unisociesc
centro universitário

Aprendizagem de máquina

Inteligência artificial

Prof. Claudinei Dias
Prof. Maicol

Aprendizagem de máquina

- Uma das principais áreas de pesquisa da inteligência artificial
  - Objetiva o desenvolvimento de técnicas computacionais para
    - Construção de sistemas capazes de adquirir conhecimento automaticamente
    - Tomadas de decisão são baseadas em experiências passadas acumuladas
    - Em outras palavras, usa soluções bem sucedidas de problemas anteriores

  - Definição da aprendizagem de máquina
    - Sistema computacional que busca realizar uma tarefa **T**, aprendendo a partir de uma experiência **E** procurando melhorar uma performance **P**

- Por que é uma área tão relevante nos dias de hoje?
  - É um dos combustíveis da transformação digital
  - Poder computacional e dados em abundância

Problema de aprendizagem

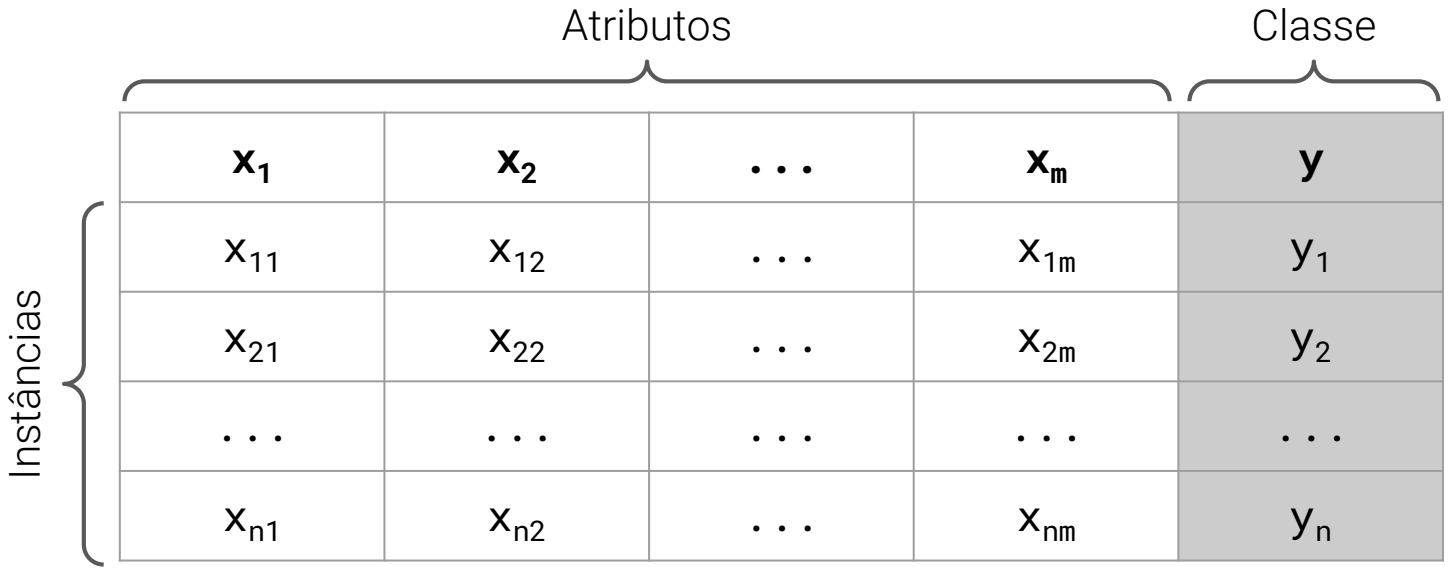
- Raciocínio por indução
  - Objetiva adquirir uma representação utilizando funções matemáticas
    - Cuja representação é geralmente aproximativa
    - Busca por uma região em um espaço de hipótese
    - Relação com otimização, analise numérica e estatística
  - Princípio da aprendizagem indutiva
    - Qualquer hipótese obtida que aproxima adequadamente da função objetivo sobre um conjunto de exemplos de treinamento suficientemente grande, também aproximará adequadamente a função objetivo sobre outros exemplos não observados

