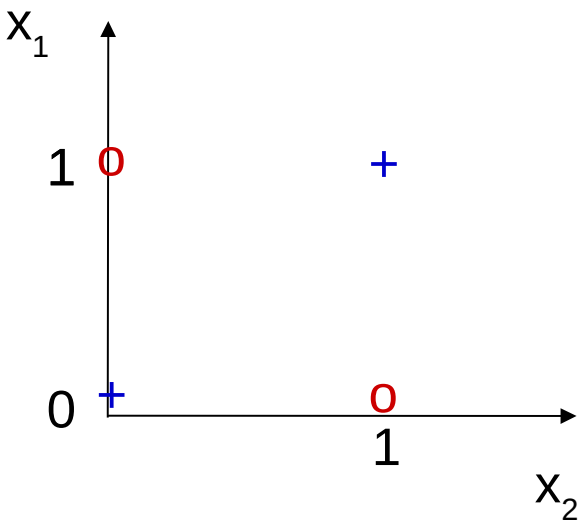
x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

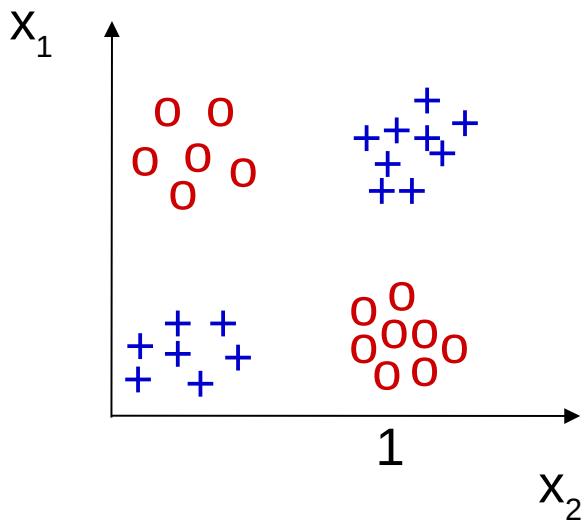


Não é linearmente separável
Ou seja, classes que podem ser separadas por uma reta (ou um hiperplano)

Precisamos combinar mais de um neurônio (possibilita combinar retas)

Função lógica XOR





Problema à esquerda é uma versão simplificada (mais fácil de analisar) do problema a direita

