U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence/ Inteligência Artificial

Lecture 10: Machine Learning Algorithms

(adaptado de Faria, 2018 e Castillo 2011)

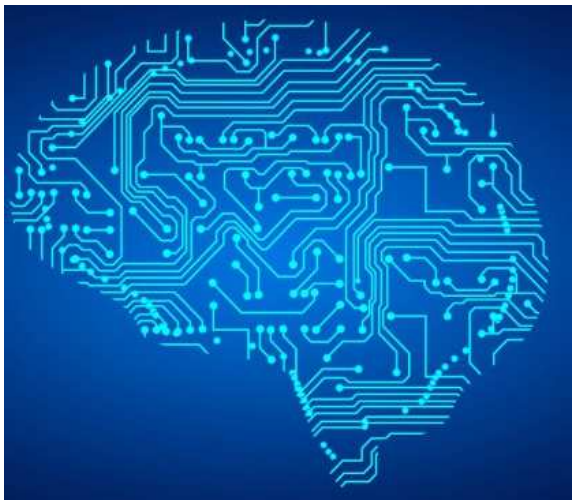
Luís Paulo Reis

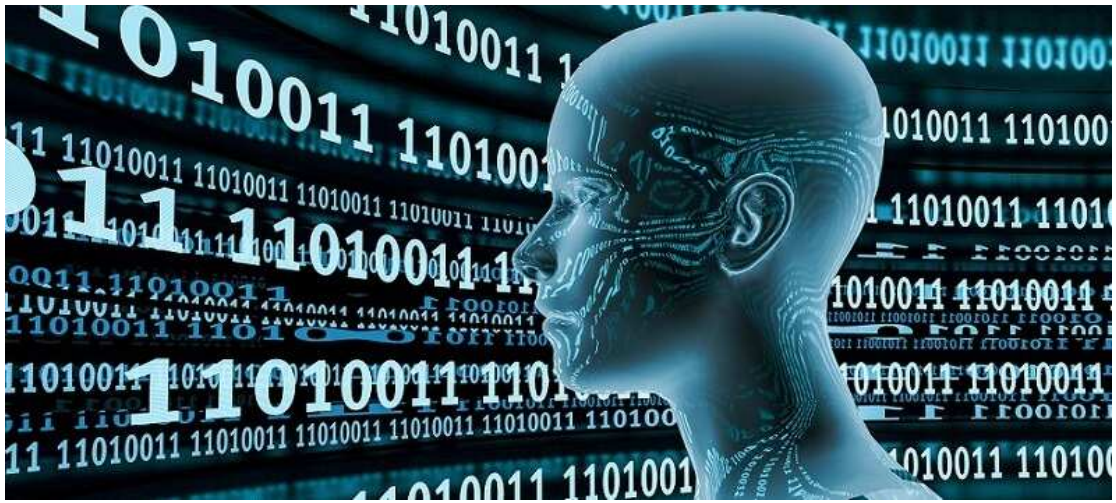
lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence

Machine Learning

- Machine learning is a field of artificial intelligence that gives computer systems the **ability to "learn"** (e.g., progressively **improve performance** on a specific task) from data/results of their actions, without being explicitly programmed

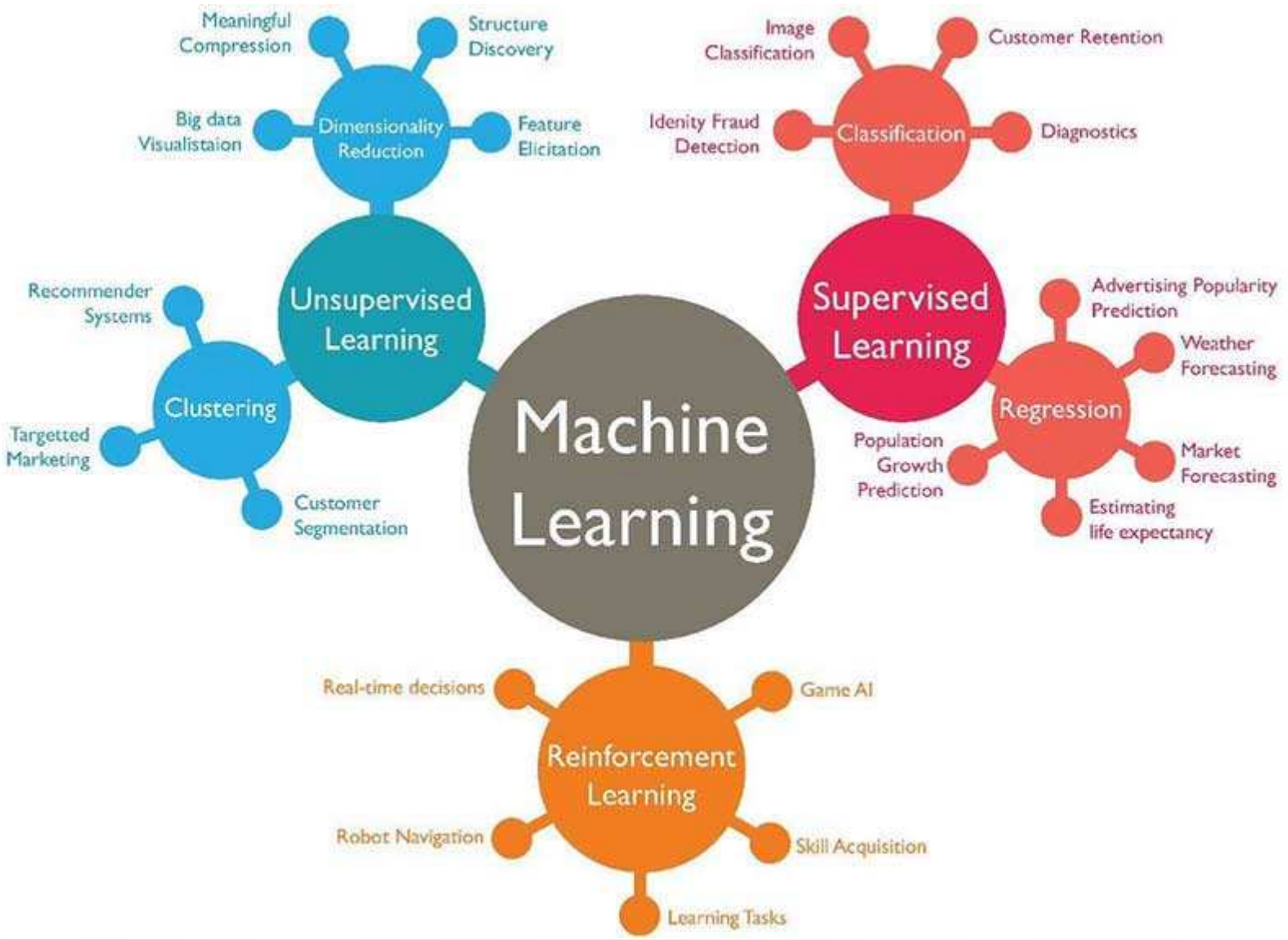




Machine Learning Tasks/Types

- Machine Learning (ML) Tasks/Types:
  - **Supervised learning:** Example inputs and desired outputs are available/given by a "teacher", and the goal is to learn how to map inputs to outputs (possibility semi-supervised)
  - **Reinforcement learning:** Data (in form of rewards and punishments) are given only as feedback to the computer/agent actions in a dynamic environment
  - **Unsupervised learning:** No labels/outputs are given to the learning algorithm, leaving it on its own to find structure in its input

Machine Learning



Métodos de Aprendizagem

- Árvores de Decisão
- Conjunto de Regras
- Baseados em Instâncias
- Bayesiana
- Redes Neuronais
- Máquinas de Suporte Vetorial

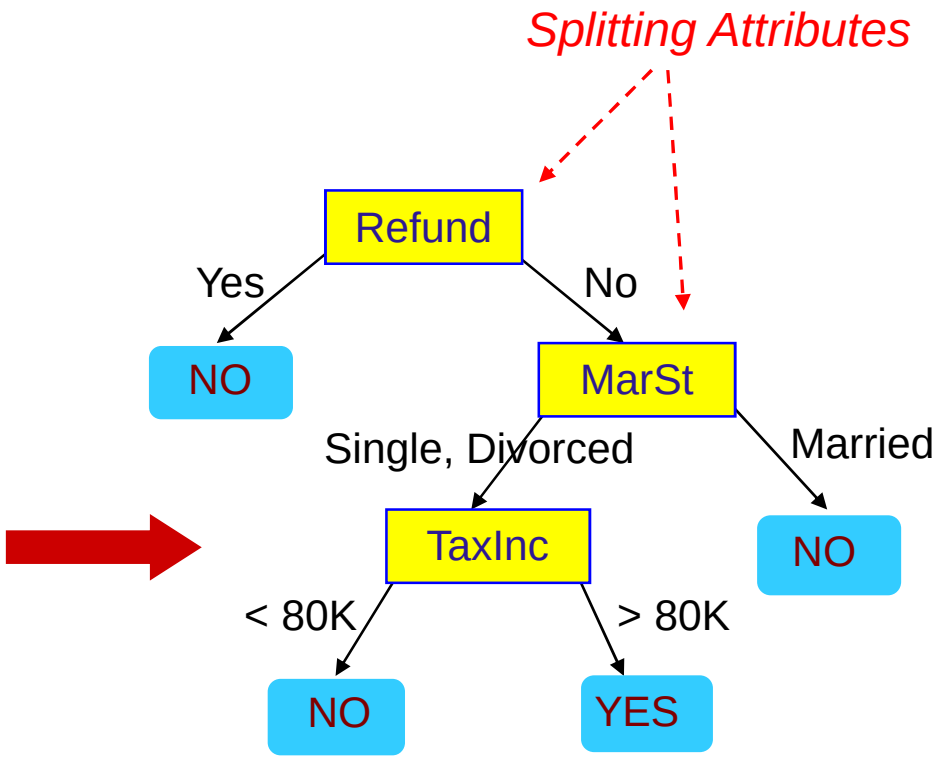
Árvores de Decisão

- **Exemplo de uma Árvore de Decisão**

Dicotómica | Nominal | Contínua | classe

Tid	Refund	Marital Status	Taxable Income	Cheat
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

Conjunto de Treino

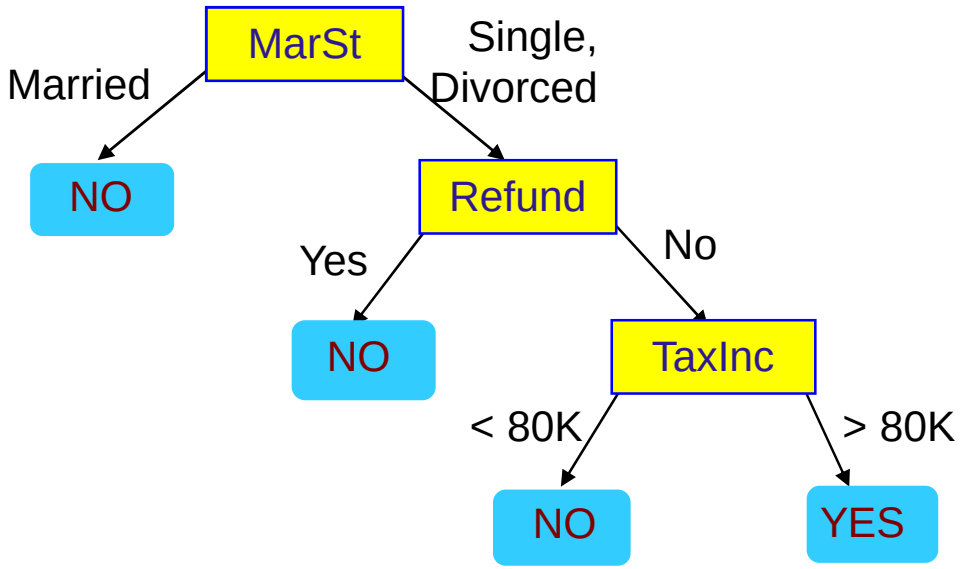


Modelo: Decision Tree

Árvores de Decisão

- **Outro exemplo de uma Árvore de Decisão**

	Dicotómica	Nominal	Contínua	Classe
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
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**

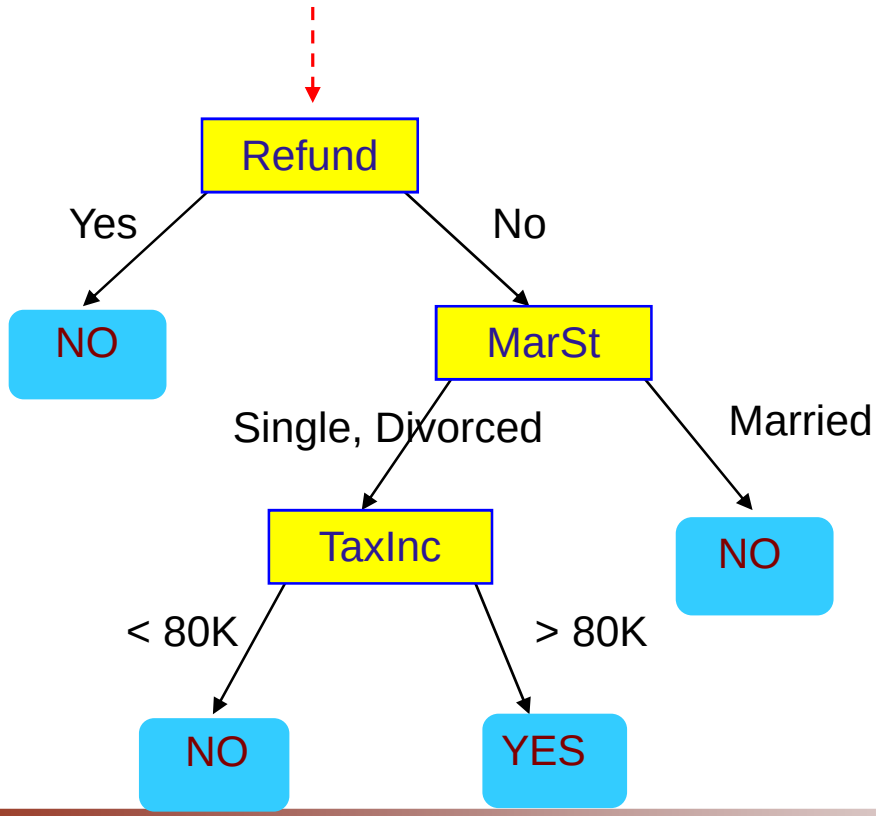


Usando diferentes algoritmos podemos obter diferentes árvores para representar os mesmos dados

Árvores de Decisão

- **Aplicação da árvore de decisão a um novo exemplo**

Começar pela raiz da árvore



Novo exemplo

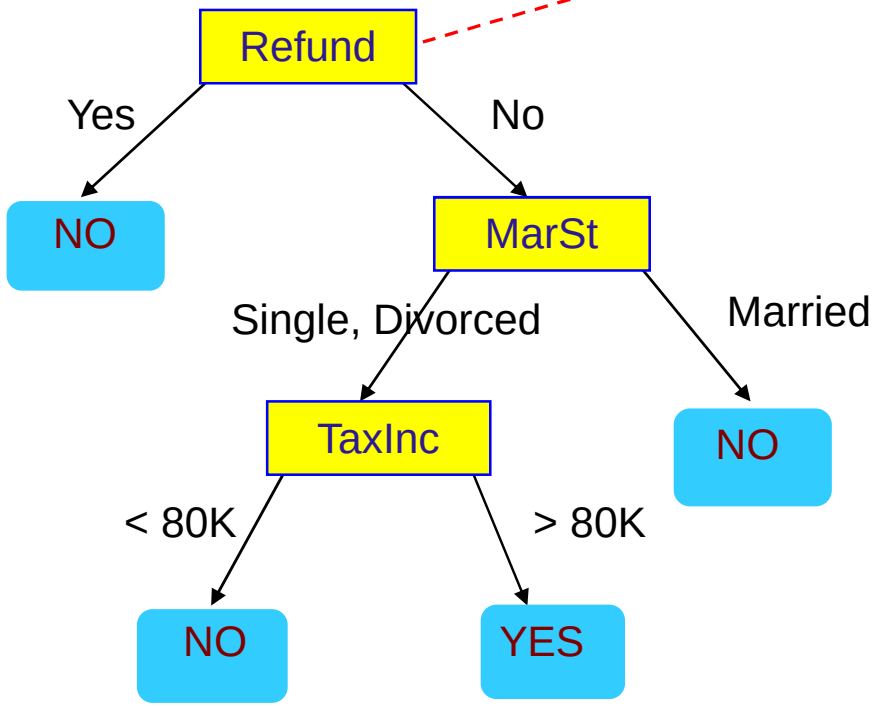
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Árvores de Decisão

- **Aplicação da árvore de decisão a um novo exemplo**

Novo exemplo

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

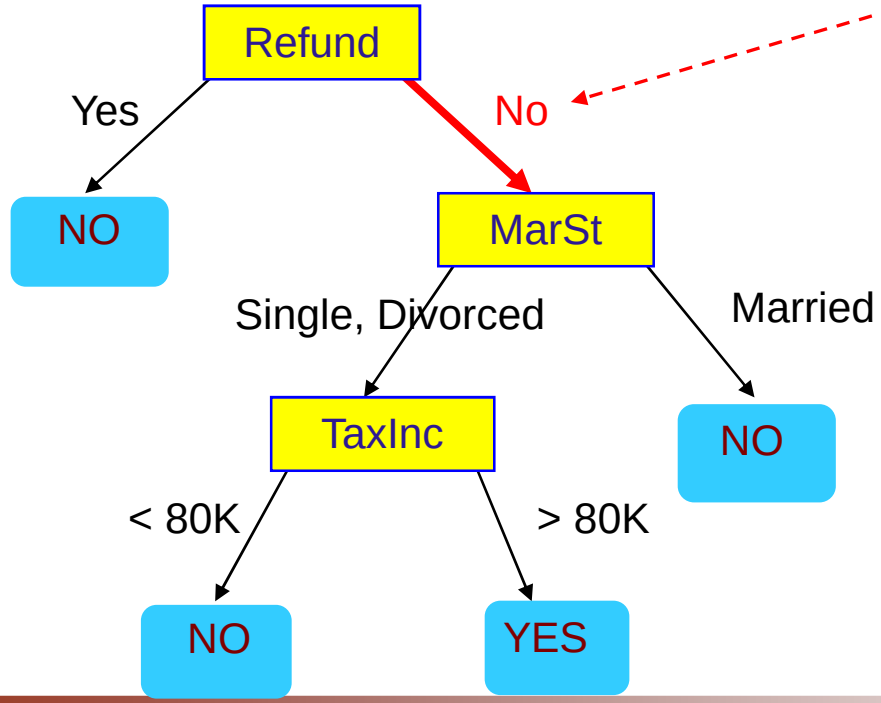


Árvores de Decisão

- **Aplicação da árvore de decisão a um novo exemplo**

Novo exemplo

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

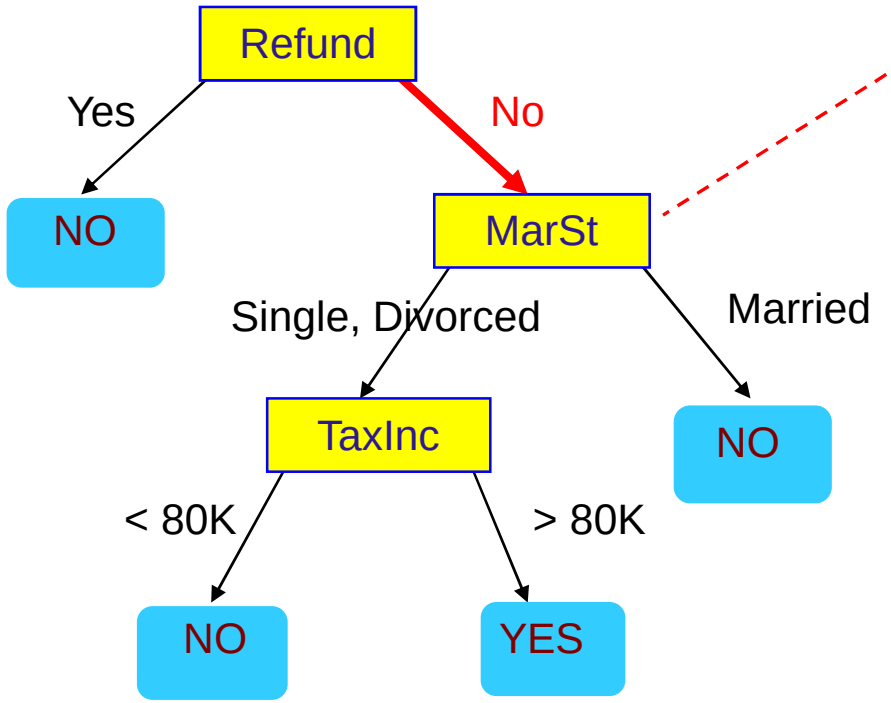


Árvores de Decisão

- **Aplicação da árvore de decisão a um novo exemplo**

Novo exemplo

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

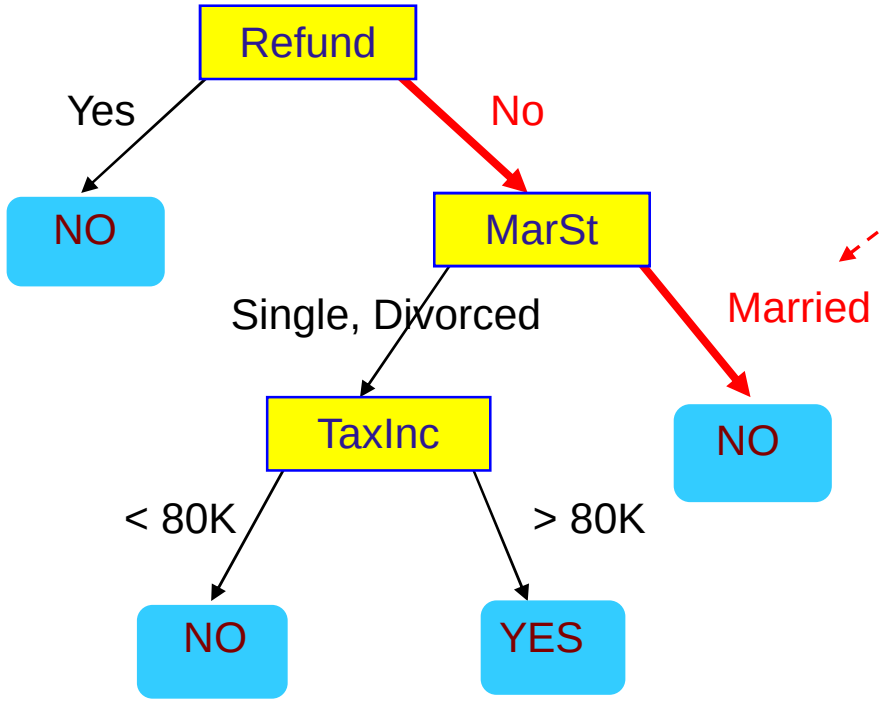


Árvores de Decisão

- **Aplicação da árvore de decisão a um novo exemplo**

Novo exemplo

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

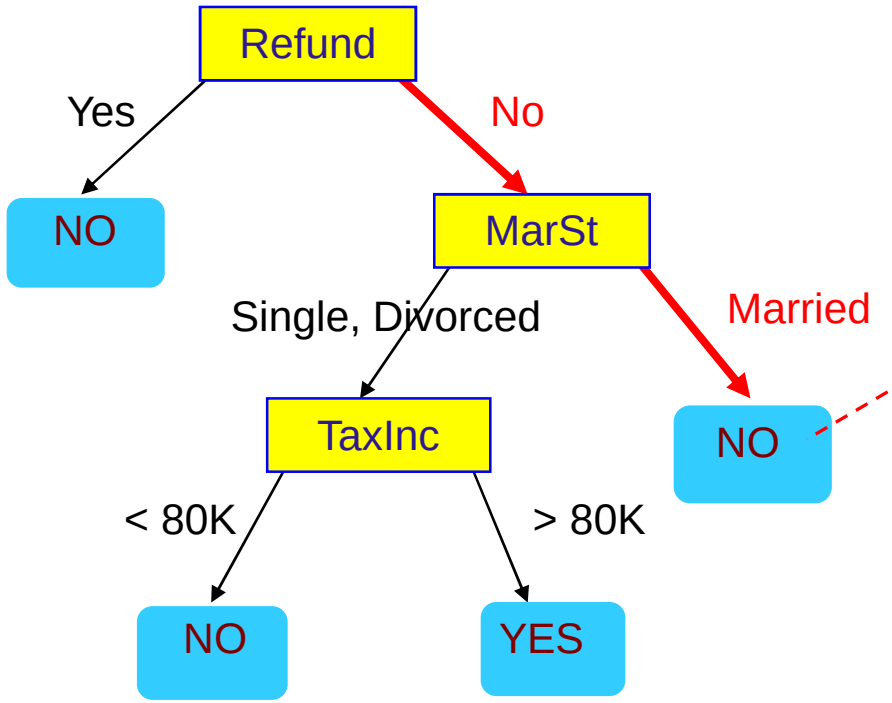


Árvores de Decisão

- **Aplicação da árvore de decisão a um novo exemplo**

Novo exemplo

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Atribuir
Cheat = “No”