Problema de aprendizagem

- Componentes da entrada
  - Conceitos: referem-se ao objetivo do aprendizado
    - Define o que se deseja aprender
  - Instâncias: exemplos individuais e independentes de um conceito
    - Exemplos conhecidos a serem utilizados como base para o aprendizado
  - Atributos: aspectos mensuráveis de uma instância
    - Possuem tipos como binários, categóricos, numéricos, etc.
  - Classe: atributo especial que descreve o fenômeno de interesse
    - Utilizado em aprendizado supervisionado

Problema de aprendizagem

- Componentes da entrada (cont.)
  - São facilmente representados por tabelas
    - Existem diversos formatos de representação (arff, csv, etc.)



x_1	x_2	. . .	x_m	y
x_{11}	x_{12}	. . .	x_{1m}	y_1
x_{21}	x_{22}	. . .	x_{2m}	y_2
. . .	. . .	. . .	. . .	. . .
x_{n1}	x_{n2}	. . .	x_{nm}	y_n

Problema de aprendizagem

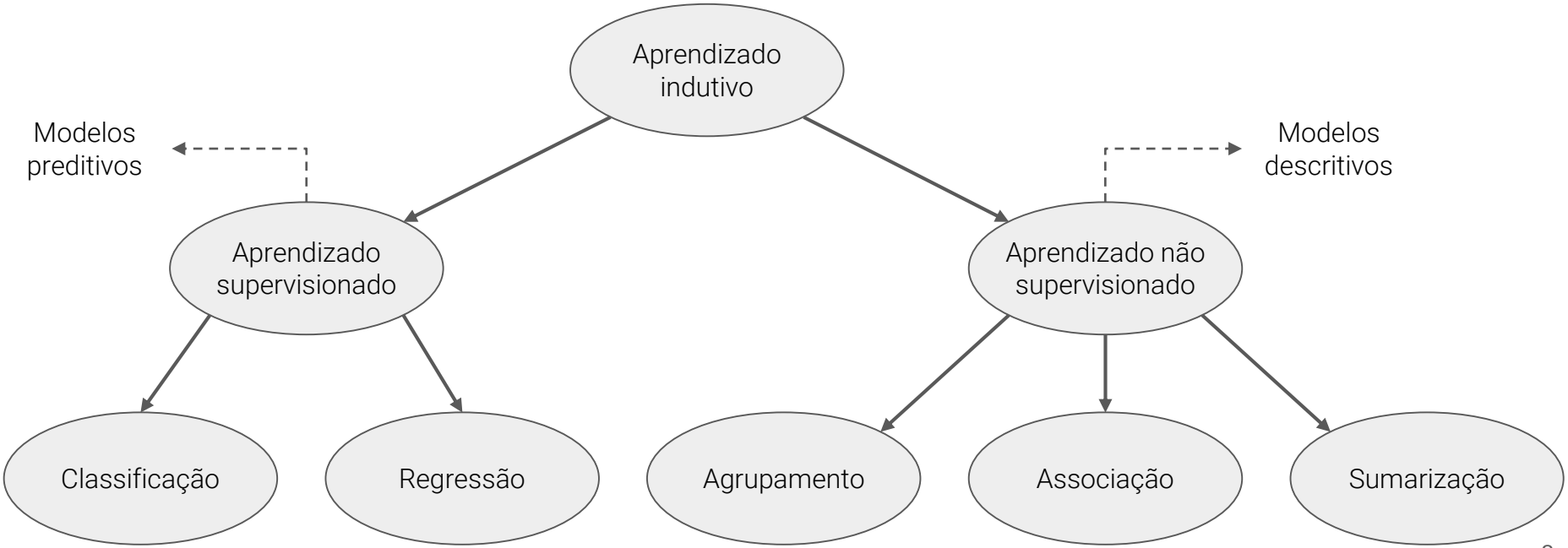
- Definição de conceitos
  - Determina a natureza do aprendizado
    - Previsão de uma classe discreta
    - Previsão de uma quantidade numérica
    - Detecção de associações ou correlações entre atributos
    - Agrupamento de instâncias similares
  - Descrição do conceito
    - Saída, isto é, o resultado do esquema de aprendizagem
  - Diferentes tipos de aprendizagem podem ser utilizados
    - Pode variar em função da definição do conceito