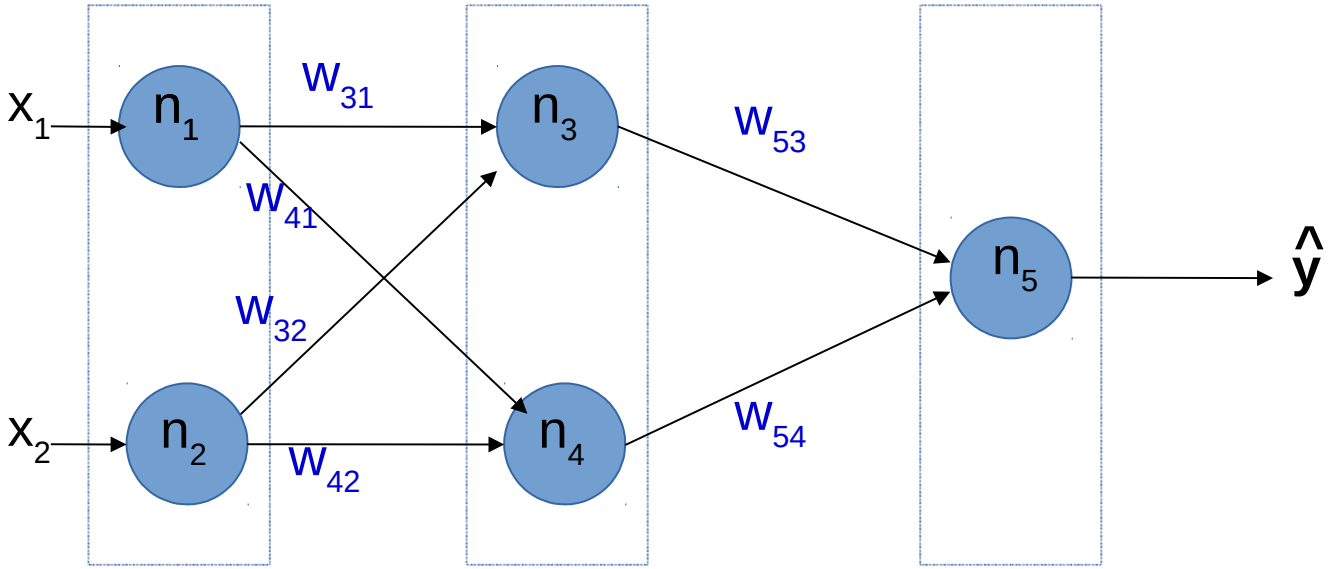
Rede neural de múltiplas camadas

Possibilidade de combinação de neurônios em uma rede de múltiplas camadas

Possibilita a obtenção de estruturas mais complexas

Pode ser útil na resolução de tarefas que envolvem superfícies de decisão não-lineares