Árvores de Decisão

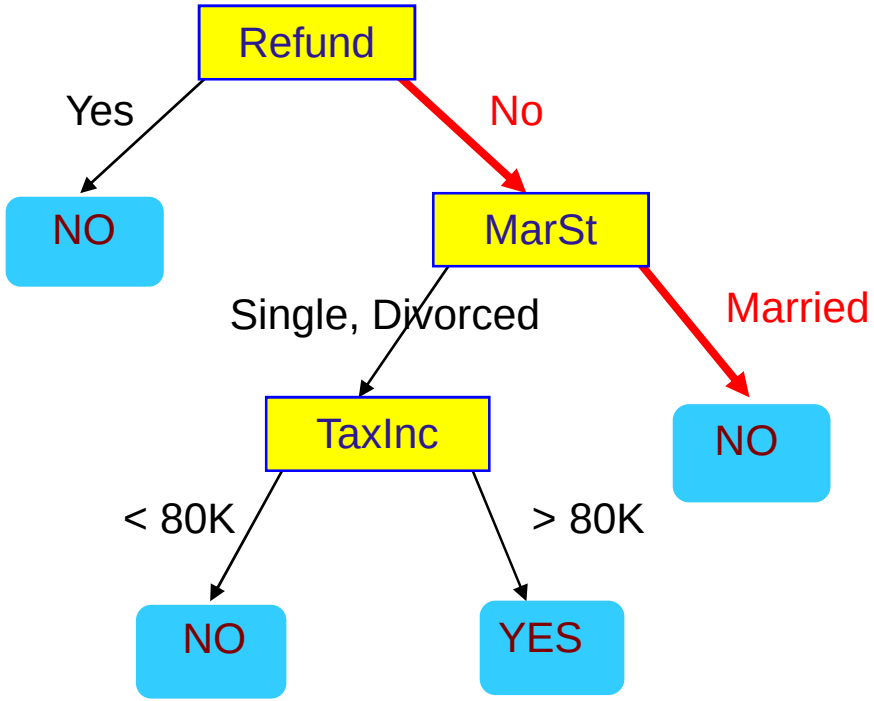
Árvore de decisão – Interpretação

Cada classe – disjunção de conjunções das condições aplicadas aos valores dos diversos atributos

- **Cada ramos da árvore são disjuntos**
- **Cada ramo na árvore = conjunção de condições**

Exemplo: Classe NO

Refund: Yes ∨ (Refund: No ∧ MarSt: Single, Divorced ∧ TaxInc: <80k) ∨ (Refund: No ∧ MarSt: Married)

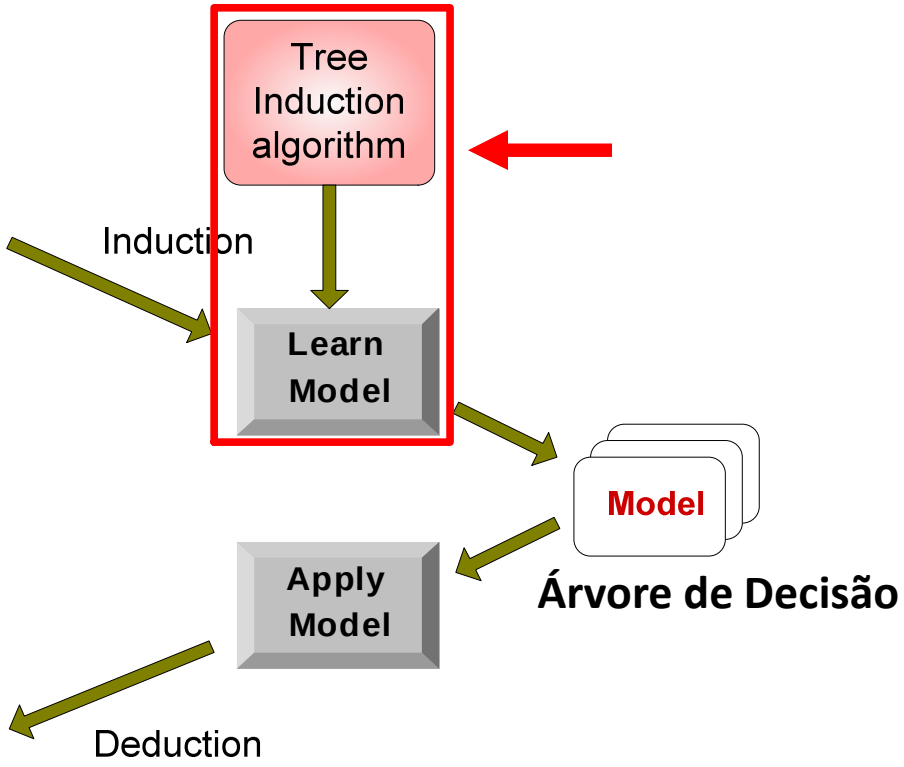


Árvores de Decisão

- **Árvore de decisão – Tarefa de Classificação**

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

Árvores de Decisão

- **Árvore de decisão – Indução**

O problema de construir uma árvore de decisão consistente com um conjunto de exemplos pode ser um problema complexo (NP completo)

  - **Implementar algoritmos heurísticos**

Vários algoritmos (Top Down Induction of Decision Trees)

  - Hunt's Algorithm (um dos primeiros)
  - CART
  - ID3
  - C4.5 (o algoritmo mais popular em ML)

- Pontos a analisar
  - Determinar como dividir os exemplos
    - Como especificar a condição de teste do atributo?
    - Como determinar a melhor partição?
  - Determinar quando parar a divisão

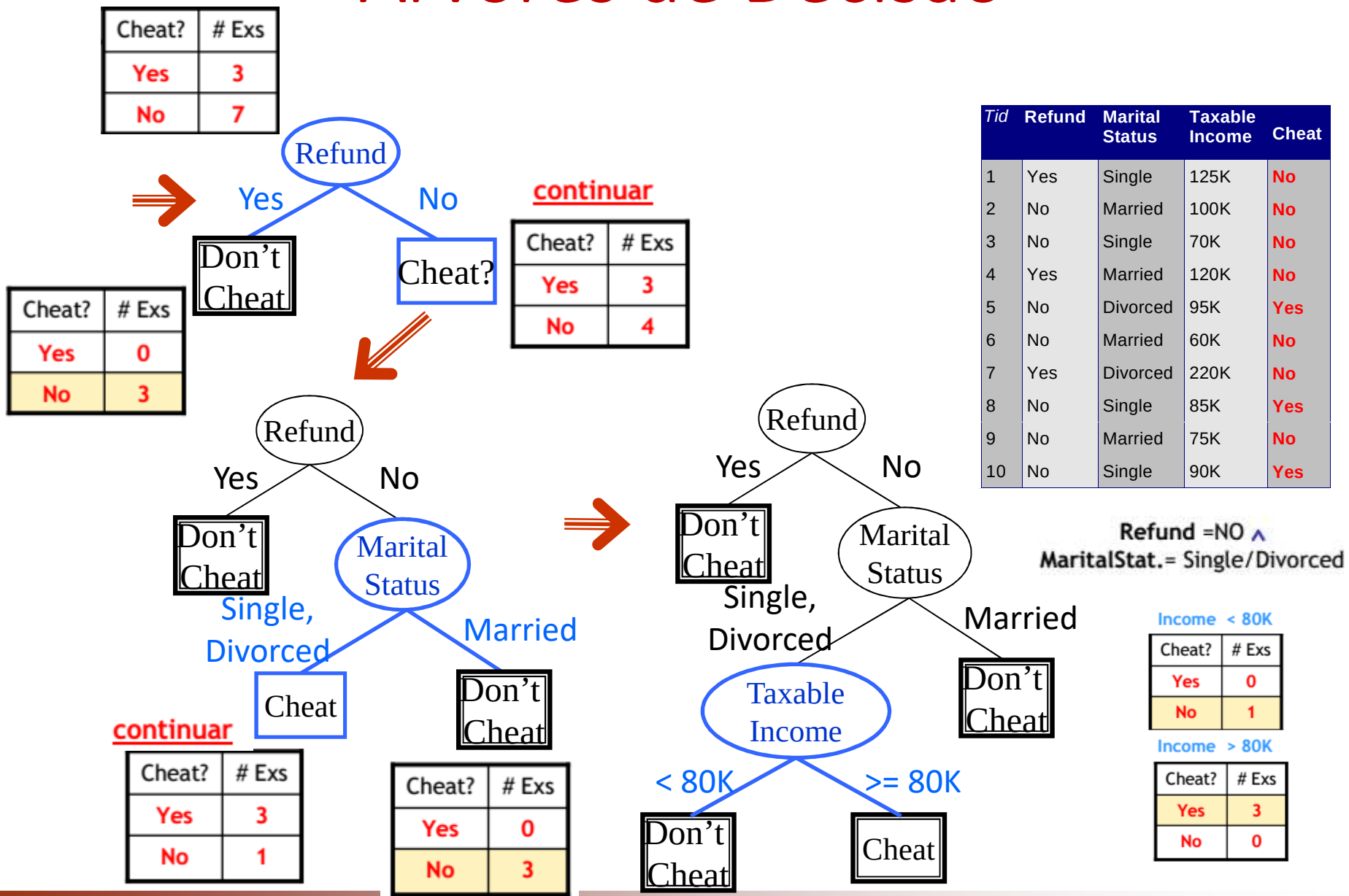
Árvores de Decisão

- Top Down Induction of Decision Trees
- Greedy TDIDT: (estratégia de busca gulosa)
  - guia a pesquisa sobre o espaço de possíveis árvores
  - divide os exemplos segundo um teste do atributo que otimiza um critério de divisão dado
  - constrói a árvore de uma forma recursiva e descendente
  - algoritmo míope: toma decisão olhando para a frente apenas um passo; não reconsidera as opções tomadas

Início: Começar com uma árvore vazia

1. Escolher o melhor atributo A como raiz
   o mais informativo ≡ o que melhor discrimina os exemplos dados
2. Particionar os exemplos de acordo com os valores de A
3. Para cada valor de A:
   estender a árvore adicionando um ramo (subárvore)
   passar os exemplos para as folhas
4. Para cada folha:
   se todos os exemplos são da mesma classe ⇒ associar essa classe à folha
   Senão ⇒ repetir os passos 1 a 4 (procedimento recursivo)

Árvores de Decisão



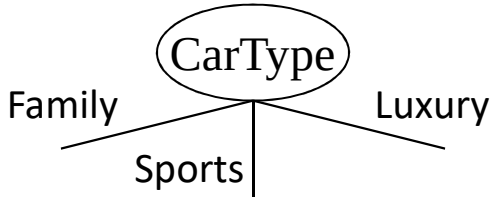
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Árvores de Decisão

- Como especificar a condição de teste do atributo?
  - Depende do tipo de atributo
    - Nominal
    - Ordinal
    - Contínuo
  - Depende de como dividir
    - Em dois (2-way split)
    - Em vários (Multi-way split)

Árvores de Decisão

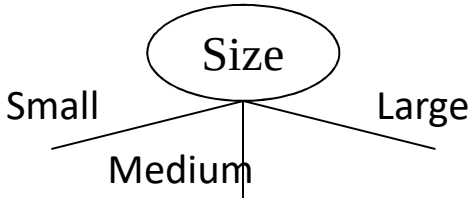
- Partição de Atributos Nominais
- Multi-way split: Usar tantas partições quantos valores



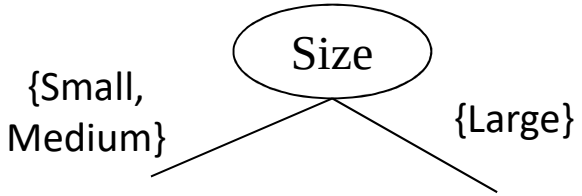
- Binary split: Dividir os valores em 2 subconjuntos.
  Precisa encontrar a partição ótima

Árvores de Decisão

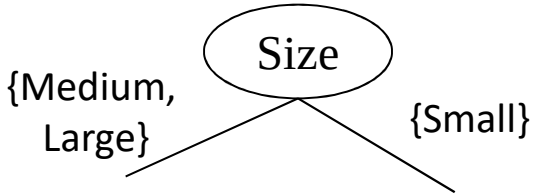
- Partição de Atributos Ordinais
- Multi-way split: Usar tantas partições quantos valores



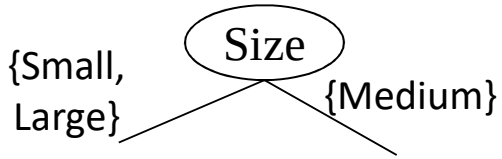
- Binary split: Dividir os valores em 2 subconjuntos. Precisa-se encontrar a partição ótima



Ou

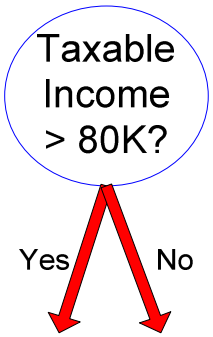


- E esta partição?

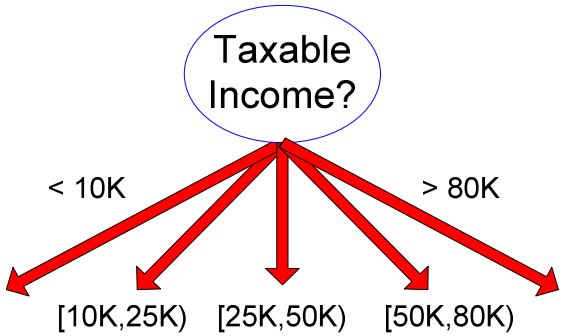


Árvores de Decisão

- Partição de Atributos Contínuos
  - Multi-way split:
    - os intervalos podem ser determinados usando um método de discretização (iguais intervalos, iguais frequências, etc.)
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - considerar todas as possíveis partições e selecionar a melhor
    - pode resultar computacionalmente custoso



(i) Binary split

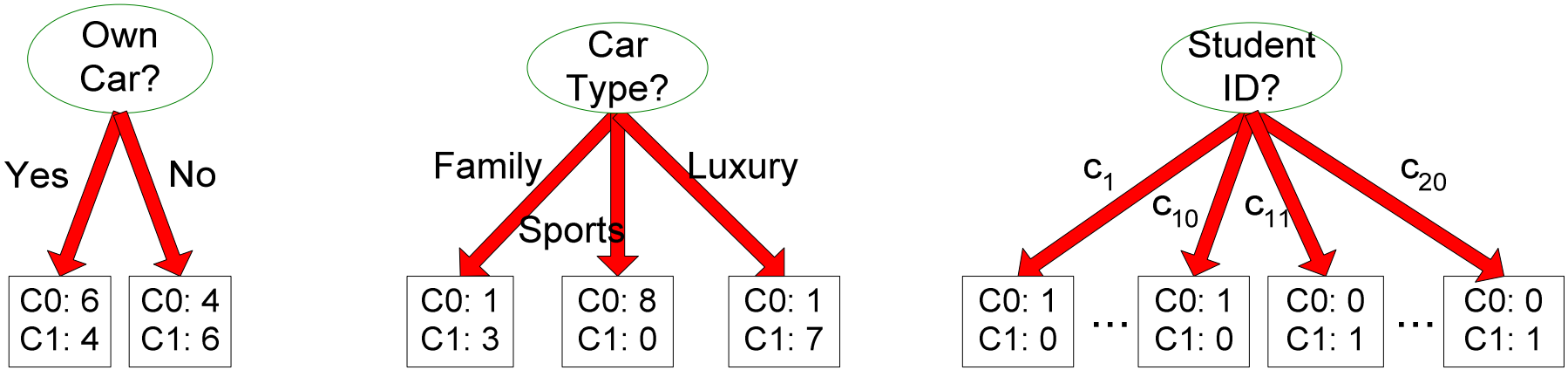


(ii) Multi-way split

Árvores de Decisão

- Qual atributo selecionar?
  - Greedy TDIDT: (estratégia de busca gulosa)
    - guia a procura sobre o espaço de possíveis árvores
    - **particiona os exemplos segundo um teste do atributo que optimiza um critério de divisão dado**
    - constrói a árvore de uma forma recursiva e descendente

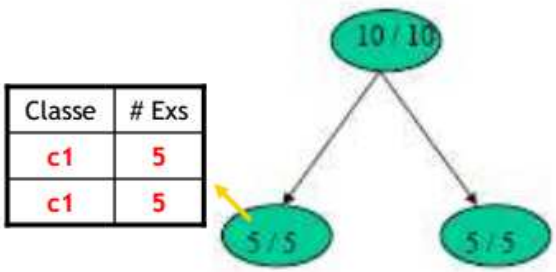
Before Splitting: 10 records of class 0,
10 records of class 1



Qual atributo escolher para testar num dado nó? Qual a melhor condição de teste?

Árvores de Decisão

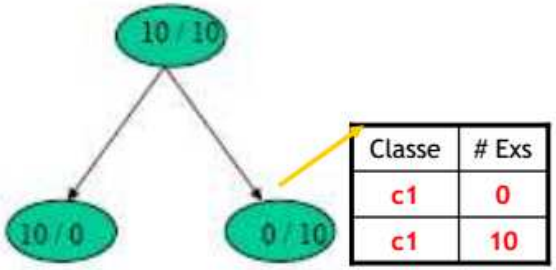
- Qual a melhor divisão?
  - Uma divisão que mantêm as proporções de classes em todas as partições é inútil
  - Uma divisão onde em cada partição todos os exemplos são da mesma classe tem utilidade máxima
  - Nós com distribuição de classe homogénea são preferidos
  - Valorizar a impureza das partições ⇒ medir impureza de um nó



Classe	# Exs
c1	5
c1	5

Nó no-homogéneo

Alto grau de impureza



Classe	# Exs
c1	0
c1	10

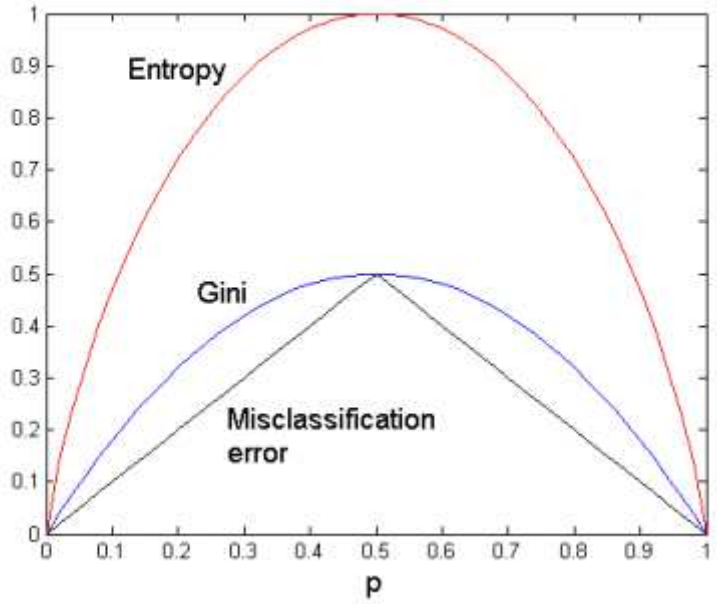
Nó Homogéneo

Sem impureza

Árvores de Decisão

- Medidas de Impureza de um Nó
  - Gini Index
  - **Entropy**
  - Misclassification error

Para um problema com duas classes

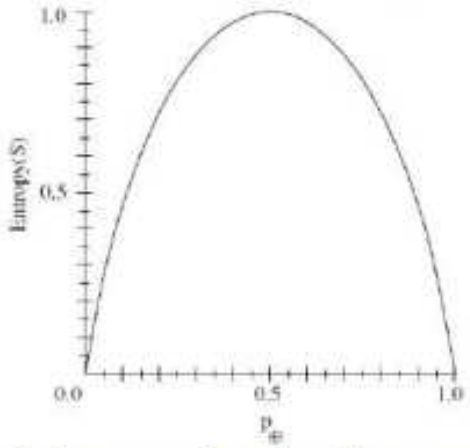


Árvores de Decisão

- **Entropia** - mede o grau de impureza de um conjunto de exemplos
- Dado um conjunto S com exemplos de diversas classes c_1, c_2, …, c_m, a entropia é definida por

$$Entropia(S) = -\sum_{j=1}^{m} p_j \log_2 p_j$$

onde p_j - proporção de exemplos da classe c_j



P⊕ is the proportion of positive examples in S

Para uma variável aleatória discreta X que toma valores $x_1, x_2, \ldots x_n$ com $p_i = P(X=x_i)$

- ✓ A entropia tem máximo ($\log_2 p_i$) se $p_i = p_j$ para qualquer $i <> j$ (distribuição uniforme) ⇒, o max=1 se duas classes
- ✓ $Entropia(X) = 0$ se existe um i tal que $p_i = 1$ (assumindo que $0 * \log_2 0 = 0$)
- ✓ Entropia é uma medida muito usada em Teoria da Informação (TI):
  - ✓ mede a quantidade de informação que existe numa v.a. X
  - ✓ interpretada como o número esperado de bits que são necessários para codificar a v.a. X
  - ⇒ a entropia é definida em bits
- ✓ $-\log_2 p(x)$ representa o número de bits necessário para codificar um símbolo x com probabilidade de ocorrência $p(x)$ segundo a codificação de Shannon