Problema de aprendizagem

- Principais tipos de aprendizagem
  - Aprendizagem supervisionada
    - Problemas de classificação ou regressão
    - Requer uma base com instâncias já classificadas por um especialista
  - Aprendizagem não supervisionada
    - Problemas de agrupamento ou associação, onde não há uma saída correta
    - Visa correlacionar de alguma forma as instâncias de uma base
  - Aprendizagem por reforço
    - Problema cuja saída é desconhecida
    - Visa estabelecer política de ações que maximize as recompensas recebidas

Problema de aprendizagem

- Hierarquia de aprendizagem



Aprendizagem supervisionada

- Objetivos
  - Classificar novas instâncias tão precisamente quanto possível a partir de um conjunto de instâncias previamente classificadas
    - Assume-se que o conjunto de instâncias foi classificado por um especialista
    - As instâncias já classificadas são chamadas de conjunto de treinamento

  - Exemplo de problemas reais envolvendo classificação
    - Diagnóstico de câncer a partir de exames já analisados por médicos
    - Categorização de músicas a partir de informações como autor e álbum
    - Análise de fraude em declarações de imposto de renda
    - Identificação de textos ofensivos em redes sociais
    - Reconhecimento de placas de veículos a partir de imagens ou vídeo
    - Reconhecimento facial a partir de imagens ou vídeo

Aprendizagem supervisionada

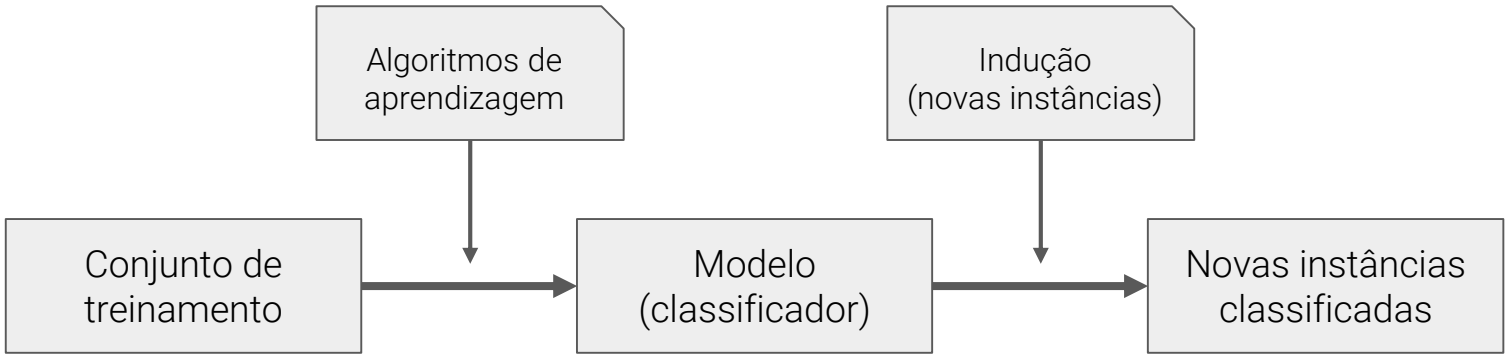
- Problemas de classificação
  - Rótulos da classe pertencem a um conjunto finito e discreto
  - Exemplo: a classe é uma categorização não ordenada

- Problemas de regressão
  - Rótulos da classe pertencem a um conjunto contínuo
  - Exemplo: a classe representa um número real

- Tarefa de classificação
  - Gerar modelos aproximados que permitam classificar novas instâncias
  - Os modelos (classificadores) são gerados por algoritmos de aprendizagem a partir do conjunto de treinamento