Rede neural de múltiplas camadas

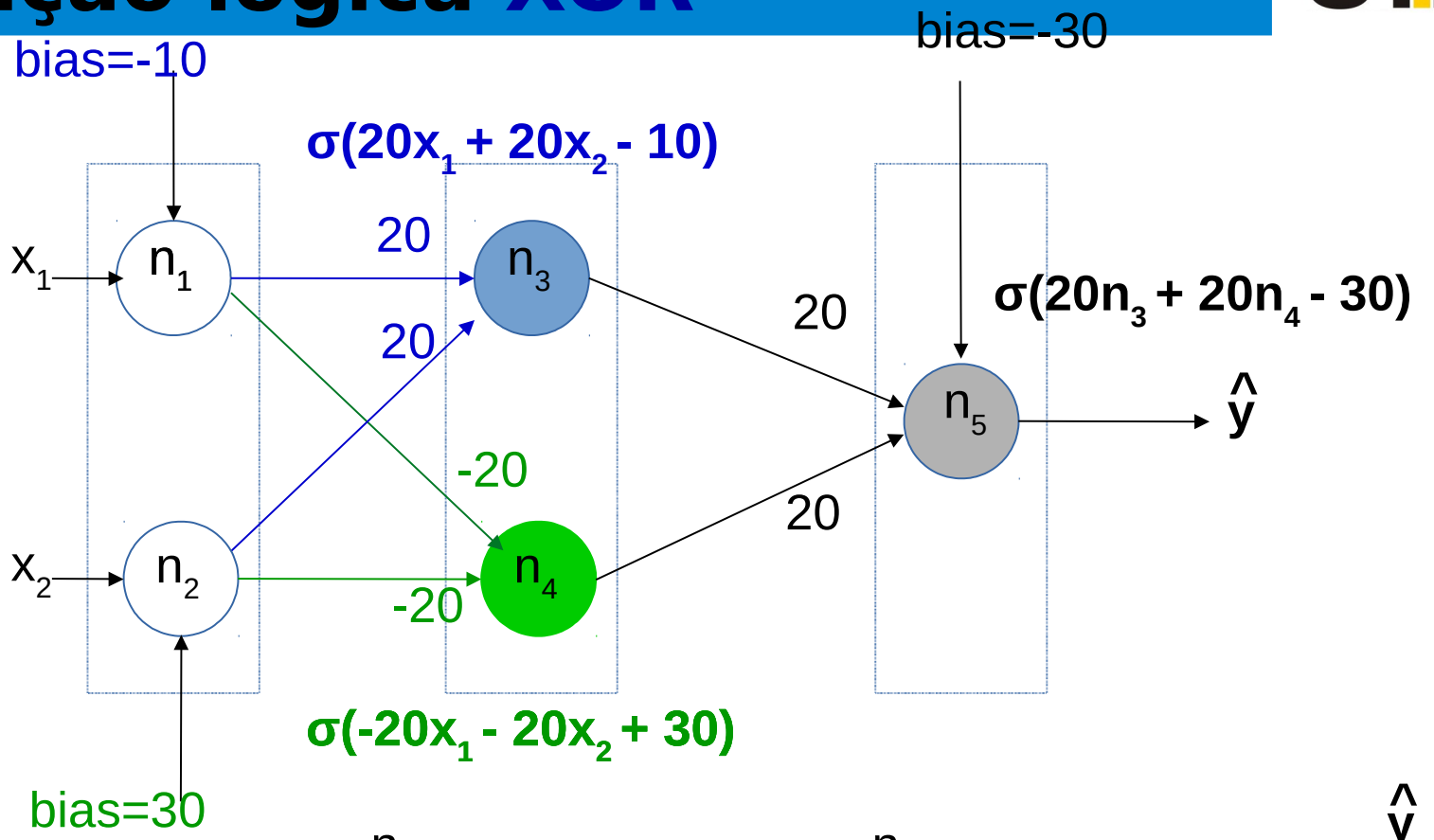


Camada de entrada

Camada escondida

Camada de saída

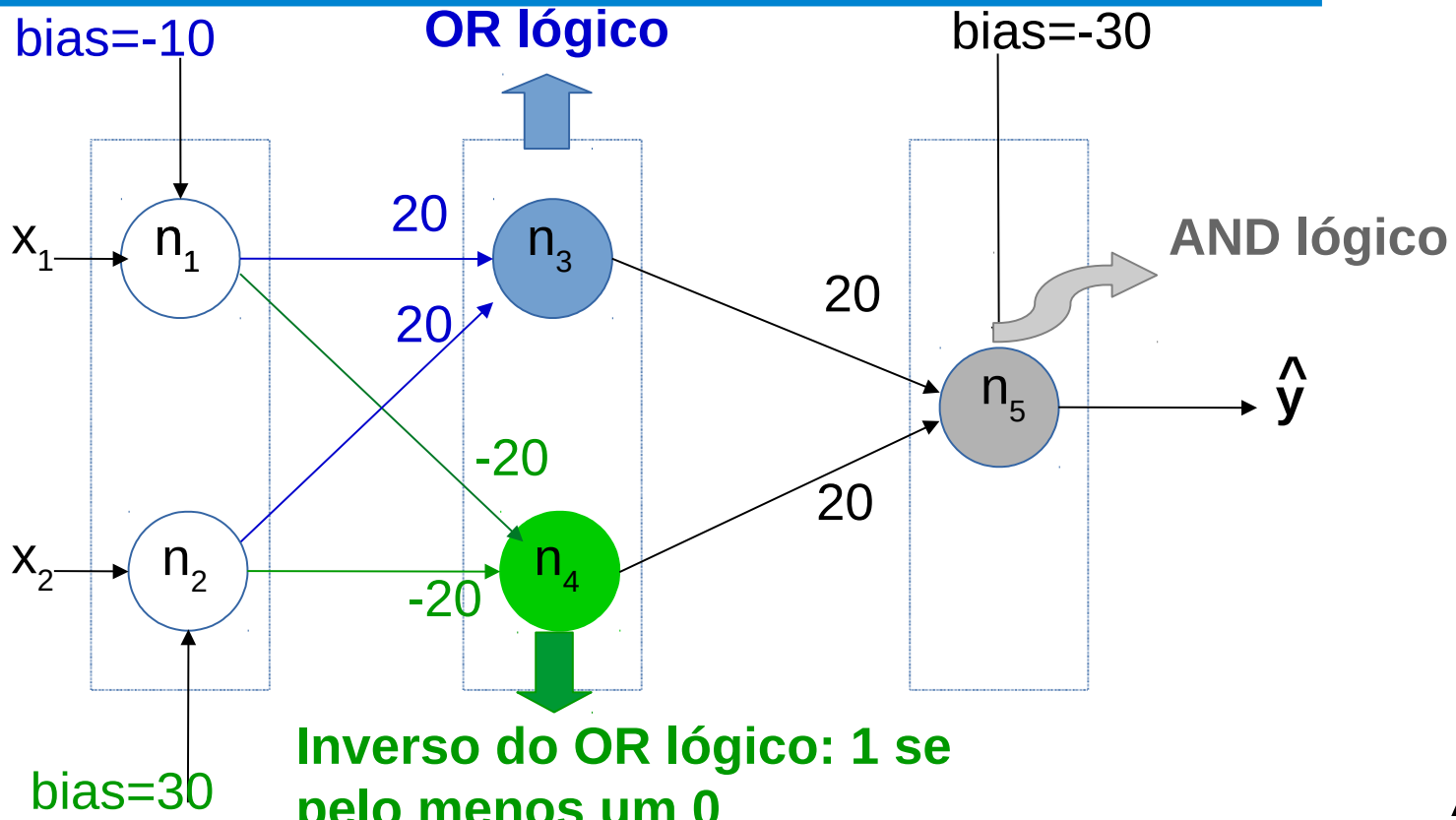
Função lógica XOR



x_1	x_2
0	0
0	1
1	0
1	1

	n_3	n_4	$\hat{y}$