Árvores de Decisão

- Entropia - mede o grau de impureza de um conjunto de exemplos

Distribuição dos exemplos por classes

C1	**0**
C2	**6**

Problema de Classificação Binário

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = - 0 log 0 - 1 log 1 = - 0 - 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = $- (1/6) \log_2 (1/6) - (5/6) \log_2 (1/6) = 0.65$

C1	**2**
C2	**4**

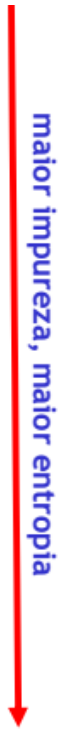
P(C1) = 2/6 P(C2) = 4/6

Entropy = $- (2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$

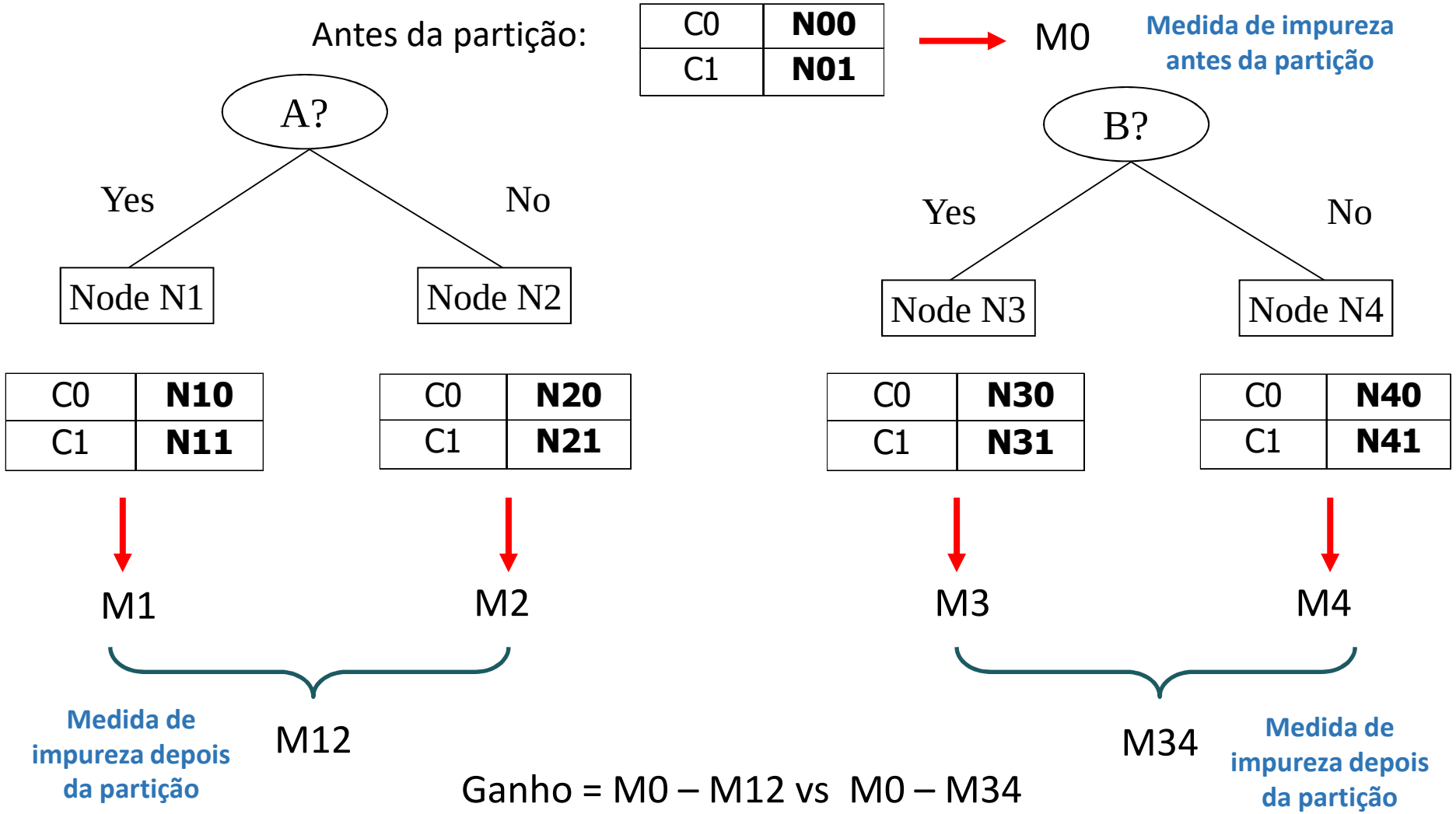
C1	**3**
C2	**3**

P(C1) = 3/6 P(C2) = 3/6

Entropy = $- (1/2) \log_2 (1/2) - (1/2) \log_2 (1/2) = 1$



Árvores de Decisão



Árvores de Decisão

- **Ganho de Informação** - mede o grau de redução da entropia que se obtém pela partição dos exemplos a partir do atributo selecionado
- O ganho de informação obtido por um atributo *A*, num conjunto de exemplos S é definido pela seguinte expressão

$$Ganho(S, A) = Entropia(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropia(S_v)$$

onde S_v – subconjuntos de exemplos de S cujo valor de atributo *A* é igual a *v*

A construção de uma árvore de decisão pode ser **guiada pelo objectivo de diminuir a entropia**
⇔ diminuir o grau de impureza do conjunto S quando os exemplos são particionados pelo teste de um atributo

Qual atributo escolher para nó de decisão, A1 ou A2?

Calcular *Ganho*(S, A1) e *Ganho*(S, A2) e escolher aquele atributo com maior ganho

Árvores de Decisão

- Ganho de Informação
- Quando parar a divisão dos exemplos?
  - Todos os exemplos pertencem à mesma classe
  - Todos os exemplos têm os mesmos valores dos atributos (mas diferentes classes)
  - O número de exemplos é inferior a um certo limite
  - O mérito de todos os possíveis testes de partição dos exemplos é muito baixo

Árvores de Decisão

- Ganho de Informação – Atributos Numéricos
- Um teste num atributo numérico A produz uma partição binária do conjunto de exemplos dada um ponto de partição k, ou seja

$$S = (S \mid A < k) \cup (S \mid A \geq k)$$

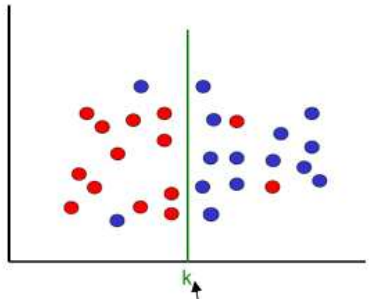
- Como escolher o melhor ponto de partição k ?
- Avaliar o ganho de informação para todos os possíveis "cut points"
  - Ordenar os exemplos por ordem crescente dos valores do atributo numérico e considerar para avaliação o valor médio entre dois valores diferentes e consecutivos

Onde particionar os valores do atributo Temperatura do exemplo Play?

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Y	N	Y	Y	Y	N	N	Y	Y	Y	N	Y	Y	No

suponhamos k=71.5

- ✓ (S|temperature <71.5): yes/4, no/2
- ✓ (S|temperature ≥71.5): yes/5, no/3
- ✓ Info([4,2],[5,3]) = 6/14 info([4,2]) + 8/14 info([5,3]) = 0.939 bits



Escolher o k que maximiza o ganho de informação

Árvores de Decisão

- Ganho de Informação – Atributos com Muitos Valores

$$Ganho(S,A) = Entropia(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropia(S_v)$$

- O ganho de informação para um atributo com muitos valores distintos pode-se tornar muito elevado

Exemplo: suponhamos um atributo toma um valor distinto para cada um dos exemplos existentes

- Como o nó de decisão tem tantos ramos como valores nos atributos
  - cada ramo teria apenas um exemplo
  - entropia nula ⇒ ganho máximo
- Está provado que o algoritmo TDIDT guiado pela medida do ganho de informação pode beneficiar este tipo de atributos
- Solução: penalizá-los durante a seleção usando uma medida que tenha em conta o número e cardinalidade dos subconjuntos criados

  ⇒ usar **Rácio do Ganho** em vez do Ganho

Árvores de Decisão

- Ganho de Informação – Rácio do Ganho
- O Rácio do Ganho é obtido dividindo o valor do ganho de informação pela entropia da partição

$$RacioGanho(S,A) = \frac{Ganho(S,A)}{PartiçãoInfo(S,A)}$$

onde $$PartiçãoInfo = \sum_{v \in Valores(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}$$

Usando esta medida como guia no processo de busca garante-se que

- Sejam penalizadas as partições com elevada entropia (um grande número de pequenas partições)
  - não sejam selecionados atributos com muitos valores distintos
  - não sejam obtidas árvores tão complexas

Árvores de Decisão

ID3 – Algoritmo de Construção de Árvores de Decisão

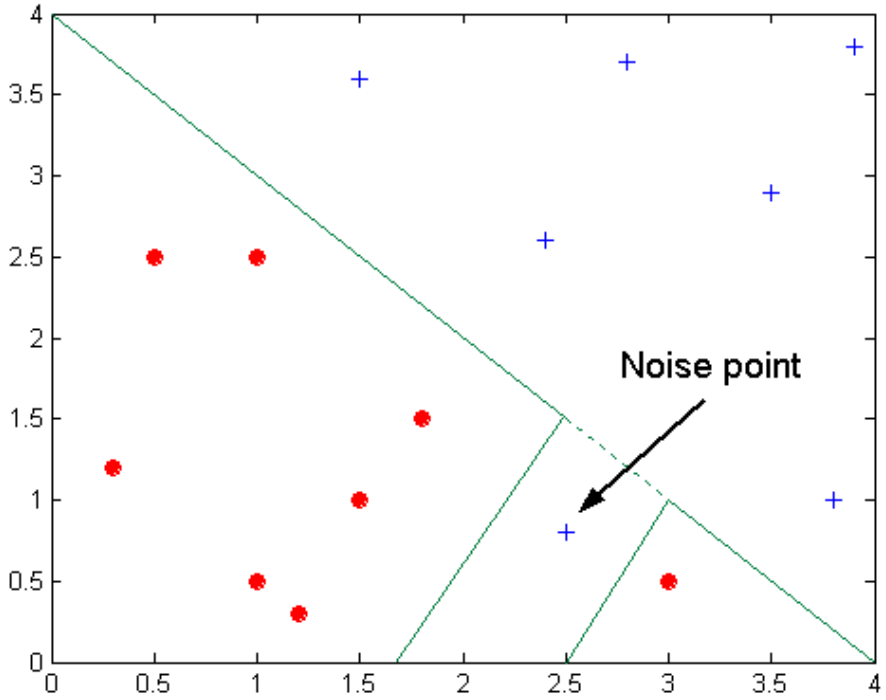
- Algoritmo da família TDIDT: (estratégia de busca greedy)
  - guia a pesquisa sobre o espaço de possíveis árvores
  - divide os exemplos segundo um teste do atributo que otimiza um critério de divisão dado
  - constrói a árvore de uma forma recursiva e descendente (top down)
  - Algoritmo míope: considera apenas próximo passo

Medida para selecionar atributos: ganho de informação
Permite atributos numéricos: não
Tratamento de valores desconhecidos: não

Apesar de quando foi lançado o algoritmo ID3 pareceu eficiente, verificaram-se alguns problemas, como é o caso do sobre-ajustamento da árvore aos dados de treino (Overfitting)

Árvores de Decisão

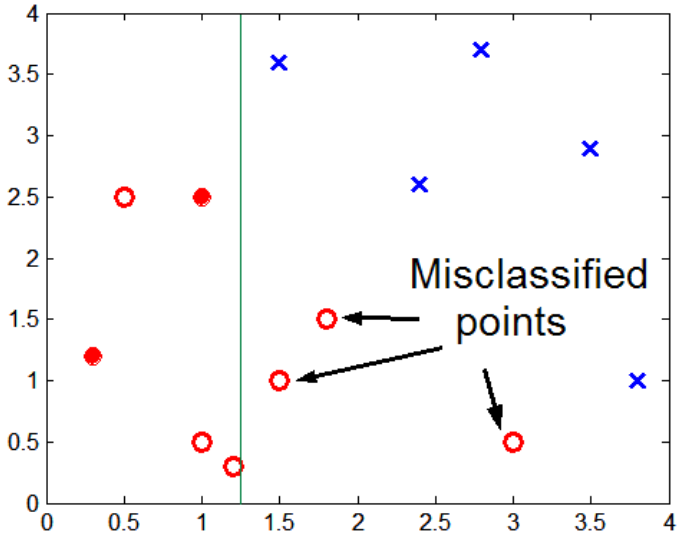
- Overfitting devido ao ruído



A fronteira de decisão é distorsida pelo ponto de ruído

Árvores de Decisão

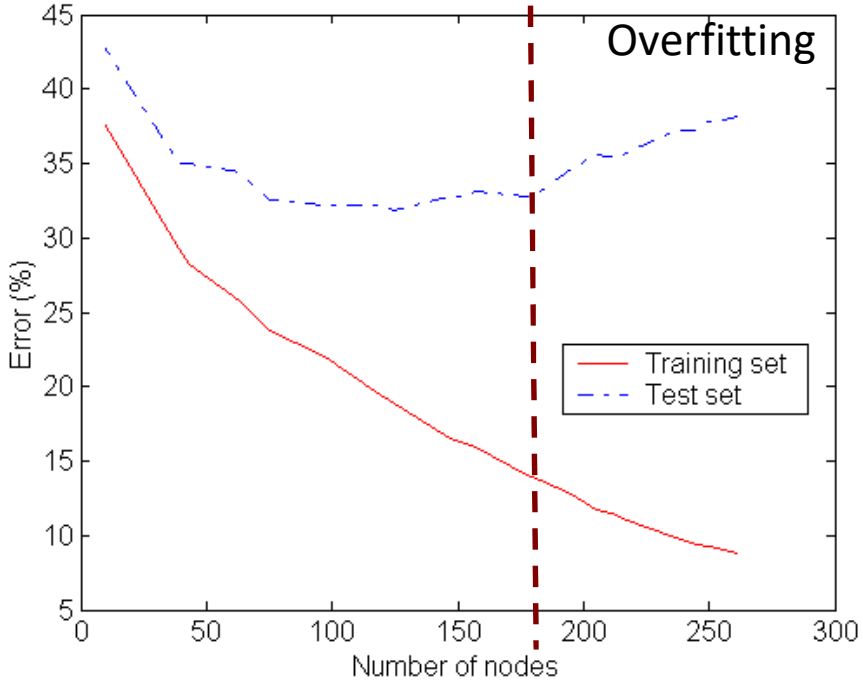
- Overfitting devido a poucos exemplos de treino



- A falta de exemplos na metade inferior do diagrama torna difícil prever corretamente as classes naquela região
- Número insuficiente de exemplos na região faz árvore de decisão prever exemplos de teste usando outros exemplos de treino que são irrelevantes para a tarefa de classificação

Árvores de Decisão

- Overfitting em ID3



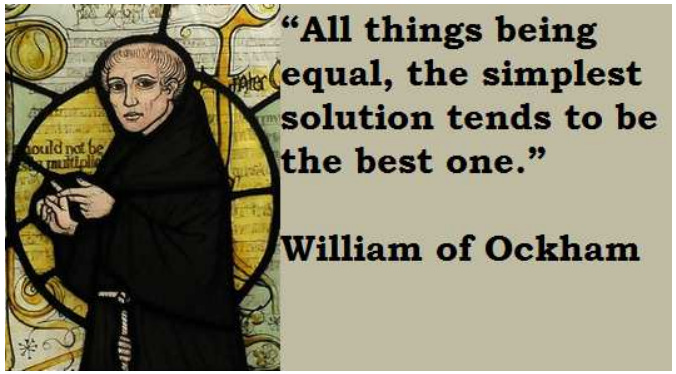
O número de parâmetros de uma árvore de decisão cresce linearmente com o número de exemplos.

Uma árvore de decisão pode obter um ajuste perfeito aos dados de treino.

À medida que a complexidade da árvore aumenta, o modelo vai se ajustar melhor ao conjunto de treino, mas a partir de um certo ponto começa a diminuir a sua capacidade de generalização nos exemplos de teste

Árvores de Decisão

- Navalha de Occam



Dada duas hipóteses que explicam os dados, escolher a mais simples!

- Existem menos hipóteses simples do que complexas
  - Se uma hipótese simples explica os dados é pouco provável que seja uma coincidência
  - Pelo contrário, uma hipótese complexa pode explicar os dados apenas por coincidência
- A complexidade deve ter-se em conta no processo de seleção da melhor hipótese que explica os dados

Árvores de Decisão

- **Como abordar o Overfitting?** – Paragem Antecipada (**Pré-Poda**)

Parar o crescimento da árvore antes que se atinja um ponto onde o erro do conjunto de treino é nulo e a árvore seja demasiada complexa

- Condições de paragem para um nó
  - parar se todas as instâncias pertencem à mesma classe
  - parar se todos os valores dos atributos são iguais
- Condições mais restritas, parar se
  - o número de exemplos é menor que um dado *threshold*
  - a expansão do nó atual não melhora o seu grau de impureza

Árvores de Decisão

- Como abordar o Overfitting? – Paragem Antecipada (**Pós-Poda**)

Crescer a árvore completa e a seguir podá-la, i.e., substituir alguns ramos por folhas

- Ajustar iterativamente os nós da árvore de decisão de maneira ascendente (bottom-up)

para cada nó de decisão:

- Avaliar duas hipóteses: árvore de decisão atual vs. árvore podada
- Se a estimativa do erro de generalização melhora após a poda
  - Substituir pela com a classe maioritária de todos os exemplos associados à sub-árvore podada

Árvores de Decisão

- Estimativas do Erro Verdadeiro – Paragem Antecipada (**Pós-Poda**)

Podar só se a estimativa do erro de generalização (erro verdadeiro) melhora

Como obter estimativas fiáveis do erro verdadeiro durante o processo de poda?

- "reduced-error pruning"
  - estimar o erro num conjunto de teste independente do conjunto de treino que é usado para construir a árvore (holdout validation)

Árvores de Decisão

C4.5 – Algoritmo de Construção de Árvores de Decisão

C4.5 é uma extensão do ID3

- medida para selecionar atributos: rácio do ganho
- Tratamento de valores numéricos
- Tratamento de valores desconhecidos
- Implementa pós-poda para evitar overfitting
- Gera regras de decisão a partir da árvore (C4.5 rules)
- C4.5 é um dos mais populares algoritmos em ML criado por Quinlan

Árvores de Decisão

- Árvores de Decisão - **Vantagens**
- Modelo não-paramétrico
  - Não assume nenhuma distribuição particular para os dados
  - Pode construir modelos para qualquer função desde que o numero de exemplos de treino seja suficiente.
- A estrutura da árvore de decisão é independente da escala das variáveis
  - Transformações monótonas das variáveis (log x, 2*x, ...) não alteram a estrutura
- Elevado grau de interpretação
  - Uma decisão complexa (prever o valor da classe) é decomposto numa sucessão de decisões elementares.
- Os algoritmos de indução são eficientes
- Robusto à presença de outliers e atributos redundantes ou irrelevantes
- Inclui mecanismo de seleção de atributos (embedded feature selection)

Árvores de Decisão

- Árvores de Decisão - **Desvantagens**
- Instabilidade
  - Pequenas perturbações do conjunto de treino podem provocar grandes alterações no modelo aprendido
- Presença de valores desconhecidos
- Fragmentação de conceitos
- Replicação de sub-árvores

Conjunto de Regras

- Classificadores baseados em Regras - Definição
- Classificador = conjunto de regras: if … then …
- Regra: $(Condição) \rightarrow y$
  - Onde
    - Condição = conjunto de atributos
    - *Y* é a classe
    - LHS (left-hand side): antecedente da regra ou condição
    - RHS (right-hand side: consequente da regra
- Exemplos

$$(\text{Blood Type=Warm}) \wedge (\text{Lay Eggs=Yes}) \rightarrow \text{Birds}$$
$$(\text{Taxable Income} < 50\text{K}) \wedge (\text{Refund=Yes}) \rightarrow \text{Evade=No}$$

Conjunto de Regras

- Classificadores baseados em Regras - Exemplo

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds
R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes
R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals
R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles
R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Conjunto de Regras

- Classificadores baseados em Regras - Exemplo

Uma regra ***r*** cobre um exemplo (instância) **x**
se os seus atributos satisfazem a condição da regra

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds
R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes
R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals
R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles
R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

A regra **R1** cobre a **hawk** => **Bird**
A regra **R3** cobre the **grizzly bear** => **Mammal**

Conjunto de Regras

- Classificadores baseados em Regras – Características
- Regras mutuamente exclusivas
  - Um classificador contém regras mutuamente exclusivas se as regras são independentes entre si
  - Cada exemplo é coberto no máximo por uma regra
- Regras exaustivas
  - Classificador tem cobertura exaustiva se ele leva em conta toda possível combinação dos valores dos atributos
  - Cada registo é coberto por pelo menos uma regra

Conjunto de Regras

- Classificadores baseados em Regras – Classificação

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

lemur acciona a regra R3, por isso é classificado como **mamífero**

turtle acciona as regras R4 and R5 ⇒ conflito!

se lista de regras ordenada por prioridade (decision list),
então atribui-se a classe da regra com mais prioridade ⇒ **reptil**

dogfish shark não acciona nenhuma das regras

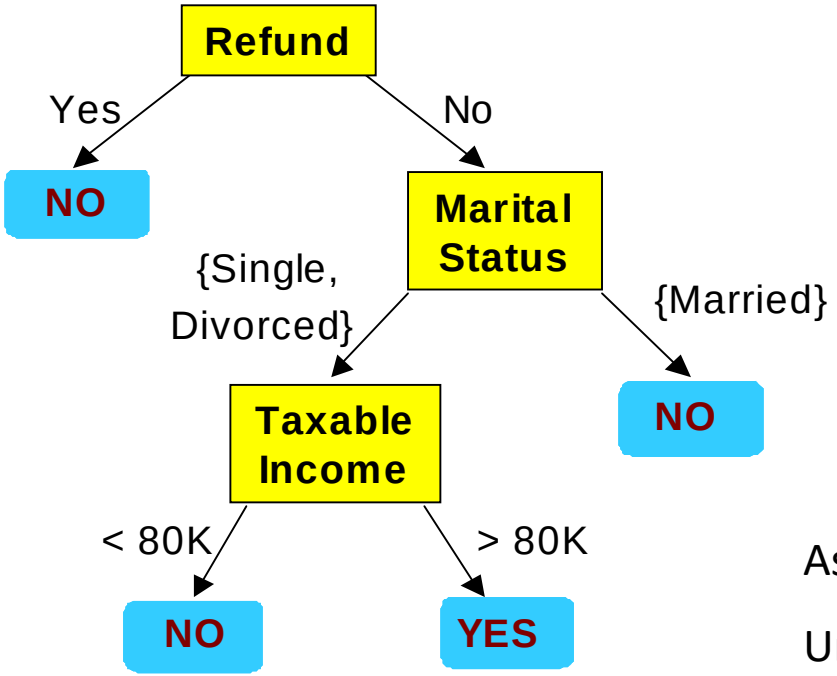
⇒ o classificador não tem um coverage exaustivo
(cada exemplo deve ser coberto pelo menos por uma regra)

Conjunto de Regras

- Classificadores baseados em Regras – Aprendizagem
- Métodos Diretos
  - Extraem as regras diretamente dos dados
  - Ex: RIPPER, CN2, Holte's 1R
- Métodos Indiretos
  - Extraem as regras de outros modelos de classificação
  - Ex: árvores de decisão, redes neuronais, …
  - mais popular: C4.5rules

Conjunto de Regras

- Classificadores baseados em Regras – Árvore de Decisão



Classification Rules

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

As regras são mutuamente exclusivas e exaustivas

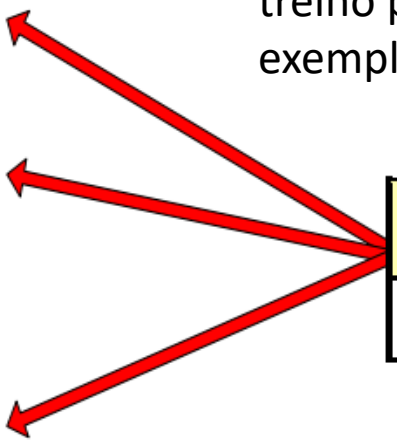
Um conjunto de regras contem tanta informação quanto a árvore

Classificadores baseados em Instâncias

- Classificadores baseados em Instâncias (Lazy classifier)

Conjunto de Treino

Atr1	.....	AtrN	Class
			A
			B
			B
			C
			A
			C
			B



Novo exemplo

Atr1	.....	AtrN

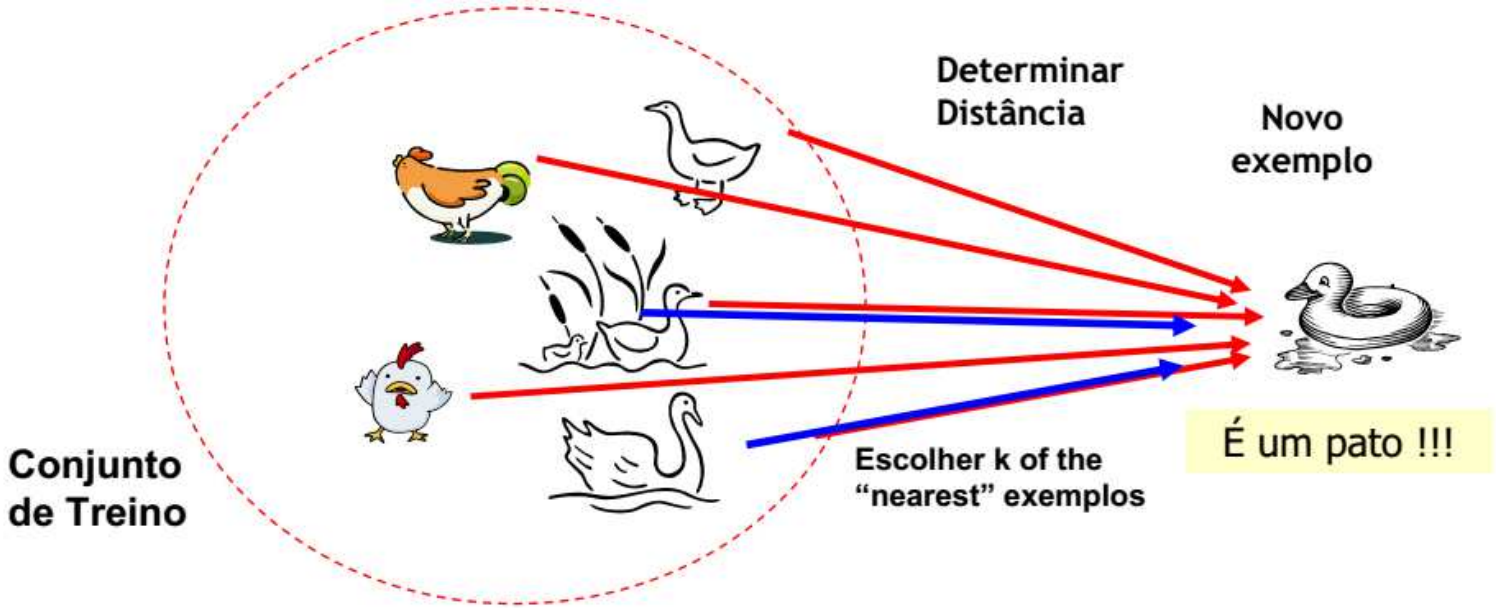
- Não constrói um modelo dos dados, por isso é chamado preguiçoso
- Usa apenas o conjunto de treino para classificar os novos exemplos

Classificadores baseados em Instâncias

- Classificadores baseados em Instâncias - Exemplos
- Rote-learner
  - Memoriza todos os exemplos do conjunto de treino e realiza a classificação de um novo exemplo somente se os atributos deste correspondem exatamente a um dos exemplos do conjunto de treino
- k-vizinhos mais próximos (k-nearest neighbor)
  - Usa os k pontos “mais próximos” (vizinhos mais próximos) para realizar a classificação de um novo exemplo

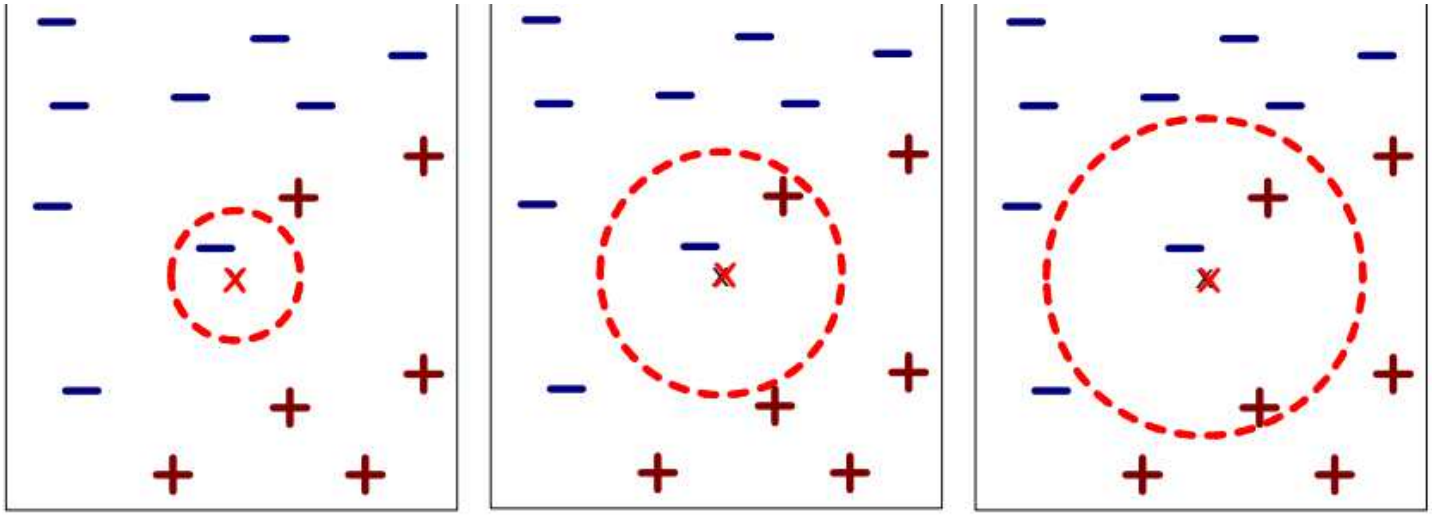
Classificadores baseados em Instâncias

- k-vizinhos mais próximos (k-nearest neighbor)
  - Ideia básica: Se ele anda como um pato, grasna como um pato então é provavelmente um pato



Classificadores baseados em Instâncias

- **k-vizinhos mais próximos (k-nearest neighbor)**

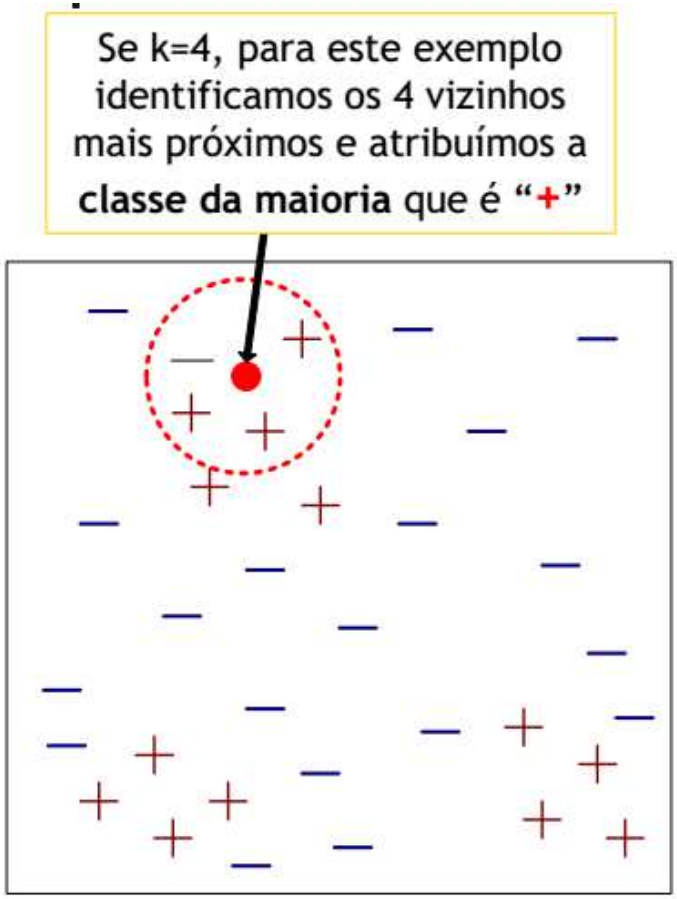


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Os k-vizinhos mais próximos são os k pontos do conjunto de treino que estão a menor distância do novo exemplo x

Classificadores baseados em Instâncias

- **k-vizinhos mais próximos (k-nearest neighbor)**



Requer:

1. Conjunto de treino
2. Métrica de distância: usualmente distância Euclidiana
3. Valor de k: o número de vizinhos mais próximos

Para classificar um novo exemplo:

1. Calcular a distância entre o novo exemplo e cada um dos exemplos do conjunto de treino
2. Identificar os k vizinhos mais próximos
3. A classe do novo exemplo pode ser atribuída pelo "voto da maioria"

Demo: http://vision.stanford.edu/teaching/cs231n-demos/knn/

Classificadores Bayesianos

- Os resultados da classificação são expressos através de uma distribuição de probabilidades
  - em vez de usar f: X → C usamos f : X → P(C| X)
  - se input: exemplo x=(x_1,x_2, …x_n) ∈X ⇒ output: P(c_j| x) para cada classe ou seja a probabilidade que o exemplo x pertença a cada uma das classes

A classe c^* atribuída a um exemplo **x** é aquela que maximiza **P(c_j| x)** :

$$c^* = h_C(\mathbf{x}) = \arg \max_{j=1...m} P(c_j \mid \mathbf{x})$$

Como determinar P(c_j | x) ? Usando o Teorema de Bayes

posterior α prior x likelihood

$$P(c_j \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid c_j)P(c_j)}{P(\mathbf{x})}$$

P(x) pode ser ignorado porque é o mesmo para todas as classes

Maximum a posteriori classification

$$h_{MAP}(\mathbf{x}) = \arg \max_{j=1...m} P(\mathbf{x} \mid c_j)P(c_j)$$

Classificadores Bayesianos

- Classificador Naive Bayes

Bayesian Classification

$$h_{MAP}(\mathbf{x}) = \arg \max_{j=1...m} P(\mathbf{x} \mid c_j) P(c_j)$$

Como determinar P(x| c_j) ?

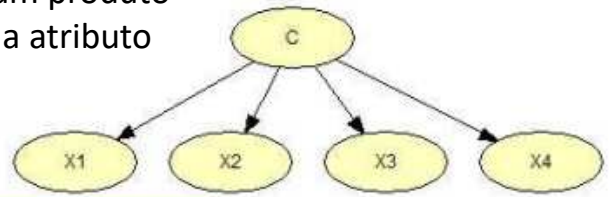
Se o espaço de atributos é de dimensão elevada (muitos atributos, muitos valores para cada atributo) é **computacionalmente muito difícil** (muitas combinações a ter em conta)

Solução: Naïve Bayes - assume uma **"very naïve"** suposição:

Todos os atributos são condicionalmente independentes dado a classe

Podemos decompô-lo num produto de n termos, um por cada atributo

$$P(\mathbf{x} \mid c_j) = \prod_{i=1}^{n} P(X_i = x_i \mid c_j)$$



NB classification

$$c^* = h_{NB}(\mathbf{x}) = \arg \max_{j=1...m} \prod_{i=1}^{n} P(X_i = x_i \mid c_j) P(c_j)$$

Classificadores Bayesianos

- Classificador Naive Bayes – classificar um exemplo

x =(age<=30 , income =medium, student=yes,credit_rating=fair)

Conjunto de Treino

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

1. Estimar as probabilidade apriori $P(c_j)$ de cada classe
   P(buys_computer = "yes") = 9/14 = 0.643
   P(buys_computer = "no") = 5/14= 0.357

2. Estimar as condicionais $P(x_i|c_j)$ para cada atributo
   P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
   P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
   P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
   P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
   P(student = "yes" | buys_computer = "yes") = 6/9 = 0.667
   P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
   P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
   P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

3. Calcular $P(x|c_j)$ para cada classe
   P(x|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
   P(x|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

x pertence à classe "buys_PC=yes"

máximo

4. Usar Teorema de Bayes para calcular $P(c_j|x)$ para cada classe
   P(buys_computer = "yes" |x) = P(x|buys_computer = "yes") * P(buys_computer = "yes")= **0.028**
   P(buys_computer= "no" |x) = P(x|buys_computer = "no") * P(buys_computer = "no") = 0.007

Exemplo do livro Data Mining: Concepts and Techniques, Han & Kamber

Classificadores Bayesianos

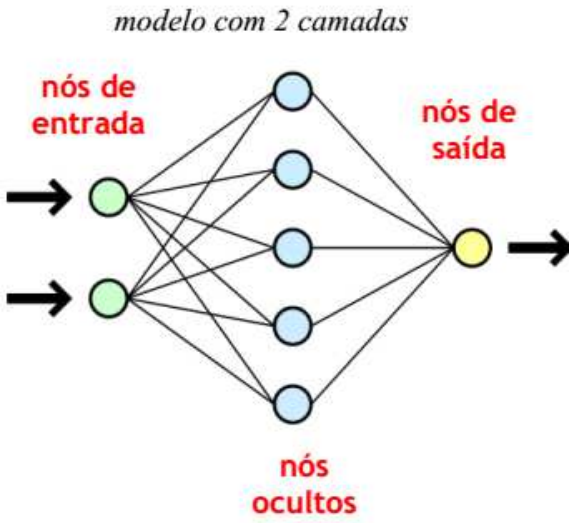
- Classificador Naive Bayes – Conclusões
  - Robusto a isolar pontos de ruído
  - Lida com valores omissos ignorando a instância durante a estimação do cálculo das probabilidades
  - Robusto para atributos irrelevantes
  - Pressuposto da independência poderá não ser suportada por alguns atributos
    - Utilizar outras técnicas como Bayesian Belief Networks (BBN)

Redes Neuronais Artificiais (RNAs)

- As RNAs estão inspiradas no funcionamento do cérebro humano
- Como o cérebro, as RNAs estão compostas dum elevado número de neurónios altamente inter-ligados trabalhando em paralelo na resolução de problemas
- O conhecimento é adquirido através da aprendizagem e armazenado nas ligações

Redes Neuronais Artificiais (RNAs)

- As RNAs contêm tem 3 tipos de camadas (layers)
  - input layer: informação que entra no neurónio
  - hidden layer: camadas intermédias, podem ser várias
  - output layer: informação que sai



Redes Neuronais Artificiais (RNAs)

- Modelos mais populares
  - Perceptron
    - modelo mais simples, não usa hidden layers
  - Multi-layer Feed-Forward
    - os nós de uma camada estão ligados apenas aos nós da seguinte camada

Redes Neuronais Artificiais (RNAs)

• Modelo de um Neurónio Artificial

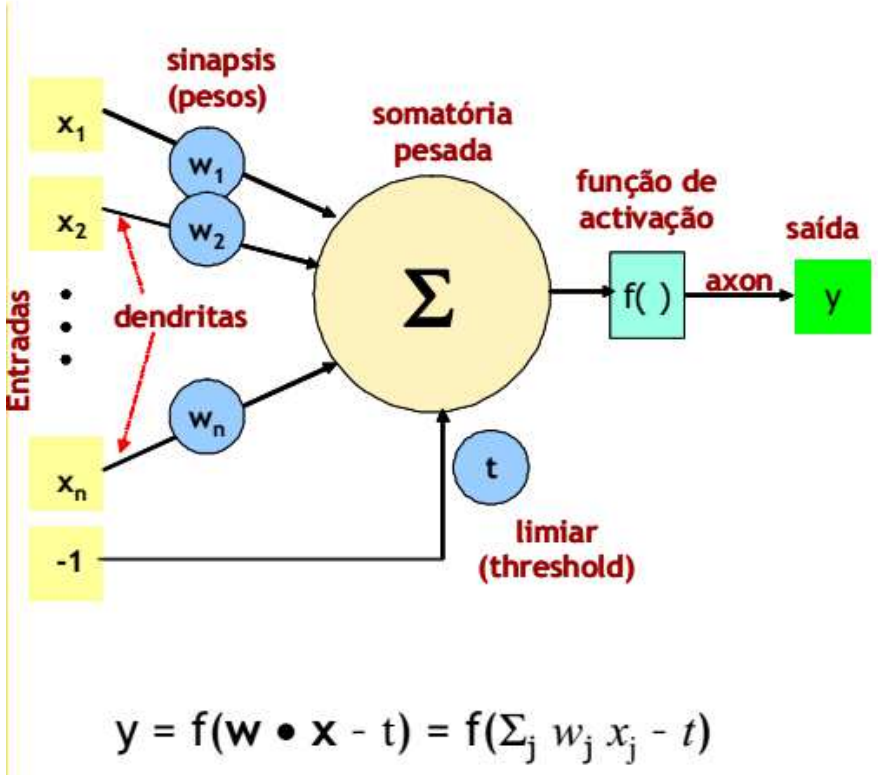
Neurónio = processador de informação
recebe e envia impulso a milhares de neurónios

- dendritos: canais de entrada
- Corpo celular: órgão de cômputo (soma ponderada, função não linear)
- axónio: canal de saída distribuição a outros neurónios

- Potencial pos-sináptico
  $h(\mathbf{w}, \mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = \sum_j w_j x_j$

- O fator bias t é usualmente subtraído ao potencial pos-sináptico

- função de ativação para obter y
  $y = f(\mathbf{w}.\mathbf{x} - t)$



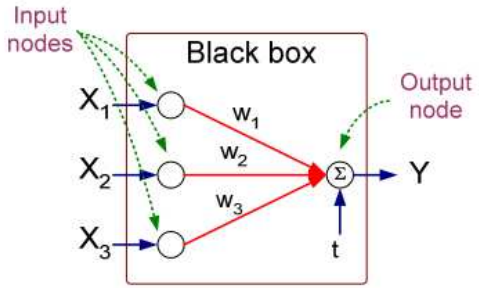
$y = f(\mathbf{w} \bullet \mathbf{x} - t) = f(\Sigma_j w_j x_j - t)$

Redes Neuronais Artificiais (RNAs)

- Perceptron
  - Modelo mais simples das RNAs concebido em 1957 no Cornell Aeronautical Laboratory por Frank Rosenblatt

Um perceptron é um classificador linear binário, ou seja, uma função f: X → Y que mapea um vector binário x a um valor binário y

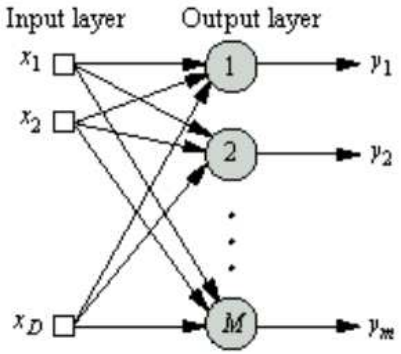
Modelo simples de um neurónio artificial



função de activação: a função sinal

$$y = sign(\sum_i w_i x_i - t) = \begin{cases} 1 & \text{if } \mathbf{w}.\mathbf{x} - t > 0 \\ -1 & \text{otherwise} \end{cases}$$

Modelo geral com vários outputs onde cada output é modelado com o modelo do neurónio simples)

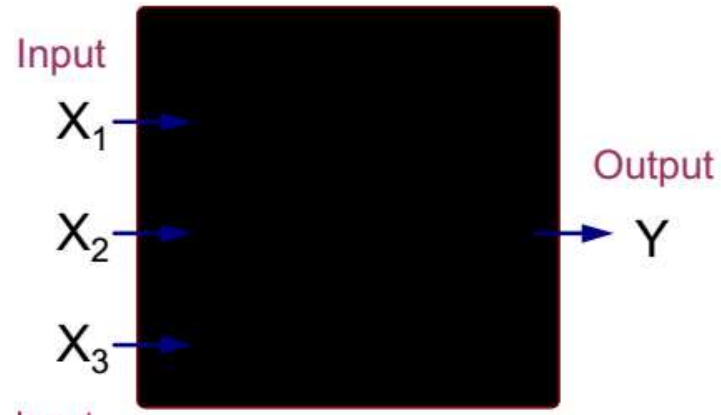


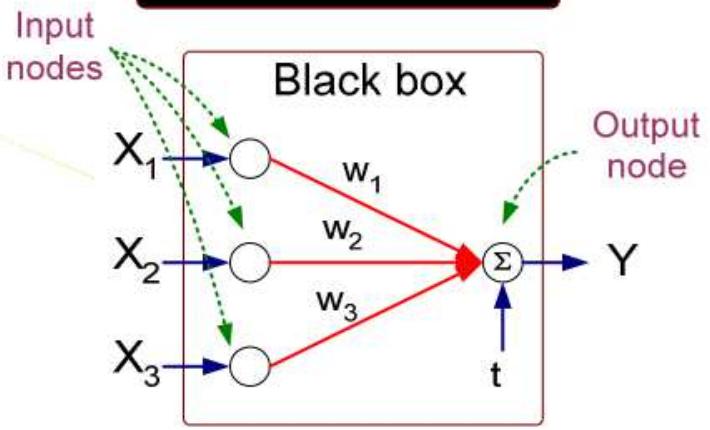
Redes Neuronais Artificiais (RNAs)

- RNAs – Modelo Caixa Preta (Black-Box)

Conjunto de Treino

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





Aprender um Perceptron dos dados significa aprender os pesos que melhor ajustem o modelo oculto nos dados

Redes Neuronais Artificiais (RNAs)

- RNAs – Modelação de uma Função Booleana

Função alvo: (target function)

o valor de y é true se pelo menos 2 inputs são true

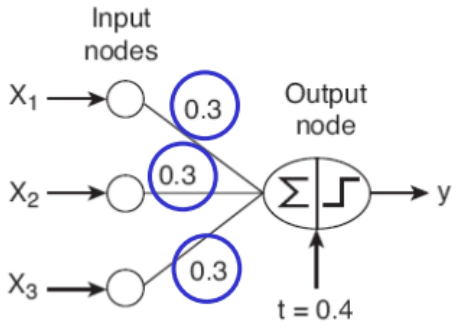
$$y = f(\mathbf{x}) = \begin{cases} 1 & \text{se } (x_1 \wedge x_2 = 1) \vee (x_1 \wedge x_3 = 1) \vee (x_3 \wedge x_1 = 1) \\ -1 & \text{otherwise} \end{cases}$$

Modelação com Perceptron

$$\hat{y} = sign\,(0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4)$$

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

(a) Data set.