0 0	σ(20*0 + 20*0 – 10) ~ 0	σ(-20*0 - 20*0 + 30)~1	(20*0 + 20*1 – 30)~ 0
0 1	σ(20*0 + 20*1 – 10) ~ 1	σ(-20*0 - 20*1 + 30)~1	(20*1 + 20*1 – 30)~ 1
1 0	σ(20*1 + 20*0 – 10) ~ 1	σ(-20*1 - 20*0 + 30)~1	(20*1 + 20*1 – 30)~ 1
1 1	σ(20*1 + 20*1 – 10) ~ 1	σ(-20*1 - 20*1 + 30)~0	(20*1 + 20*0 – 30)~ 0

Função lógica XOR



x_1	x_2
0	0
0	1
1	0
1	1

n_3 n_4 $\hat{y}$

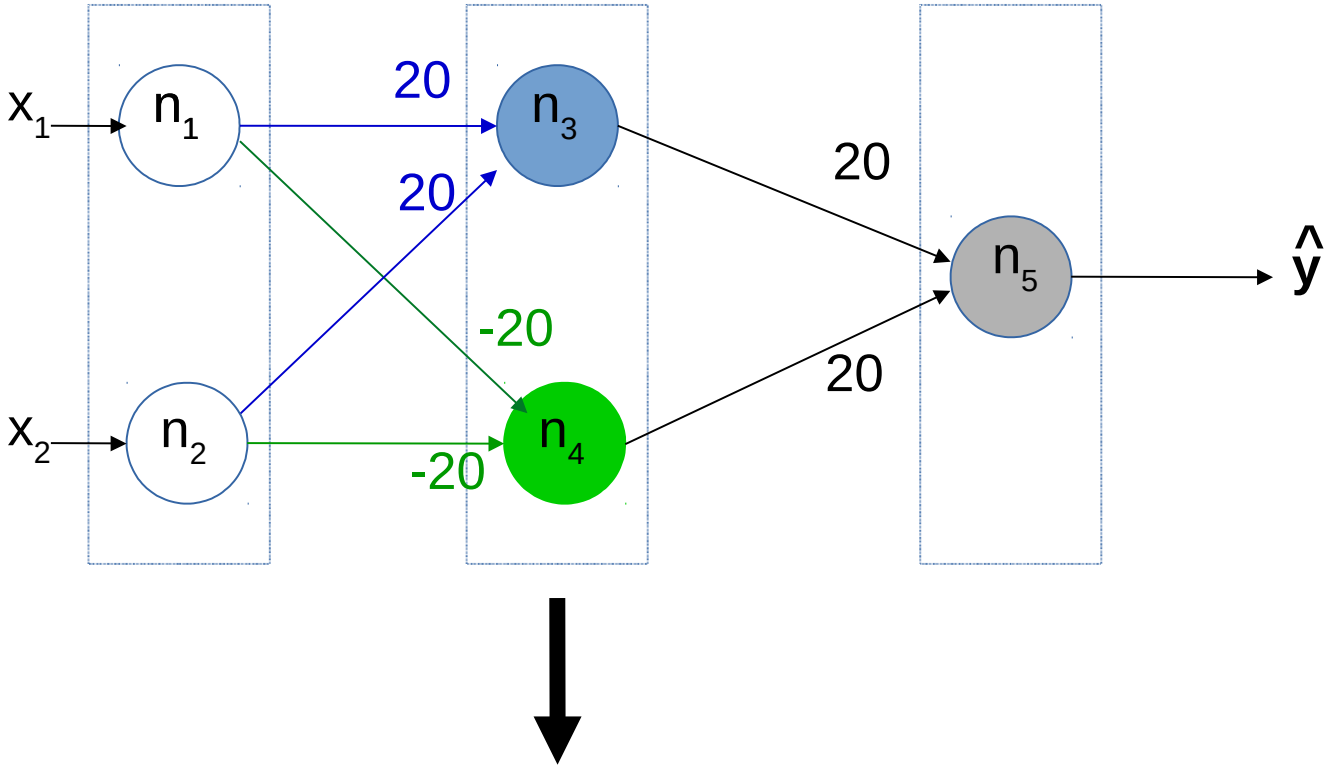
$\sigma(20*0 + 20*0 - 10) \sim 0$ $\sigma(-20*0 - 20*0 + 30) \sim 1$ $(20*0 + 20*1 - 30) \sim 0$