(b) Perceptron.

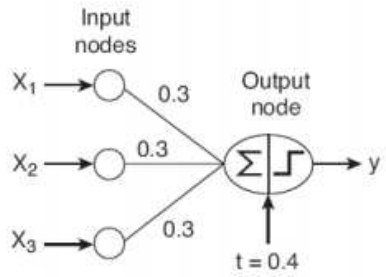
Modelar uma função booleana utilizando um Perceptron

Redes Neuronais Artificiais (RNAs)

- Perceptron – Fronteira de Decisão Linear

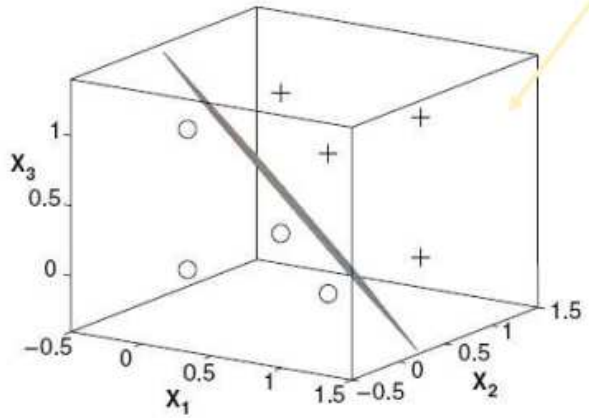
X_1	X_2	X_3	y
1	0	0	−1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	−1
0	1	0	−1
0	1	1	1
0	0	0	−1

(a) Data set.



(b) Perceptron.

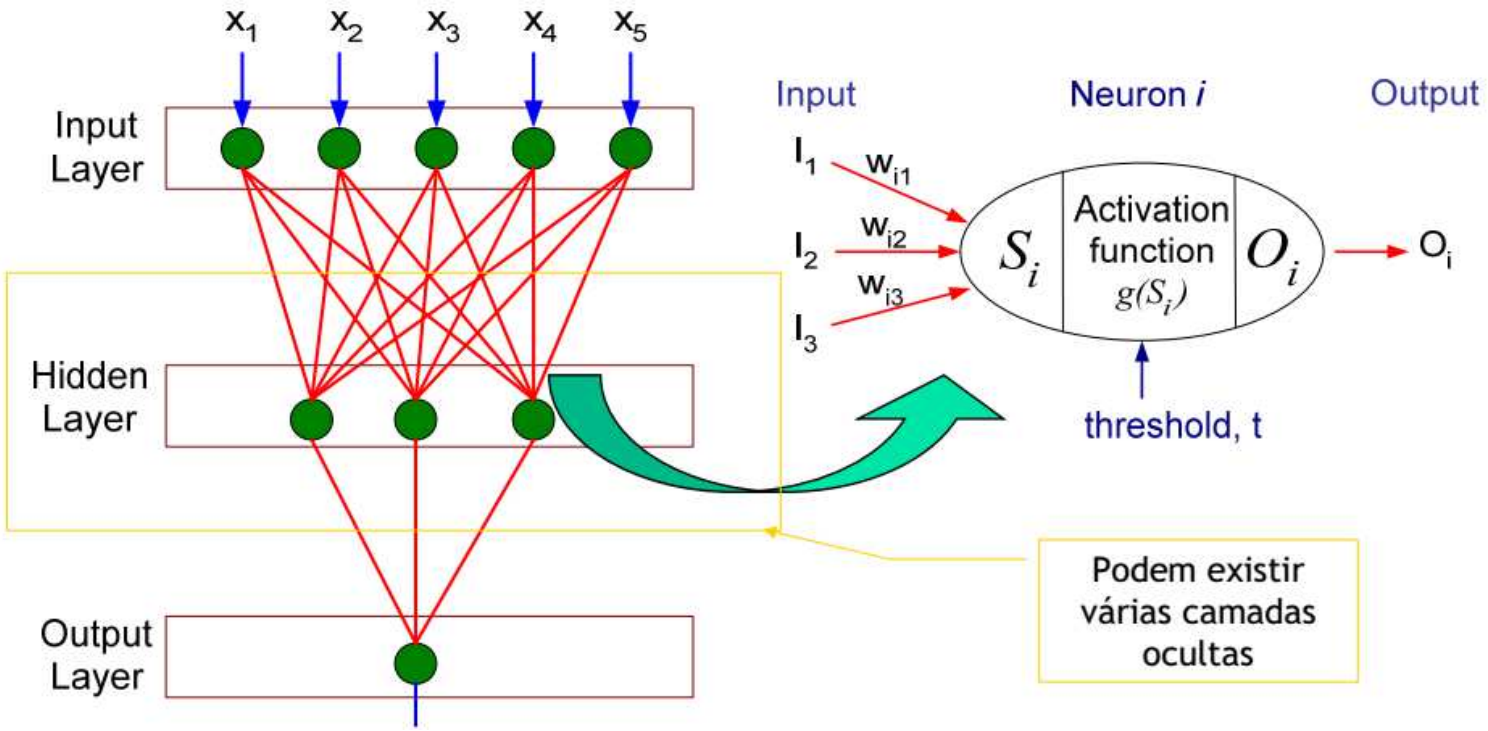
O perceptron é usado para conjunto de treinos **linearmente separáveis**



Fronteira de Decisão do Perceptron para os dados

Redes Neuronais Artificiais (RNAs)

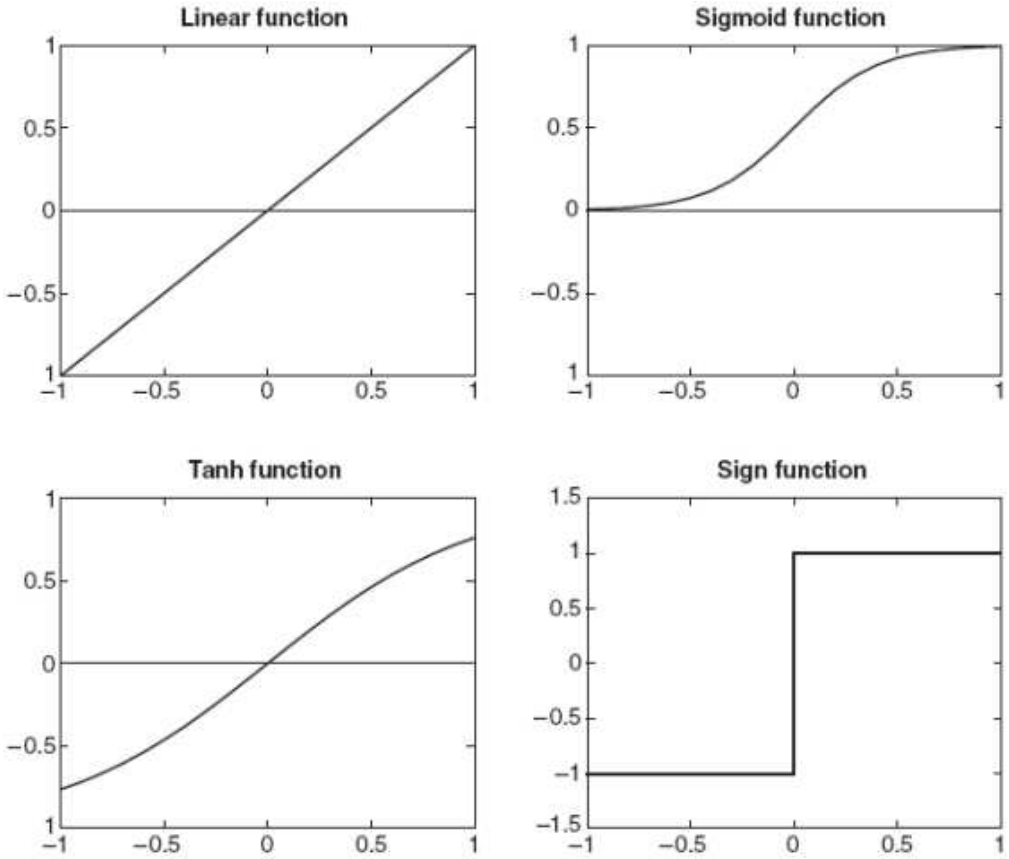
- RNAs multi-camadas Feed-Forward



Aprender RNAs significa aprender os pesos dos dados.
O algoritmo mais popular: back-propagation

Redes Neuronais Artificiais (RNAs)

- RNAs multi-camadas Feed-Forward



Tipos de função de ativação em RNAs

Redes Neuronais Artificiais (RNAs)

- RNAs – Problema de Classificação XOR

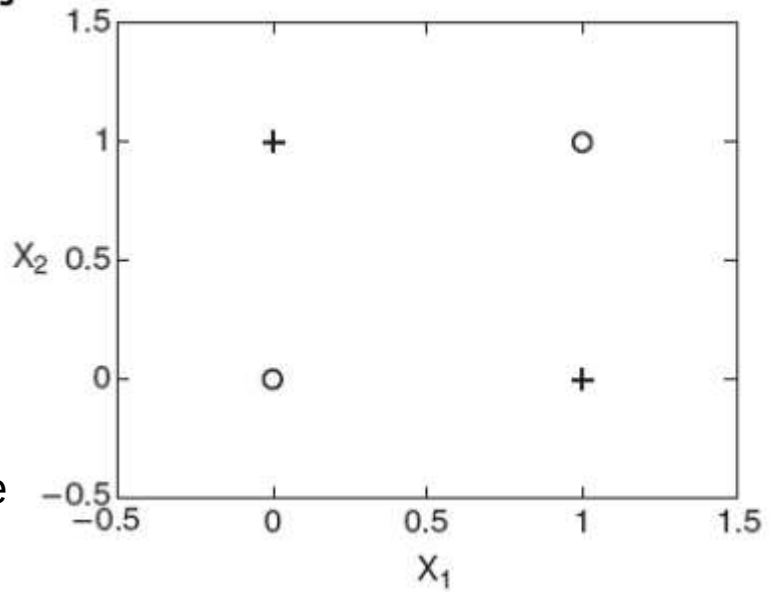
Não existe uma fronteira linear que possa separar estas duas classes
(como o Perceptron pode construir apenas um hiperplano, não podemos usa-lo para modelar este problema)

OU exclusivo

XOR(x1, X2) = true se e só se um dos inputs é true

(note que aqui true = 1, false = -1)

X_1	X_2	y
0	0	−1
1	0	1
0	1	1
1	1	−1

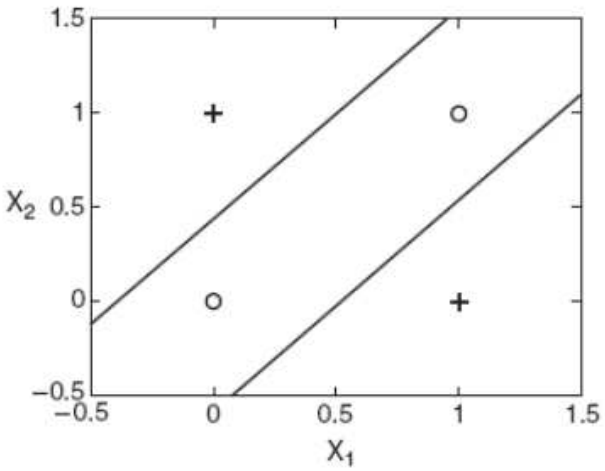


Nenhum hiperplano linear consegue separar as duas classes

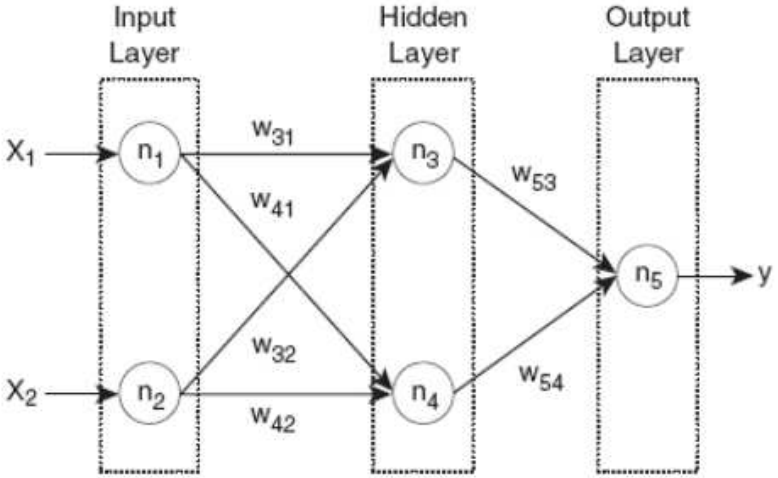
Redes Neuronais Artificiais (RNAs)

- RNAs – Problema de Classificação XOR (Duas Camadas)

  Os exemplos podem ser classificados usando 2 hiperplanos que dividem o espaço dos atributos (input space) em suas respetivas classes



(a) Decision boundary.



(b) Neural network topology.

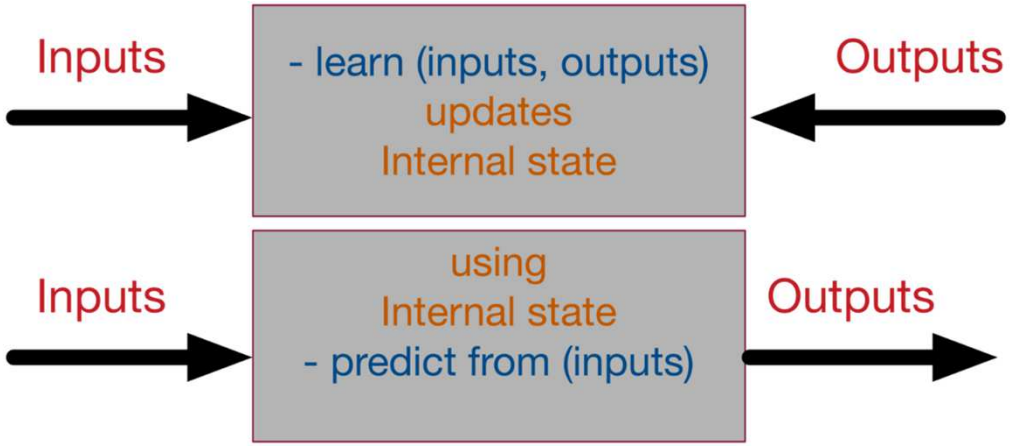
2 nós de entrada, 2 ocultos e um de saída

Feed-Forward Neural Network - Duas camadas

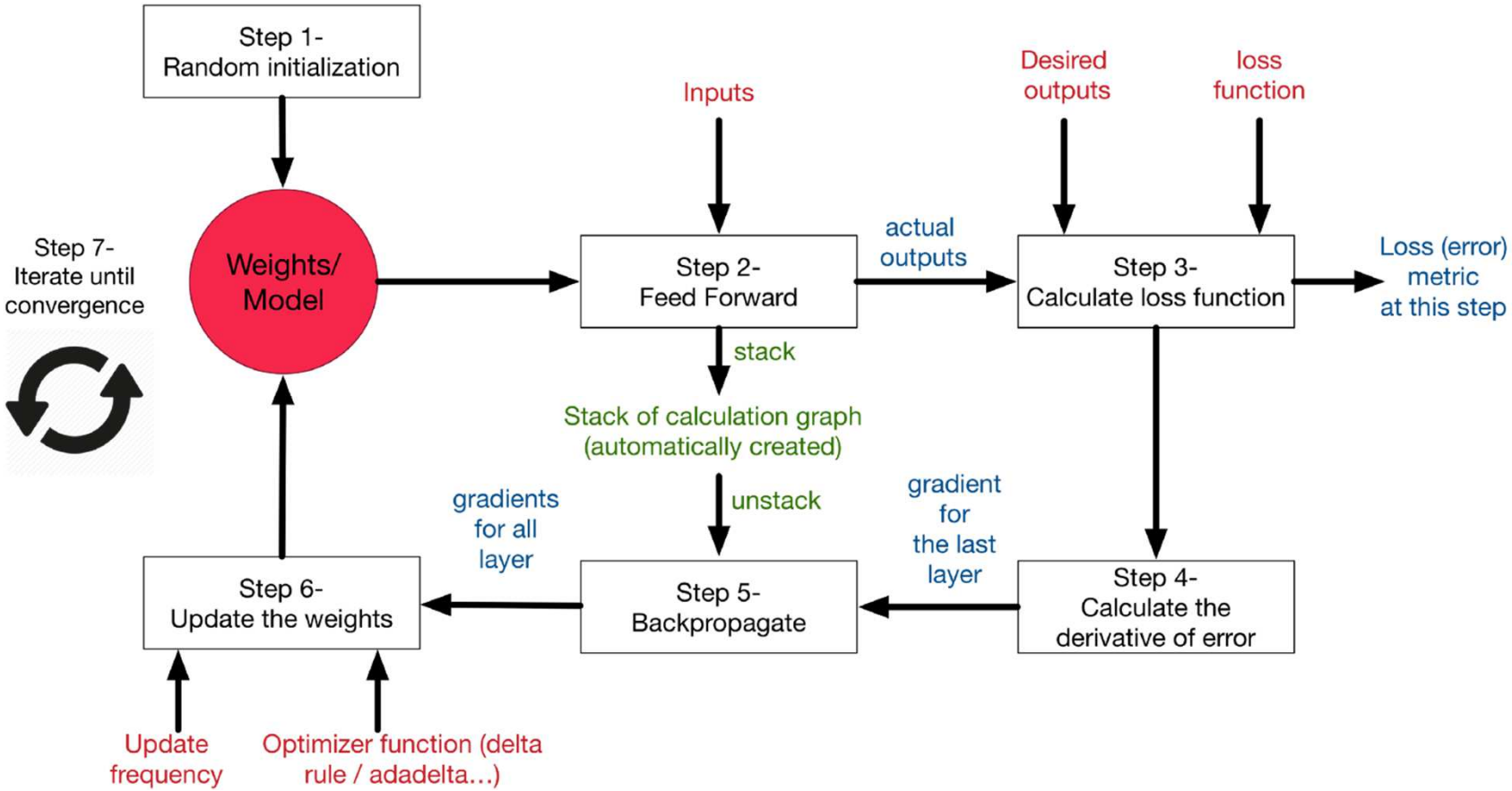
Redes Neuronais Artificiais (RNAs)

Treino com BackPropagation – Retro Propagação

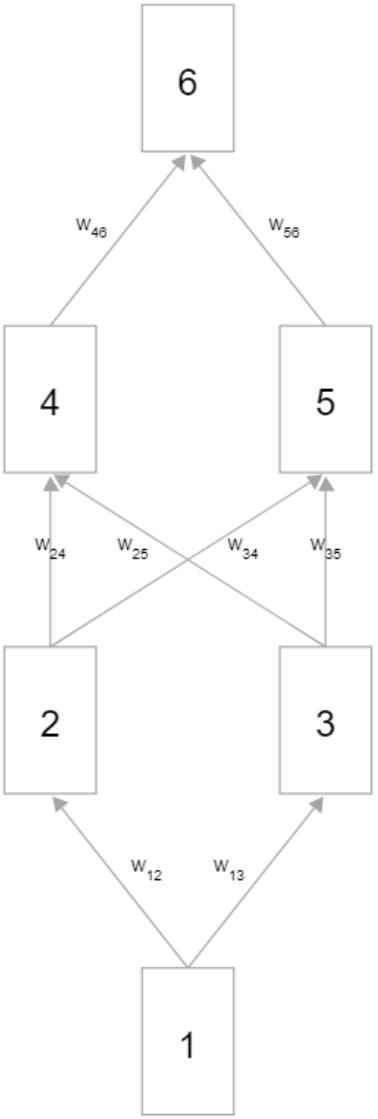
A motivação para a retro propagação é treinar uma rede neural de múltiplas camadas de modo que ela possa aprender as representações internas (pesos das ligações) apropriadas para permitir que a rede mapeie as suas entradas para as saídas



Treino da Rede: Back Propagation



Back Propagation



Simple neural network

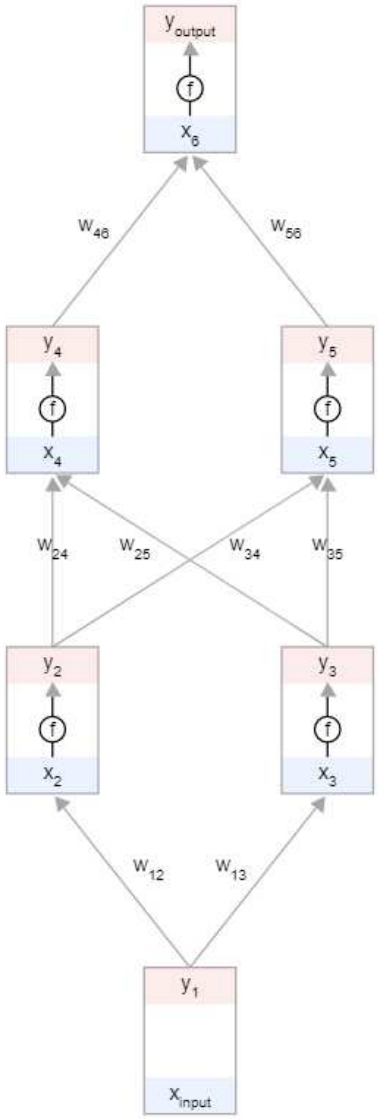
On the left you see a neural network with one input, one output node and two hidden layers of two nodes each.

Nodes in neighboring layers are connected with weights w_{ij}, which are the network parameters.

Activation function

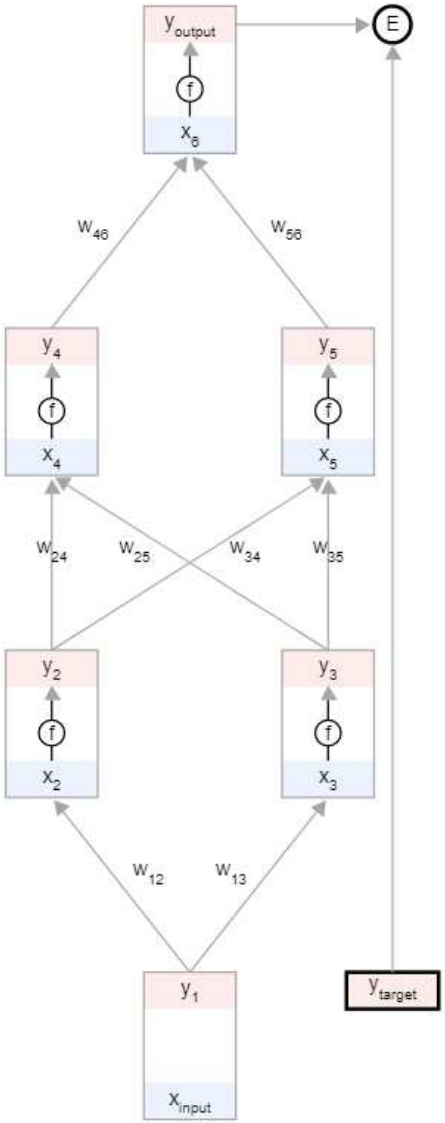
Each node has a total input x, an activation function $f(x)$ and an output $y = f(x)$. $f(x)$ has to be a non-linear function, otherwise the neural network will only be able to learn linear models.

A commonly used activation function is the Sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$.



https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/

Back Propagation



Error function

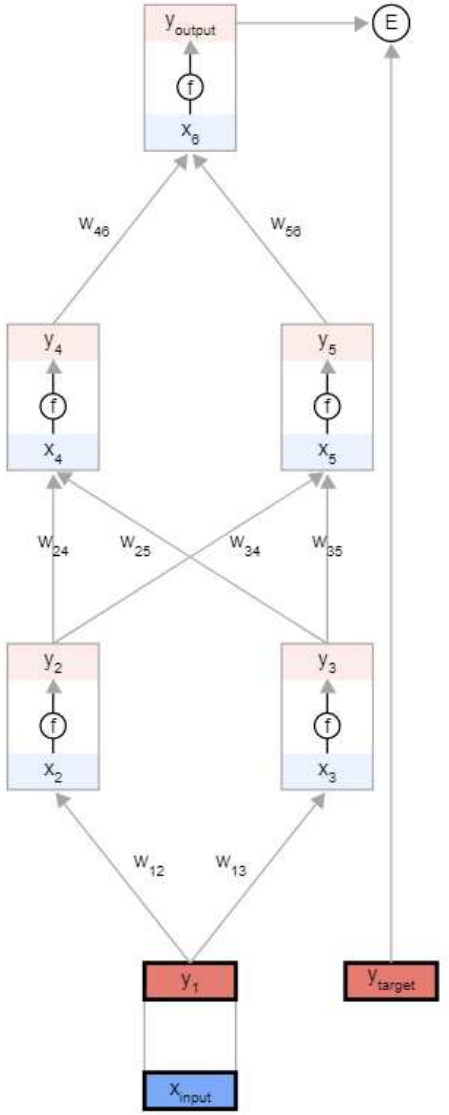
The goal is to learn the weights of the network automatically from data such that the predicted output y_{output} is close to the target y_{target} for all inputs x_{input}.

To measure how far we are from the goal, we use an error function E. A commonly used error functon is $E(y_{output}, y_{target}) = \frac{1}{2}(y_{output} - y_{target})^2$.

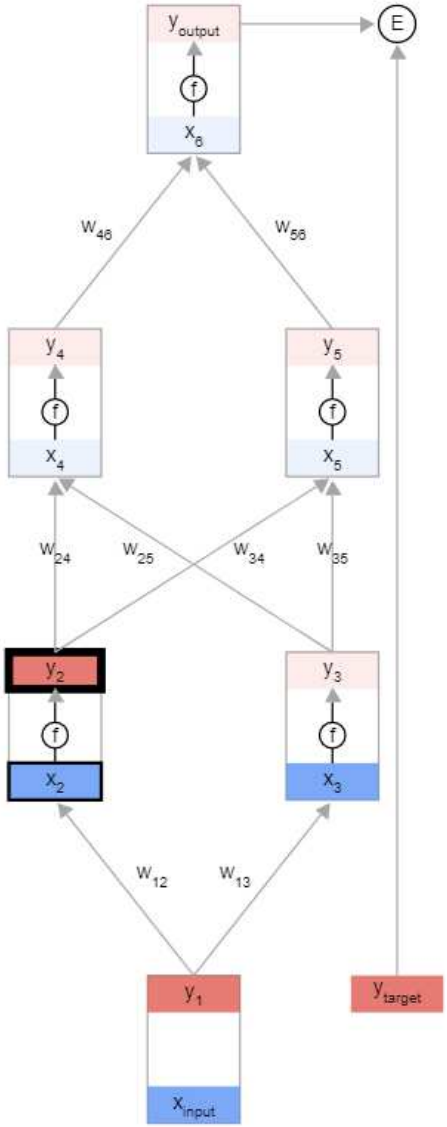
Forward propagation

We begin by taking an input example (x_{input}, y_{target}) and updating the input layer of the network.

For consistency, we consider the input to be like any other node but without an activation function so its output is equal to its input, i.e. $y_1 = x_{input}$.



Back Propagation



Forward propagation

Now, we update the first hidden layer. We take the output y of the nodes in the previous layer and use the weights to compute the input x of the nodes in the next layer.

$$x_j = \sum_{i \in in(j)} w_{ij} y_i + b_j$$

Forward propagation

Then we update the output of the nodes in the first hidden layer. For this we use the activation function, $f(x)$.

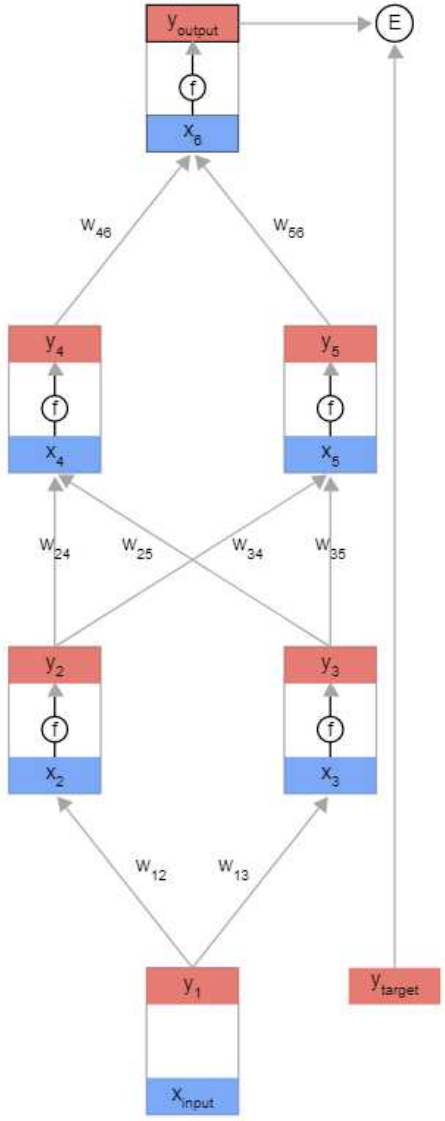
$$y = f(x)$$

Forward propagation

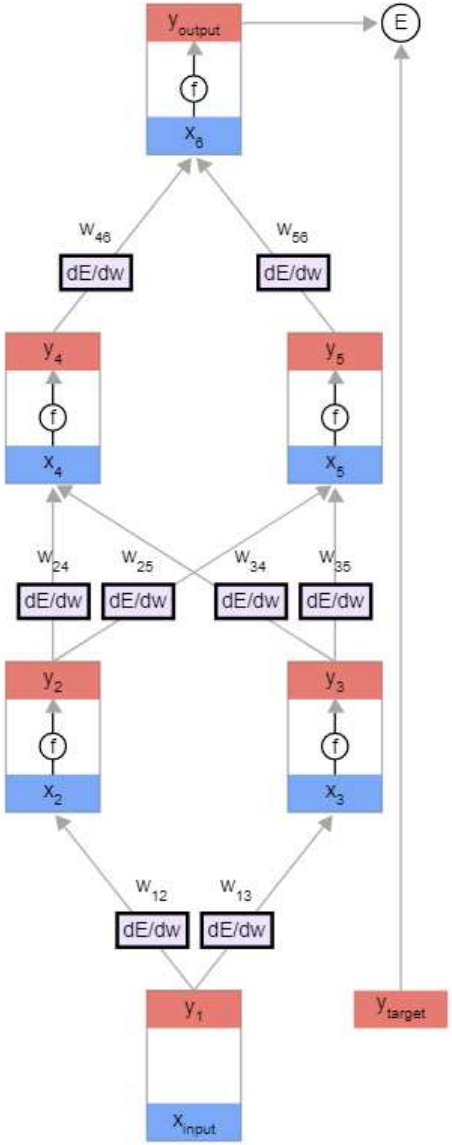
Using these 2 formulas we propagate for the rest of the network and get the final output of the network.

$$y = f(x)$$

$$x_j = \sum_{i \in in(j)} w_{ij} y_i + b_j$$



Back Propagation



Error derivative

The backpropagation algorithm decides how much to update each weight of the network after comparing the predicted output with the desired output for a particular example. For this, we need to compute how the error changes with respect to each weight $\frac{dE}{dw_{ij}}$.

Once we have the error derivatives, we can update the weights using a simple update rule:

$$w_{ij} = w_{ij} - \alpha \frac{dE}{dw_{ij}}$$

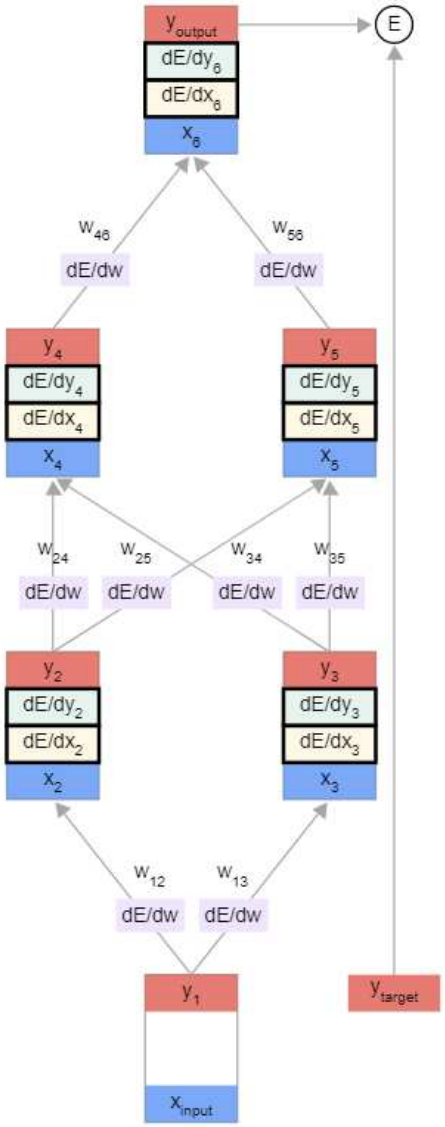
where α is a positive constant, referred to as the learning rate, which we need to fine-tune empirically.

[Note] The update rule is very simple: if the error goes down when the weight increases ($\frac{dE}{dw_{ij}} < 0$), then increase the weight, otherwise if the error goes up when the weight increases ($\frac{dE}{dw_{ij}} > 0$), then decrease the weight.

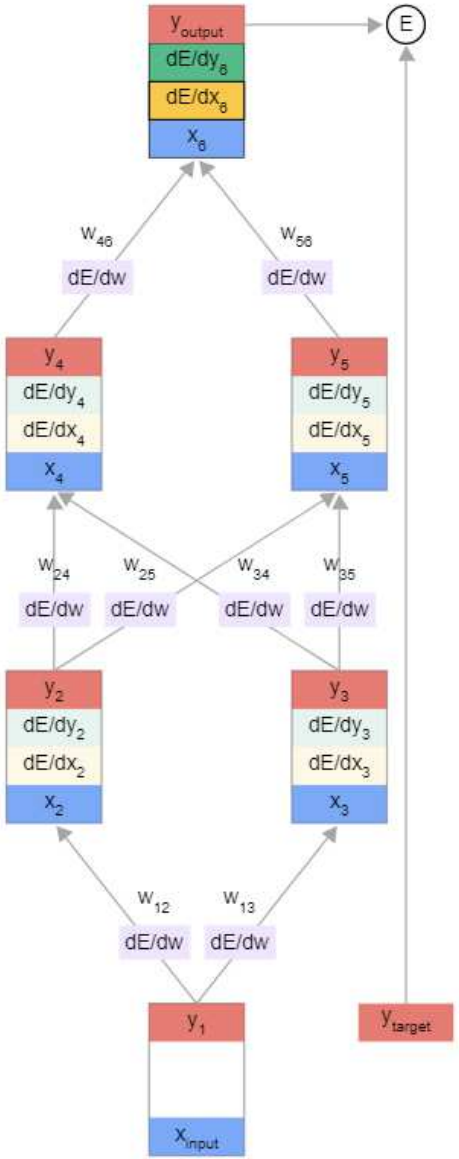
Additional derivatives

To help compute $\frac{dE}{dw_{ij}}$, we additionally store for each node two more derivatives: how the error changes with:

- the total input of the node $\frac{dE}{dx}$ and
- the output of the node $\frac{dE}{dy}$.



Back Propagation



Back propagation

Let's begin backpropagating the error derivatives. Since we have the predicted output of this particular input example, we can compute how the error changes with that output. Given our error function $E = \frac{1}{2}(y_{output} - y_{target})^2$ we have:

$$\frac{\partial E}{\partial y_{output}} = y_{output} - y_{target}$$

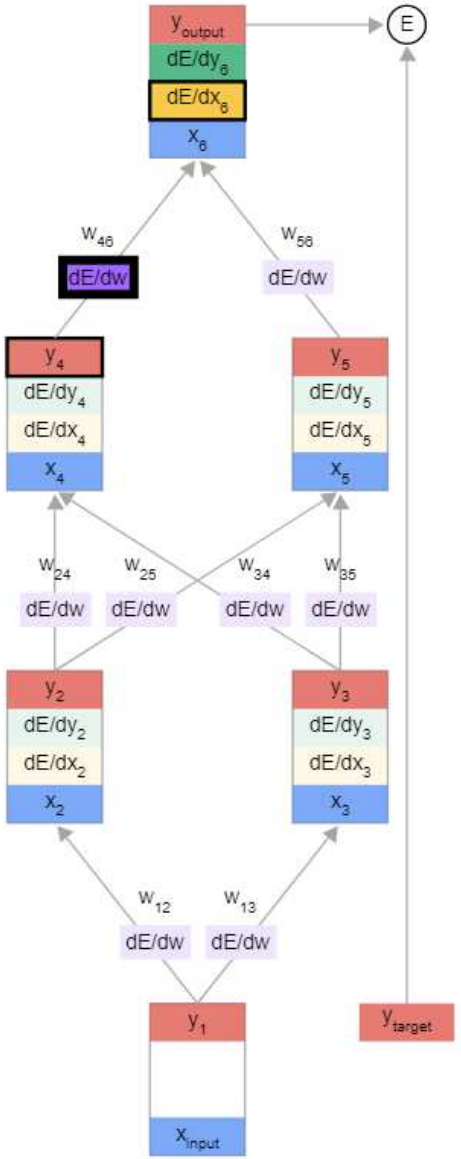
Now that we have $\frac{dE}{dy}$ we can get $\frac{dE}{dx}$ using the chain rule.

$$\frac{\partial E}{\partial x} = \frac{dy}{dx}\frac{\partial E}{\partial y} = \frac{d}{dx}f(x)\frac{\partial E}{\partial y}$$

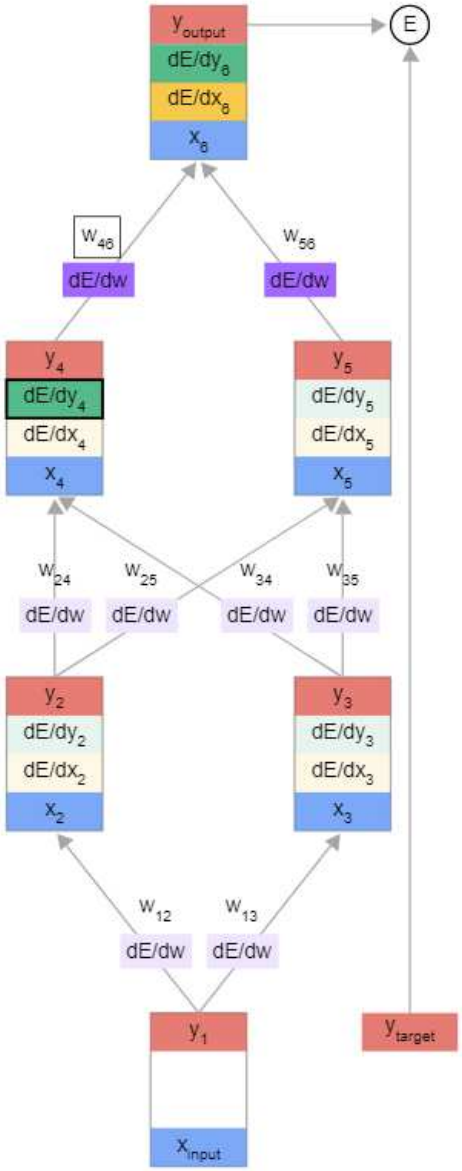
where $\frac{d}{dx}f(x) = f(x)(1 - f(x))$ when $f(x)$ is the Sigmoid activation function.

As soon as we have the error derivative with respect to the total input of a node, we can get the error derivative with respect to the weights coming into that node.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial x_j}{\partial w_{ij}}\frac{\partial E}{\partial x_j} = y_i\frac{\partial E}{\partial x_j}$$



Back Propagation



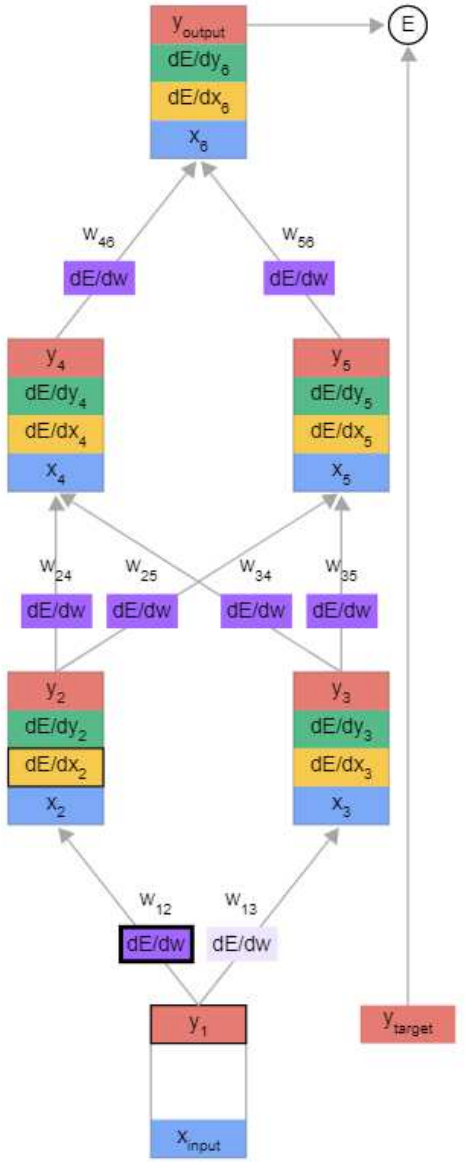
Back propagation

And using the chain rule, we can also get $\frac{dE}{dy}$ from the previous layer. We have made a full circle.

$$\frac{\partial E}{\partial y_i} = \sum_{j \in out(i)} \frac{\partial x_j}{\partial y_i} \frac{\partial E}{\partial x_j} = \sum_{j \in out(i)} w_{ij} \frac{\partial E}{\partial x_j}$$

Back propagation

All that is left to do is repeat the previous three formulas until we have computed all the error derivatives.



Resilient Back Propagation

- Rprop, abreviação de retropropagação resiliente, é uma heurística de aprendizagem para redes neuronais artificiais
- O algoritmo foi criado por Martin Riedmiller (atualmente cientista na Google Deepmind) e Heinrich Braun em 1992
- O Rprop leva em conta apenas o sinal da derivada parcial sobre todos os padrões (não a magnitude), e age independentemente em cada "peso".
- Para cada peso, se houve uma mudança de sinal da derivada parcial da função de erro total em comparação com a última iteração, o valor de atualização para esse peso é multiplicado por um fator η^-, onde $\eta^- < 1$
- Se a última iteração produziu o mesmo sinal, o valor de atualização é multiplicado por um fator de η^+, onde $\eta^+ > 1$
- Os valores de atualização são calculados para cada peso da maneira indicada acima e, finalmente, cada peso é alterado pelo seu próprio valor de atualização, na direção oposta à derivada parcial desse peso, de modo a minimizar a função de erro total
- Valores por defeito: η^+ é empiricamente ajustado para 1.2 e η^- para 0.5
- O Rprop é ainda considerado um dos mecanismos de atualização de pesos para treino de redes neuronais

Neural Network Tutorials/Demos

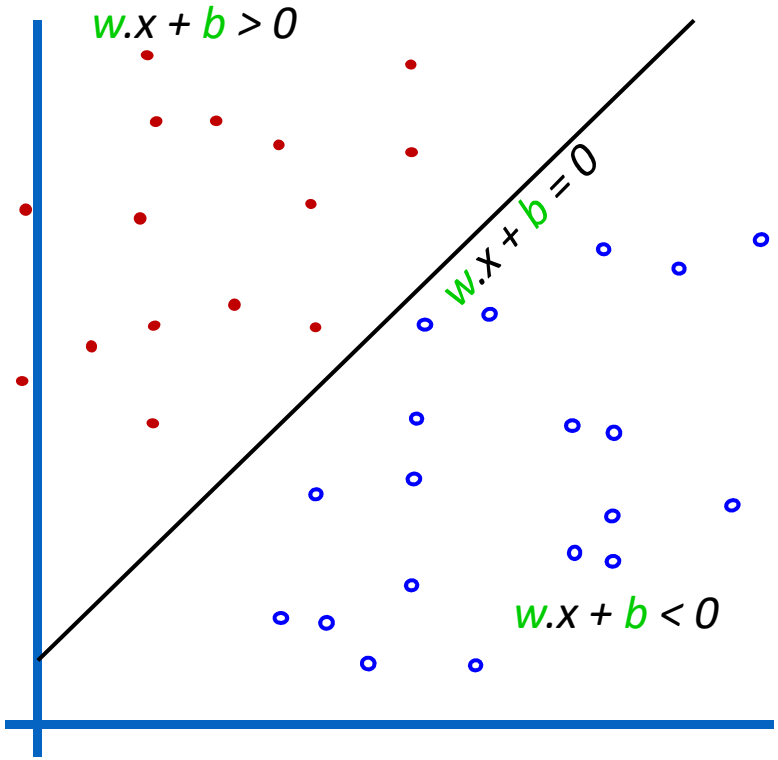
- Neural Network Playground:
  - https://playground.tensorflow.org/

Máquinas de Suporte Vetorial

- Técnica de classificação
- Raízes na teoria de aprendizagem estatística
- Várias aplicações práticas com bons resultados empíricos
- Evita o problema causado pelo aumento do volume de dados