$\sigma(20*0 + 20*1 - 10) \sim 1$ $\sigma(-20*0 - 20*1 + 30) \sim 1$ $(20*1 + 20*1 - 30) \sim 1$

$\sigma(20*1 + 20*0 - 10) \sim 1$ $\sigma(-20*1 - 20*0 + 30) \sim 1$ $(20*1 + 20*1 - 30) \sim 1$

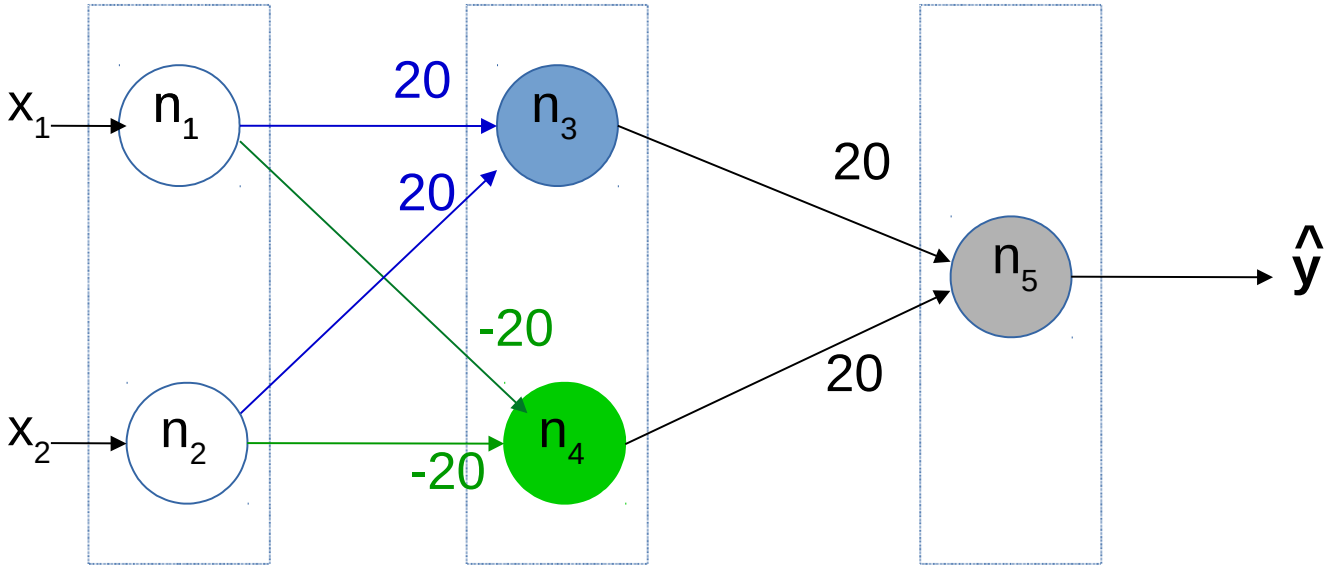
$\sigma(20*1 + 20*1 - 10) \sim 1$ $\sigma(-20*1 - 20*1 + 30) \sim 0$ $(20*1 + 20*0 - 30) \sim 0$

Função lógica XOR



Não há como calcular o erro nos neurônios da camada escondida de forma direta, pois não existe a resposta deseja para tais neurônios.