Máquinas de Suporte Vetorial

- Classificador Linear – **problema de classificação binária**



Fronteira de Decisão Linear

N exemplos no conjunto de treino

(x_i, y_i) – exemplos

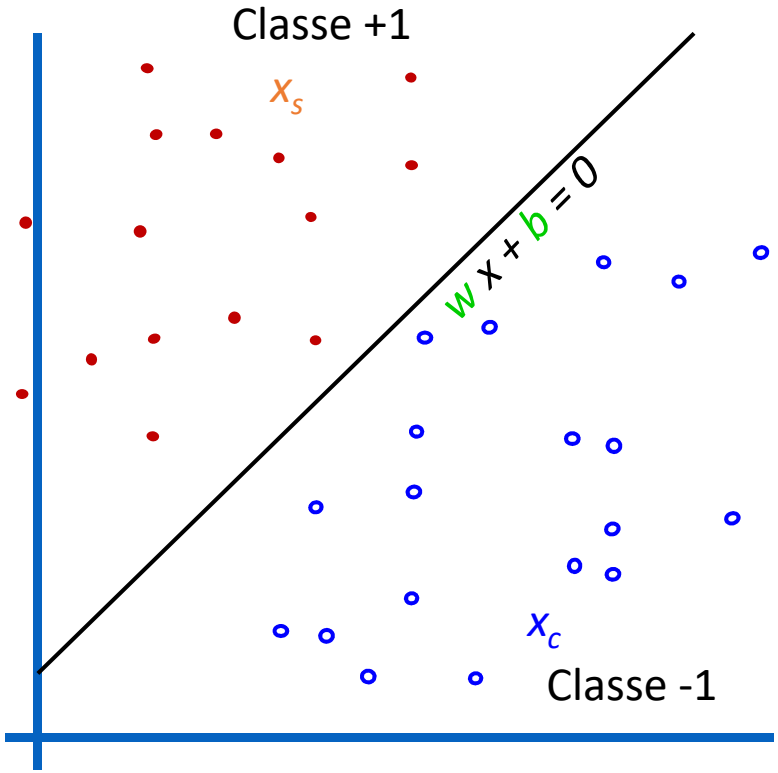
x_i – atributos

y_i – classe $\in \{-1,1\}$

w e *b* parâmetros do modelo

Máquinas de Suporte Vetorial

- Classificador Linear – **problema de classificação binária**



$w.x_s + b = k$ com $k > 0$

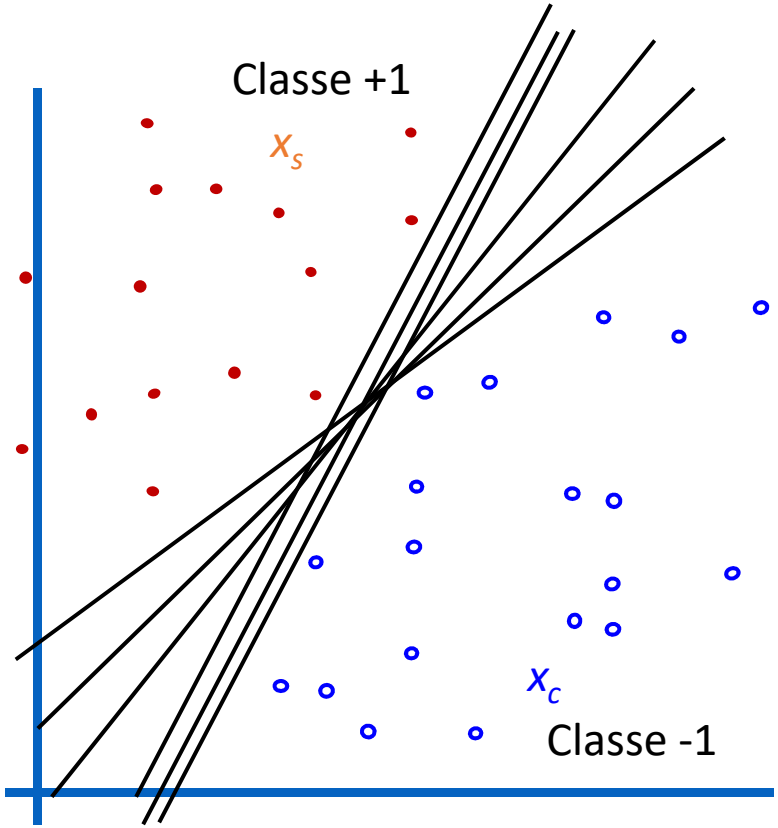
$w.x_c + b = k'$ com $k' < 0$

Como será classificado o exemplo z?

$$y = \begin{cases} 1 & se\ w \cdot z + b > 0 \\ -1 & se\ w \cdot z + b < 0 \end{cases}$$

Máquinas de Suporte Vetorial

- Classificador Linear – **problema de classificação binária**

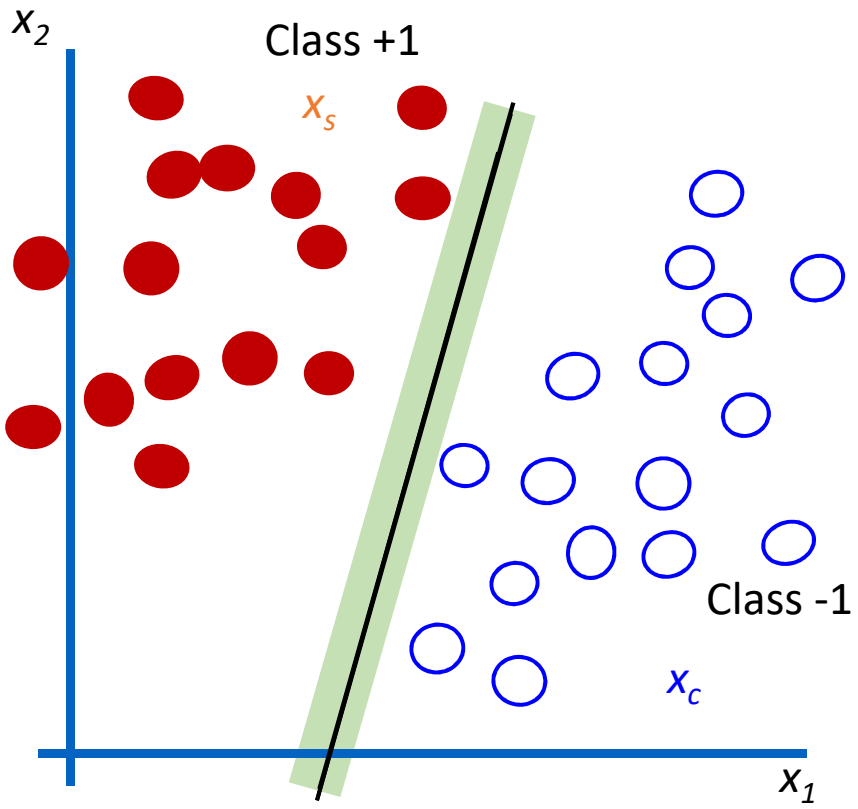


Existem infinitos hiperplanos com erro de treino igual a 0

Qual a melhor fronteira de decisão?

Máquinas de Suporte Vetorial

- Classificador Linear – **problema de classificação binária**

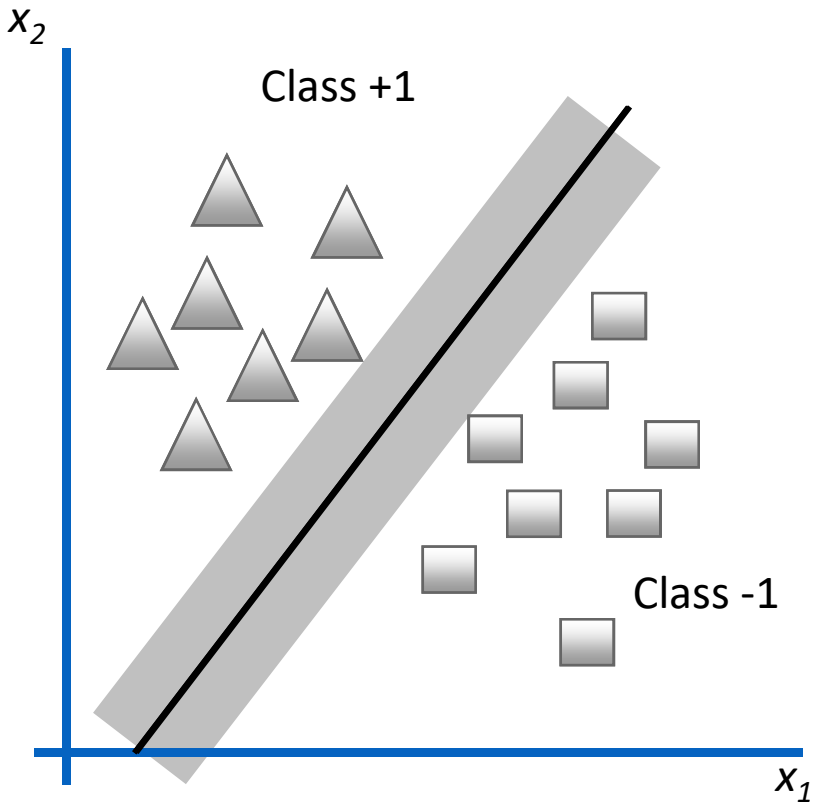


Situação A

Os exemplos nas margens designam-se vectores de suporte

Máquinas de Suporte Vetorial

- Classificador Linear – **problema de classificação binária**



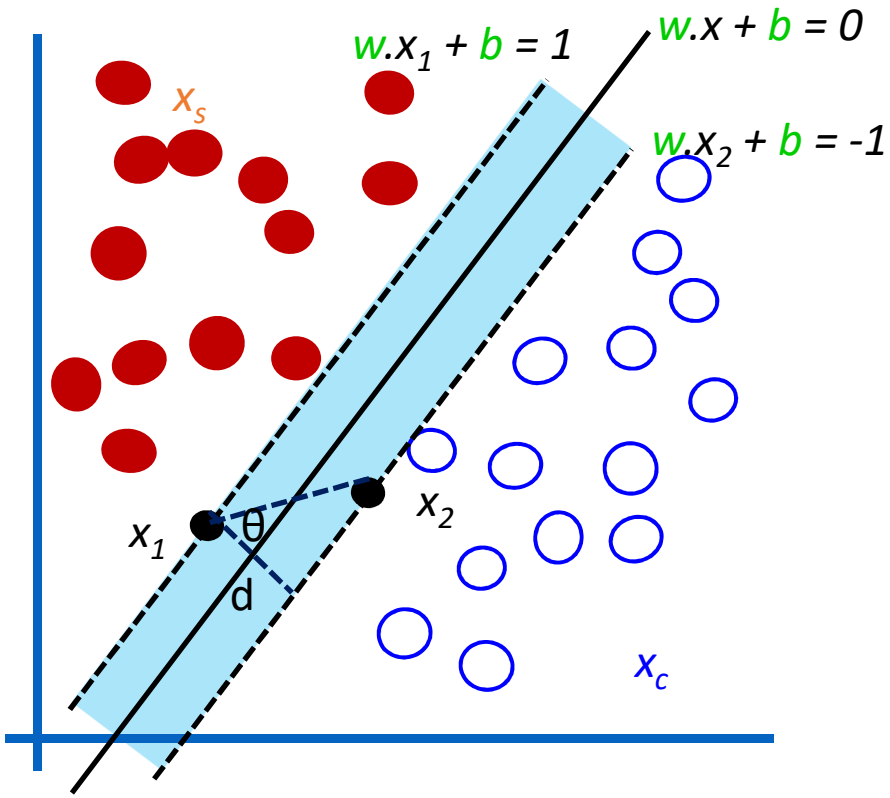
Situação B

Máquinas de Suporte Vetorial

- **Fronteira de Decisão e Margens**
  - Margens pequenas
    - qualquer perturbação na fronteira de decisão poderá ter impacto na classificação
    - mais susceptíveis de existir *overfitting*
    - fraco poder de generalização para os novos exemplos
  - Margens grandes
    - Tendência para melhores erros de generalização
    - *Structural risk minimization* – à medida que a capacidade do modelo linear aumenta a generalização do erro também aumenta

Máquinas de Suporte Vetorial

- **Fronteira de Decisão e Margens**



Assim

$$w \cdot (x_1 - x_2) = 2$$

Recordar

$$\cos\theta = \frac{w.(x_1 - x_2)}{\|w\| \times \|x_1 - x_2\|}$$

$$\cos\theta = \frac{d}{\|x_1 - x_2\|}$$

Portanto

$$d = \frac{2}{\|w\|}$$

Máquinas de Suporte Vetorial

- Aprendizagem de MSV Linear
  - Estimar os parâmetros *w* e *b* da fronteira de decisão a partir do conjunto de treino de forma a obter o hiperplano com margem máxima

$$Maximizar \quad d = \frac{2}{\|w\|}$$

  - Sendo satisfeitas as condições
    - $w.x_i + b \geq 1 \; se \; y_i = 1$
    - $w.x_i + b \leq -1 \; se \; y_i = -1$

$$y_i \, (w.x_i + b) \geq 1 \qquad (i = 1,2,\ldots, N)$$

Máquinas de Suporte Vetorial

- Aprendizagem de MSV Linear
  - Maximar a margem é equivalente a minimizar a seguinte função objectivo

$$f(w) = \frac{\|w\|^2}{2}$$

  - Formalização do problema de optimização

$$\min_{w} \frac{\|w\|^2}{2} \quad \text{sujeito a} \quad y_i\,(w.x_i + b) \geq 1, \quad i = 1,2,\ldots, N$$

  - Resolver utilizando o método dos Multiplicadores de Lagrange

Máquinas de Suporte Vetorial

- Aprendizagem de MSV Linear
  - A nova função objectivo denominada Lagrangiano

$$Lp = \frac{1}{2}\|\mathrm{w}\|^2 - \sum_{i=1}^{N} \lambda_i \left(y_i \left(\mathrm{w} \cdot \mathrm{x}_\mathrm{i} + b \right) - 1 \right)$$

  - Para minimizar tem-se que

$$\frac{\partial Lp}{\partial \mathrm{w}} = 0 \Rightarrow \mathrm{w} = \sum_{i=1}^{N} \lambda_i y_i \mathrm{x}_\mathrm{i}$$

$$\frac{\partial Lp}{\partial \mathrm{b}} = 0 \Rightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

Máquinas de Suporte Vetorial

- Aprendizagem de MSV Linear
  - Como os multiplicadores de Lagrange são desconhecidos
    - Recorrer às condições de optimilidade de 1ª ordem (KKT – Karush-Kuhn-Tucker)

$$\lambda_i \geq 0$$
$$\lambda_i \left[y_i \left(w \cdot x_i + b \right) - 1 \right] = 0$$

  - Formalização do problema dual

$$L_D = \sum \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathrm{x}_\mathrm{i} \cdot \mathrm{x}_\mathrm{j}$$

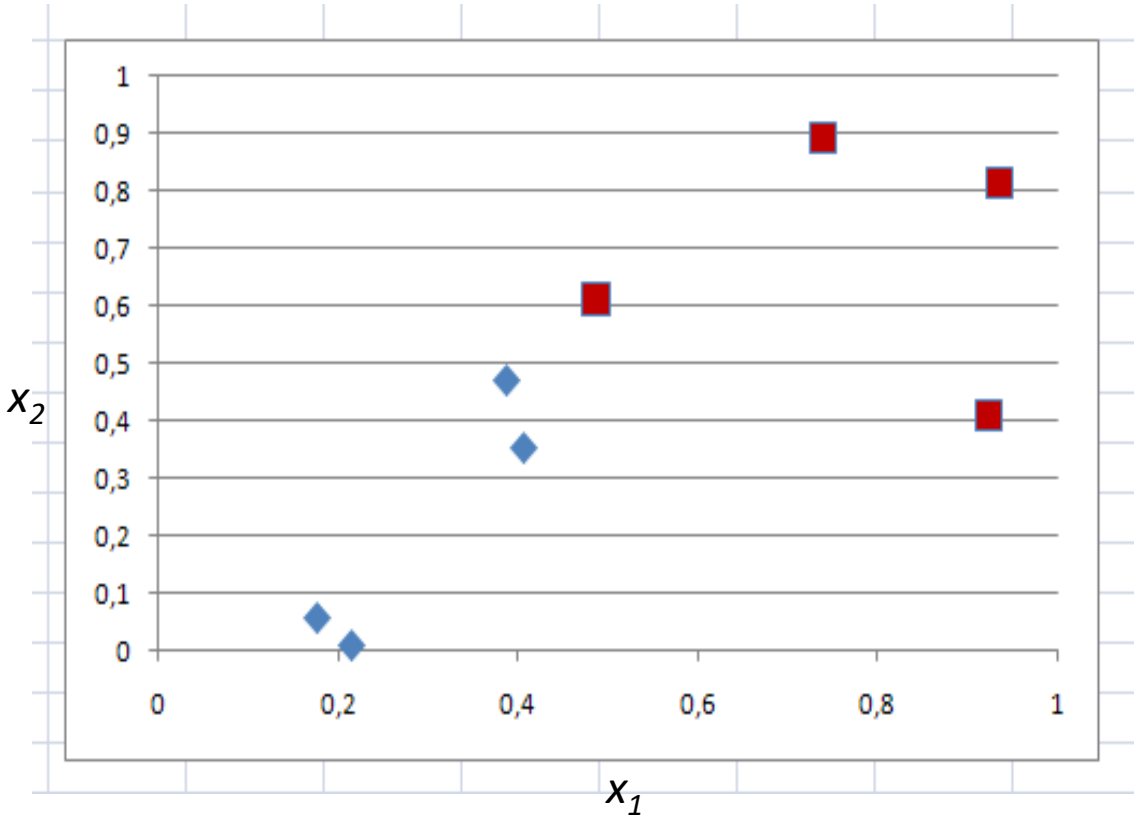
Máquinas de Suporte Vetorial

- Aprendizagem de MSV Linear
- Diferenças entre Lagrangianos primais e duais
  - Formulação Dual recorre
    - Multiplicadores de Lagrange
    - Dados de treino
  - A mudança de sinal transformou um problema de minimização de L_P num problema de maximização de L_D
  - As soluções dos dois problemas de optimização são equivalentes

Máquinas de Suporte Vetorial

- EXEMPLO

X1	X2	Y
0,3858	0,4687	1
0,4871	0,611	-1
0,9218	0,4103	-1
0,7382	0,8936	-1
0,1763	0,0579	1
0,4057	0,3529	1
0,9355	0,8132	-1
0,2146	0,0099	1

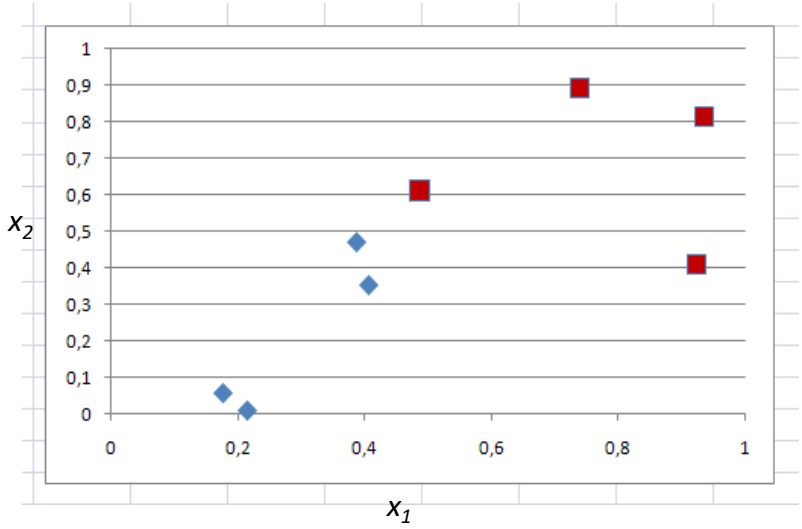


Encontrar W e b

Máquinas de Suporte Vetorial

- EXEMPLO

X1	X2	Y	λi
0,3858	0,4687	1	65,5261
0,4871	0,611	-1	65,5261
0,9218	0,4103	-1	0
0,7382	0,8936	-1	0
0,1763	0,0579	1	0
0,4057	0,3529	1	0
0,9355	0,8132	-1	0
0,2146	0,0099	1	0



- $w = (w_1, w_2)$

$$w_1 = \sum_{i=1}^{8} \lambda_i y_i x_{i1} = 65{,}5621 \times 1 \times 0{,}3858 + 65{,}5621 \times (-1) \times 0{,}4871 = -6{,}64$$

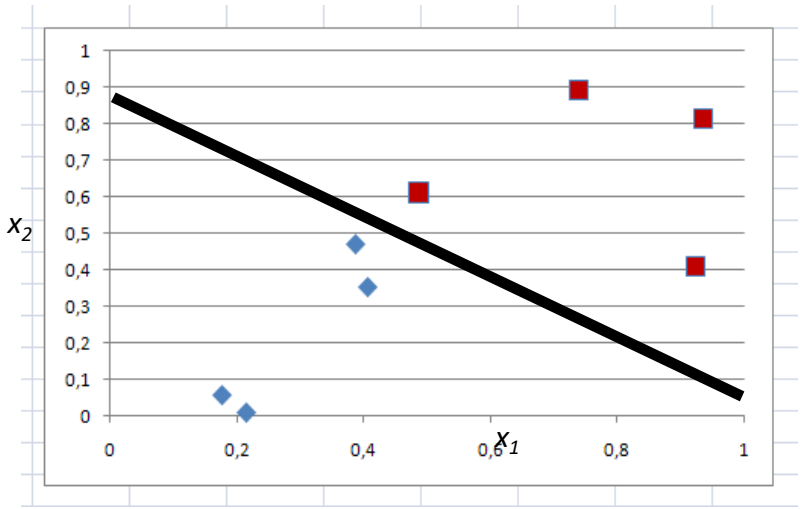
$$w_2 = \sum_{i=1}^{8} \lambda_i y_i x_{i2} = 65{,}5621 \times 1 \times 0{,}4687 + 65{,}5621 \times (-1) \times 0{,}611 = -9{,}32$$

Máquinas de Suporte Vetorial

- EXEMPLO

$-6{,}64\ x_1\ -9{,}32\ x_2 + 7{,}93 = 0$

X1	X2	Y	λi
0,3858	0,4687	1	65,5261
0,4871	0,611	-1	65,5261
0,9218	0,4103	-1	0
0,7382	0,8936	-1	0
0,1763	0,0579	1	0
0,4057	0,3529	1	0
0,9355	0,8132	-1	0
0,2146	0,0099	1	0



- O termo b pode ser calculado para cada vector de suporte

$$b^{(1)} = 1 - \mathrm{w} \cdot \mathrm{x}_1 = 1 - (-6{,}64)(0{,}3858) - (-9{,}32)(0{,}4687) = 7{,}9300$$

$$b^{(2)} = -1 - \mathrm{w} \cdot \mathrm{x}_2 = -1 - (-6{,}64)(0{,}4871) - (-9{,}32)(0{,}611) = 7{,}9289$$

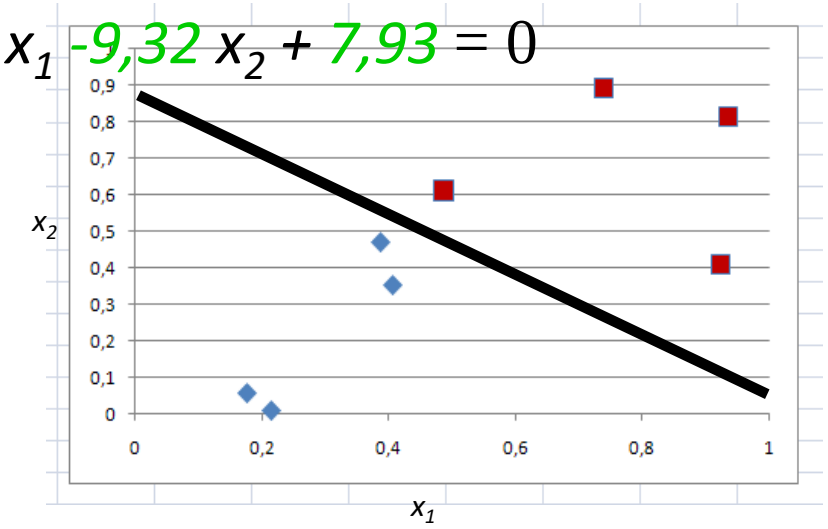
média *7,93*

Máquinas de Suporte Vetorial

- EXEMPLO

X1	X2	Y	λi
0,3858	0,4687	1	65,5261
0,4871	0,611	-1	65,5261
0,9218	0,4103	-1	0
0,7382	0,8936	-1	0
0,1763	0,0579	1	0
0,4057	0,3529	1	0
0,9355	0,8132	-1	0
0,2146	0,0099	1	0

$-6{,}64\, x_1 - 9{,}32\, x_2 + 7{,}93 = 0$



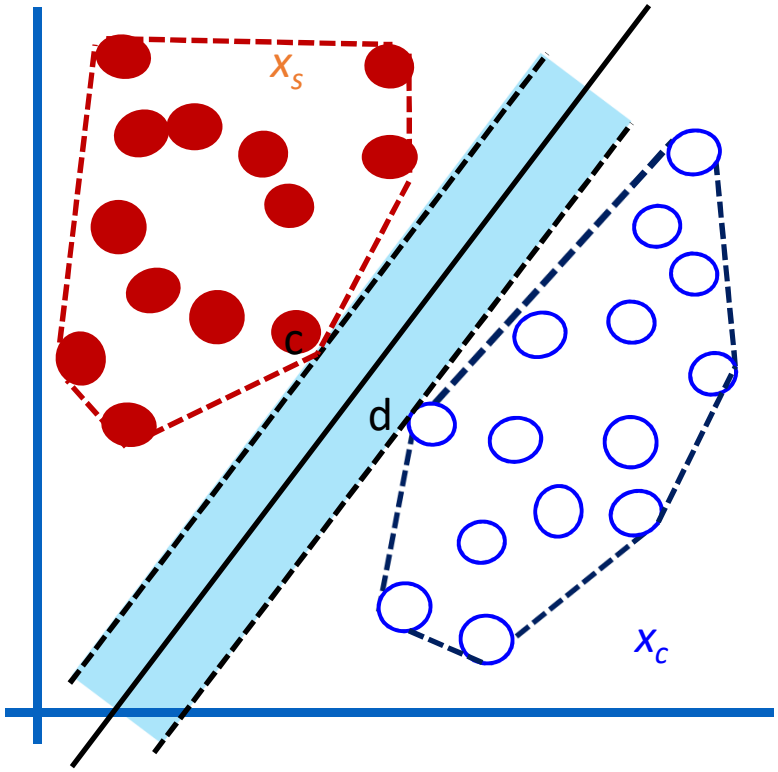
- O exemplo z é classificado seguindo

$$f(z) = sign(\mathrm{w} \cdot \mathrm{z} + \mathrm{b})$$

- Se f(z) = 1 então é classificado como 1
- Se f(z) = -1 então é classificado como -1

Máquinas de Suporte Vetorial

- Outra formalização do problema
  - A melhor fronteira de decisão bissecta os pontos mais próximos dos *Convex Hulls*



$$\min_{\lambda} \quad \frac{\|c-d\|^2}{2}$$

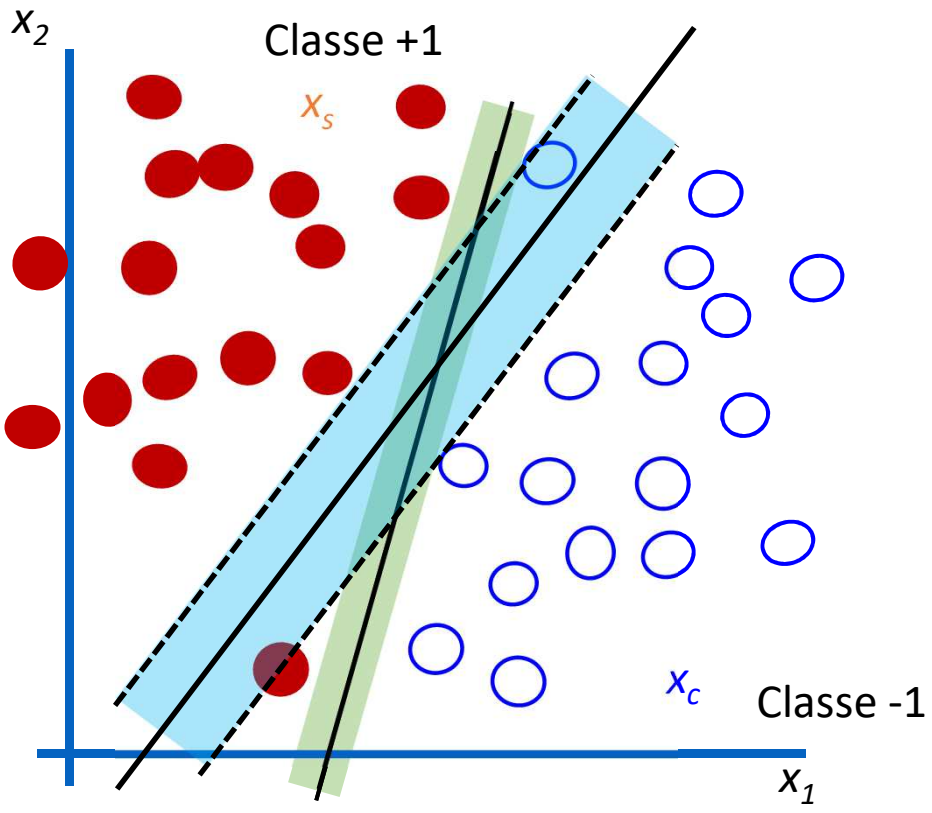
$$c = \sum_{y_i \in Classe1} \lambda_i \mathrm{x}_\mathrm{i} \qquad d = \sum_{y_i \in Classe-1} \lambda_i \mathrm{x}_\mathrm{i}$$

$$\sum_{y_i \in Classe1} \lambda_i = 1 \qquad \sum_{y_i \in Classe-1} \lambda_i = 1$$

$$\lambda_i \geq 0 \qquad i = 1,\ldots,N$$

Máquinas de Suporte Vetorial

- Classificador Linear – **caso não separável**

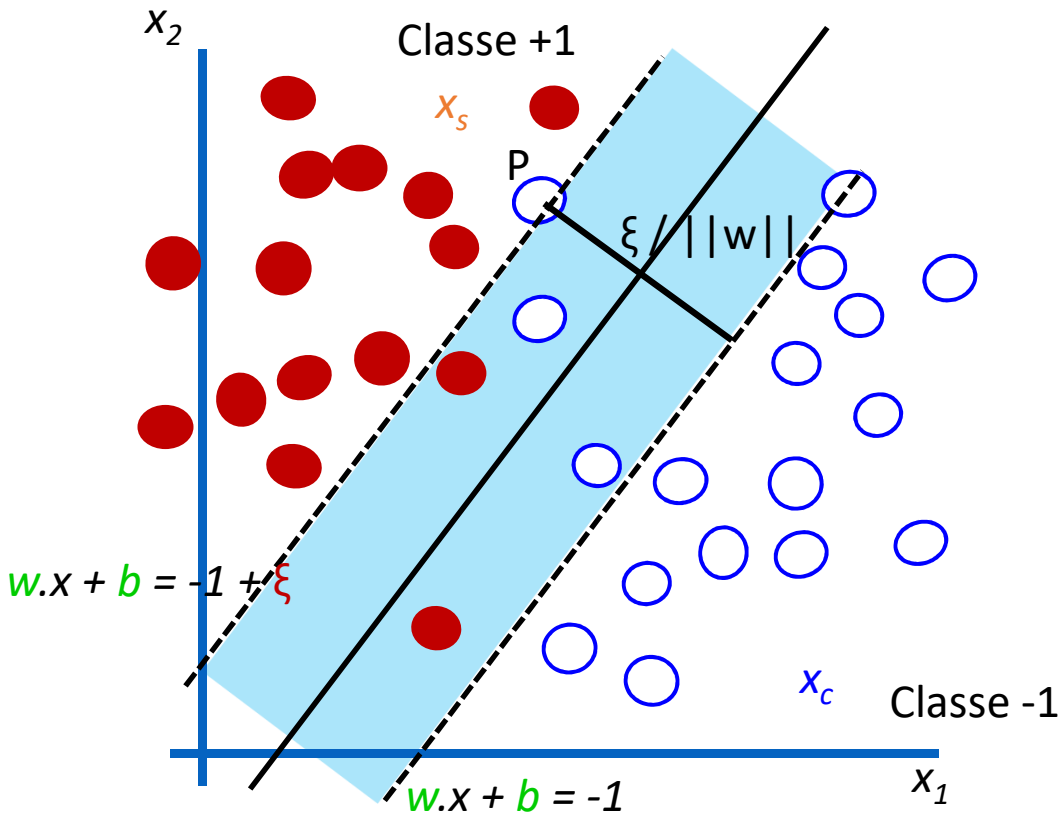


Abordagem Soft Margin

Compromisso entre a largura da margem e nº de erros cometidos pela escolha da fronteira de decisão

Máquinas de Suporte Vetorial

- Classificador Linear – **caso não separável**



Slack Variables ($\xi > 0$)

Variáveis de Folga

$$w.x_i + b \geq 1 - \xi_i \quad se\ y_i = 1$$

$$w.x_i + b \leq -1 + \xi_i \quad se\ y_i = -1$$

Máquinas de Suporte Vetorial

- ξ fornece uma estimativa de erro da fronteira de decisão no exemplo de treino P
- Modificar a função objectivo para penalizar a fronteira de decisão com valores elevados de variáveis de folga

$$f(w) = \frac{\|w\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)^k$$

- C poderá ser escolhido baseado no desempenho do modelo no conjunto de teste assumindo k igual a 1

Máquinas de Suporte Vetorial

- Lagrangiano para este problema de optimização

$$Lp = \frac{1}{2}\|\mathrm{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\lambda_i\left(y_i\left(\mathrm{w}\cdot\mathrm{x}_\mathrm{i} + b\right) - 1 + \xi_i\right) - \sum_{i=1}^{N}\mu_i\xi_i$$

Função objectivo a ser minimizada

Restrições associados às variáveis de folga

Valores não-negativos e requisitos dos ξ_i

Máquinas de Suporte Vetorial

- O dual deste problema de optimização será equivalente ao Lagrangiano dual para o caso linear separável

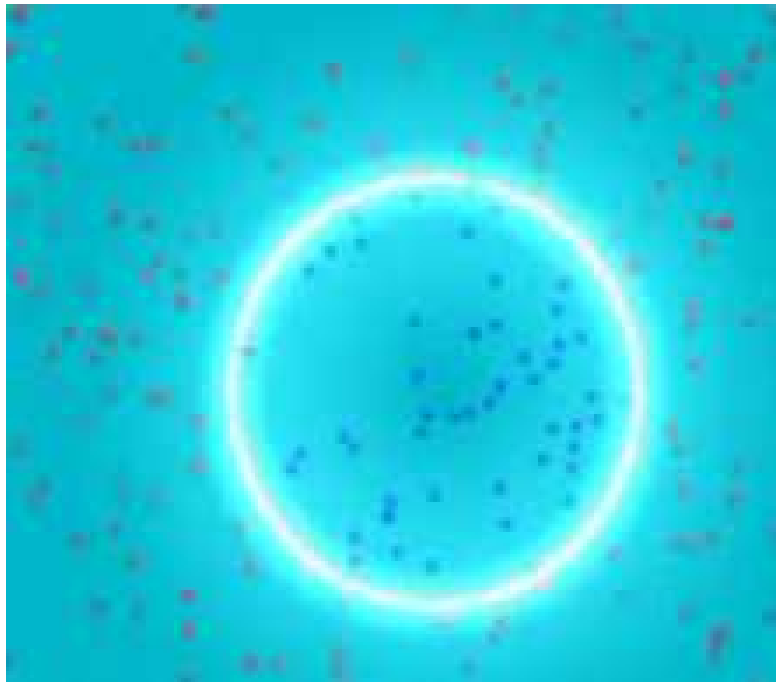
$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \mathrm{x}_\mathrm{i} \cdot \mathrm{x}_\mathrm{j}$$

- As restrições serão diferentes
  - Os multiplicadores de Lagrange para o caso linear não separável terão de satisfazer

$$0 \leq \lambda_i \leq C$$

Máquinas de Suporte Vetorial

- Classificador Não Linear

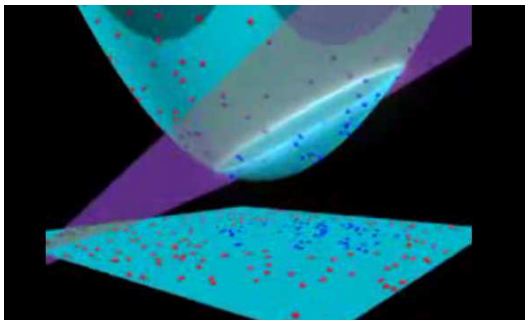


https://youtu.be/3liCbRZPrZA

Objectivo será transformar os dados com coordenadas do espaço original X para um novo espaço $\Phi(X)$ para obter uma fronteira de decisão linear

Máquinas de Suporte Vetorial

- Classificador Não Linear - Problemas
  - Dimensionalidade dos dados
  - Tipo de função para mapear os dados
  - Garantir que é construída uma fronteira de decisão linear no espaço transformado
  - Computacionalmente pesado



Máquinas de Suporte Vetorial

- Classificador Não Linear
  - Formalização do problema de optimização

$$\min_{w} \frac{\|w\|^2}{2}$$

$$y_i(\mathrm{w} \cdot \Phi(\mathrm{x}_i) + b) \geq 1, \quad i = 1,2,\ldots,N$$

Máquinas de Suporte Vetorial

- Utilização da Função Kernel
  - Método para calcular semelhança no espaço transformado utilizando os dados originais
  - Torna desnecessário encontrar a função exacta de Φ
  - Satisfaz o Teorema de Mercer
    - Assegura que as funções de Kernel podem ser expressas como produto interno de dois vectores de entrada num espaço superior

Máquinas de Suporte Vetorial

- Utilização da Função Kernel
  - Teorema de Mercer

Uma função kernel pode ser expressa como

$$K(u,v) = \Phi(u) \bullet \Phi(v)$$

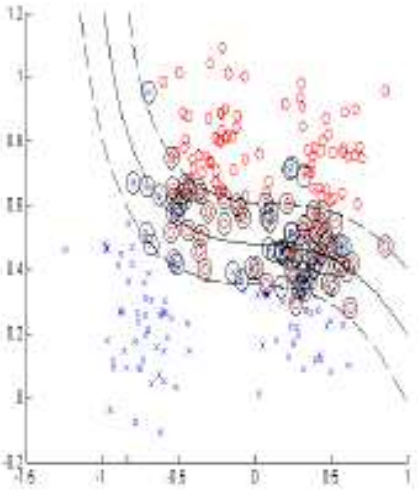
se e só se

$$\forall g(x) : \int g(x)^2 \, dx \quad \textit{é finito então}$$

$$\int K(x,y) g(x) g(y) dx dy \geq 0$$

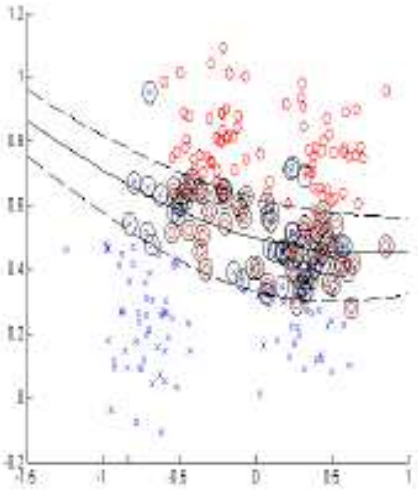
Máquinas de Suporte Vetorial

- Utilização da Função Kernel



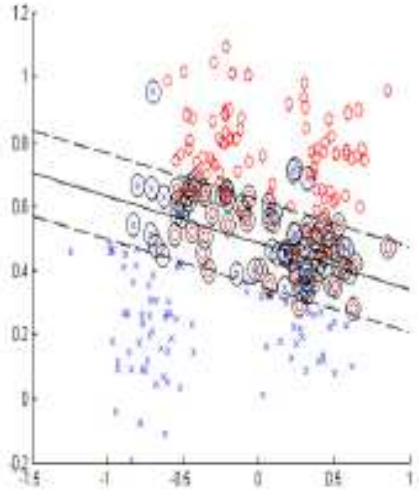
RBF kernel

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



polynomial kernel

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + c)^d$$



linear kernel

$$k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}$$

Máquinas de Suporte Vetorial

- Algoritmo/Método MSV
  - 1- Escolher
    - Parâmetro C (representa o compromisso entre minimizar o erro do conjunto de treino e maximizar a margem)
    - Função Kernel e os parâmetros
  - 2 - Resolver

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j K(\mathrm{x_i}, \mathrm{x_j})$$

ou uma formulação alternativa de MSV

  - 3 - Recuperar o limite b utilizando os vectores de suporte
  - 4 - Classificar o novo exemplo z utilizando

$$f(\mathrm{z}) = sign\left(\sum_{i=1}^{N} \lambda_i y_i K(x_i, \mathrm{z}) + b \right)$$

Máquinas de Suporte Vetorial

- Multi-Classe
  - Seja $Y = \{y_1, y_2, \ldots, y_k\}$ o conjunto de classes
  - Abordagem One-Against-Rest (1-r)
    - Decompor o problema de multi-classes em k problemas binários
    - Para cada yi, os exemplo assim classificados são considerados positivos, enquanto os restantes negativos
    - É construído um classificador binário

Máquinas de Suporte Vetorial

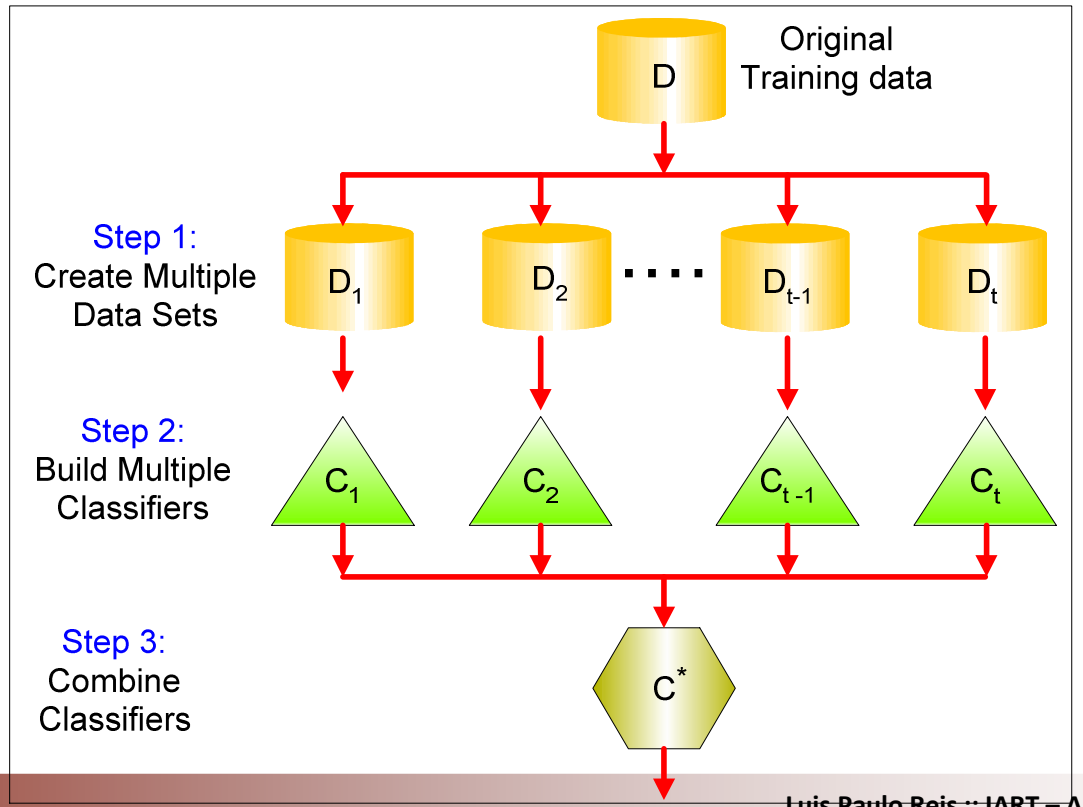
- Multi-Classe
  - Seja $Y = \{y_1, y_2, \ldots, y_k\}$ o conjunto de classes
  - Abordagem One-Against-One (1 - 1)
    - Decompor o problema em k(k-1)/2 problemas binários
    - Cada classificador distingue entre os pares de classes (y_i, y_j)
    - Classes que não pertencem a y_i ou y_j são ignoradas quando é construído o classificador binário (y_i, y_j)

Máquinas de Suporte Vetorial

- Máquinas de Suporte Vetorial - Conclusões
  - MSV é um paradigma que une a intuição geométrica, a teoria matemática e praticabilidade de algoritmos
  - As MSV podem formular o problema de aprendizagem como um problema convexo de otimização
  - MSV tendem a maximizar a margem da fronteira de decisão
  - É necessário introduzir parâmetros numa abordagem de soft margin
  - Podem ser resolvidos problemas com Multi-classes
  - Podem ser resolvidos problemas de classificação e regressão
  - Método simples de utilizar e aplicar

Métodos Ensemble

- Construir um conjunto de classificadores através do conjunto de treino
- Predizer a classe dos novos casos através das predições feitas pelos múltiplos classificadores



Bibliografia

- Tan, P., Steinbach, M. & Kumar, V. (2006). Introduction to Data Mining. Pearson Addison-Wesley.
- Adaptação de slides de “Introduction to Data Mining”, Pang-Ning Tan, Michael Steinbach, Vipin Kumar: http://www-users.cs.umn.edu/~kumar/dmbook/index.php
- Adaptação de slides de: Gladys Castillo, Aprendizagem Computacional (Machine Learning), Universidade de Aveiro, 2008
- Adaptação de slides de: B. Mónica Faria, Extração de Conhecimento, Politécnico do Porto, 2018
- Adaptação de slides de: http://www-users.cs.umn.edu/~kumar/dmbook/index.php
- Bergeron, B. (2003). Bioinformatics computing: the complete, practical guide to bioinformatics for life scientists. New Jersey: Prentice Hall.
- Santos, M. F. & Azevedo, C. (2005). Data mining: descoberta de conhecimento em bases de dados. Lisboa: FCA
- Hill M., Hill A. (2007) Investigação por Questionário, Edições Sílabo, 2ª Edição
- Maroco, J., Análise Estatística – com utilização do SPSS, Ed. Sílabo, Lda, Abril, 2003.
- Dawson-Saunders B, Trapp G (2004) Basic and Clinical Biostatistics, 4a Ed. Prentice-Hall Int. Inc
- RapidMiner: Data Science Platform, 2017, https://rapidminer.com/

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Artificial Intelligence/ Inteligência Artificial

Lecture 10: Machine Learning Algorithms

(adaptado de Faria, 2018 e Castillo 2011)

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.
Associate Professor at DEI/FEUP – Informatics Engineering Department,
Faculty of Engineering of the University of Porto, Portugal
President of APPIA – Portuguese Association for Artificial Intelligence