Função lógica XOR



x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

Erro calculado

Erro propagado

Pesos reajustados levemente

O processo é repetido para todos as entradas e saídas até que o erro seja pequeno ou outra condição imposta.
Após esse processo a rede é considerada treinada

Vantagens e Desvantagens

- Fase de treinamento demorada. As iterações no *backpropagation* podem ocorrer centenas de milhares de vezes.
- Fraca interpretabilidade
- Muitos parâmetros determinados empiricamente

- Elevada tolerância a ruídos
- Resultados tendem a ser bons

Exemplo prático

redeNeuralSimples.r
creditset.csv

Metodos de avaliação

Alguns dos métodos:

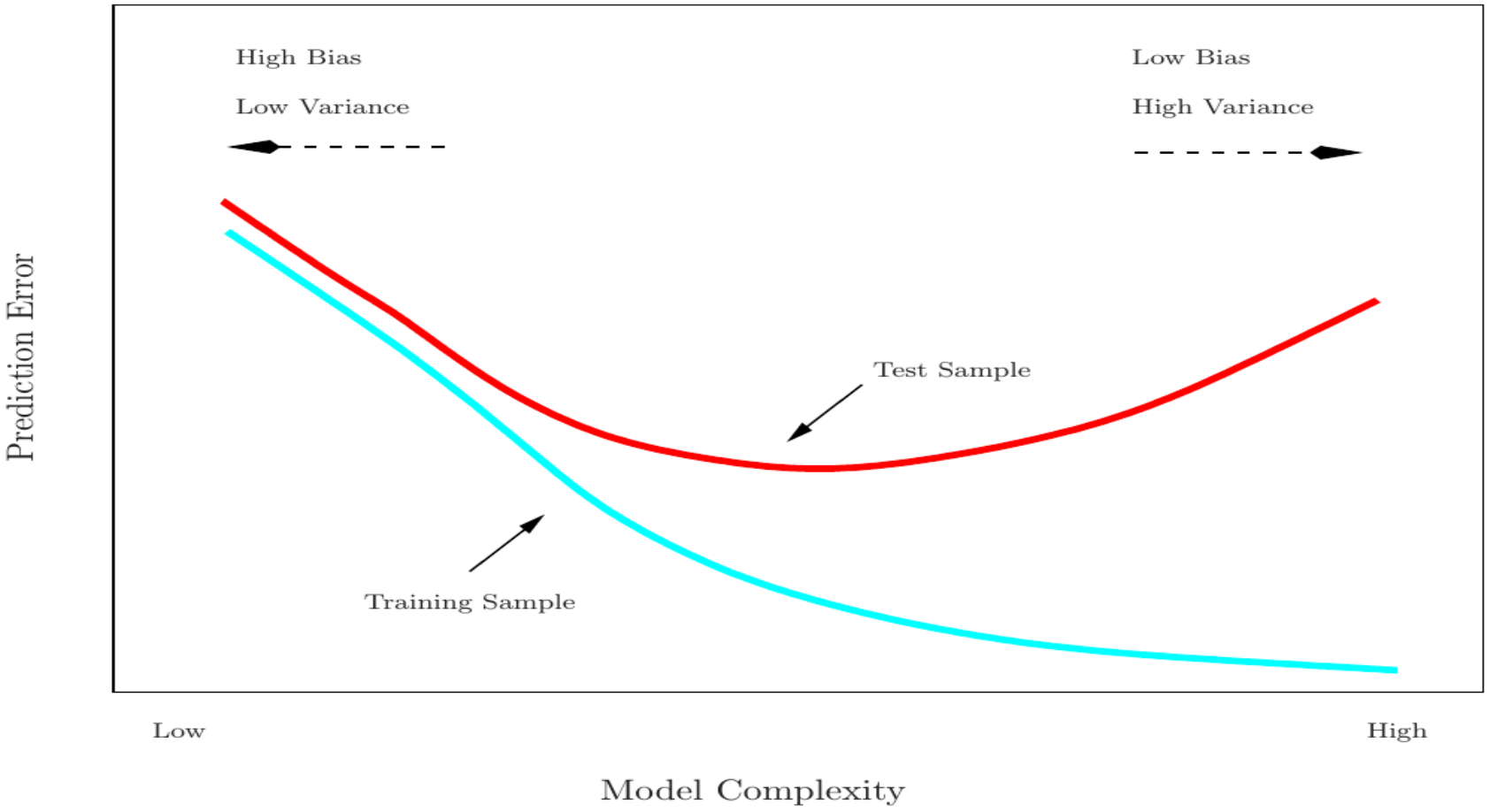
Holdout

Reserva k% para treino (100-k)% para teste

Cross validation

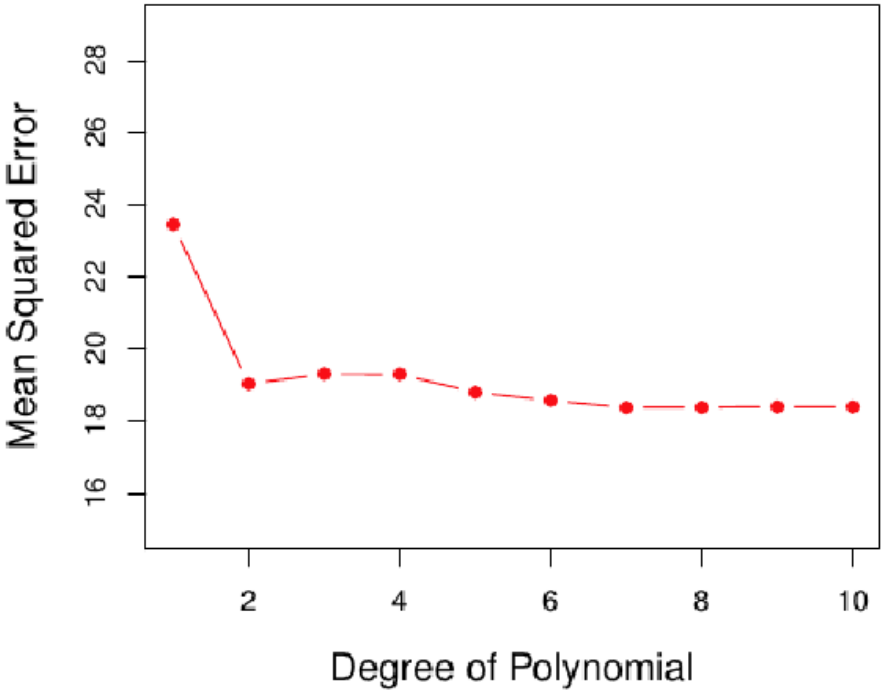
Particionar os dados em k subconjuntos disjuntos
k-fold: treinar em k-1 partições, testar na remanescente

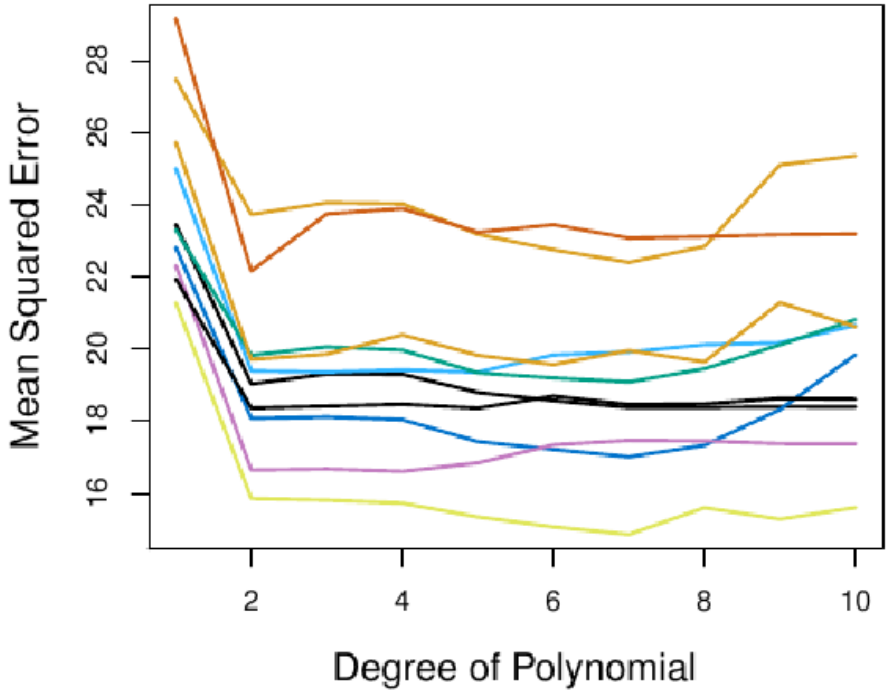
Erro de treino vs erro de teste



Não podemos controlar o erro de teste com o de treino

Divisão em duas partes





- Esquerda: uma única divisão. Direita: várias.
- Variabilidade muito grande.

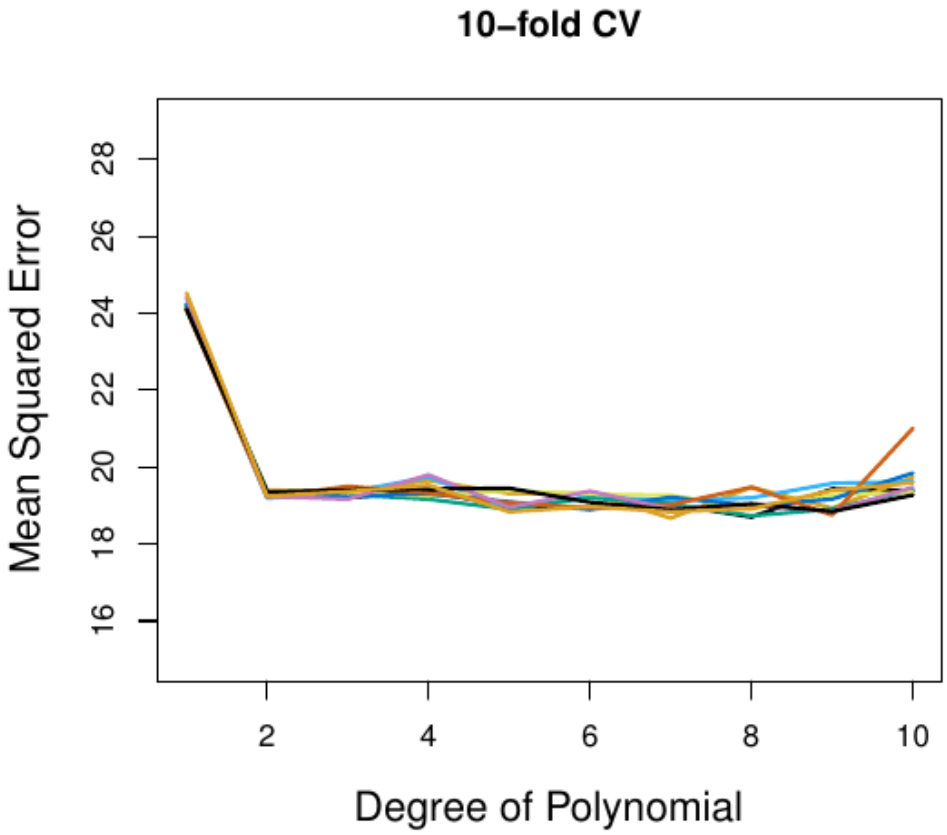
k-folds

- **Divide o dataset em partes iguais**

1	2	3	4	5
Validação	Treino	Treino	Treino	Treino

k-folds

- **Variação do resultado fica menor.**



K= 5 ou 10 são escolhas típicas

Exercício prático

- **Fazer o exercício 3 disponível no moodle.**

Agradecimentos

Alguns slides foram derivados/inspirados em:

- Livro Introduction to Data Mining - Tan, Steinbach, Kumar. 2018.