Aprendizagem supervisionada

- Tarefa de classificação



Aprendizagem supervisionada

- Avaliação do classificador
  - Desafio dos algoritmos
    - Distinguir entre padrões verdadeiros e ***overfitting***
    - Padrões que parecem significante
    - Porém, são aleatórios ou reproduzidos apenas no conjunto de treinamento
  - Estratégia para avaliação
    - Dividir as instâncias classificadas em treinamento e teste
    - O classificador é gerado com o conjunto de treinamento
    - É avaliada a efetividade do classificador com o conjunto de teste
    - Esta estratégia requer uma quantidade suficiente de instâncias classificadas

Aprendizagem supervisionada

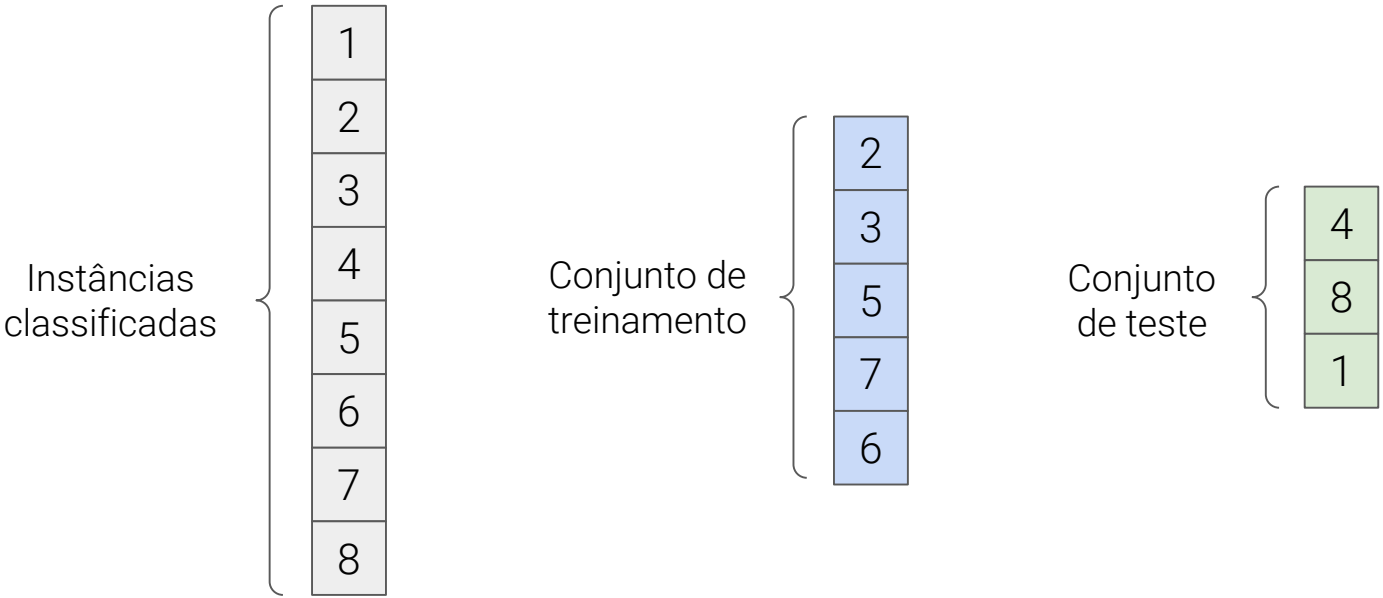
- Avaliação do classificador (cont.)
  - Dilema para geração do classificador
    - Quanto maior o conjunto de treinamento, melhor o classificador
    - Quanto maior o conjunto de teste, mais precisa é a estimativa do erro
  - Procedimentos mais populares para geração do classificador
    - ***Holdout***
    - ***Holdout*** repetido
    - Validação cruzada (***K-fold***)

Aprendizagem supervisionada

- *Holdout*
  - Estratégia do procedimento
    - Separa instâncias para teste e usa o restante para treinar
    - Em geral, usa-se 2/3 para treinamento e 1/3 para teste
    - Um desafio é que os exemplos podem não ser significativos
    - Além disso as classes podem ter distribuição de instâncias não uniforme
  - Versão ***holdout*** estratificada
    - Usa-se em casos onde há uma desigualdade entre as classes
    - Assegura que cada classe tenha proporções “iguais” entre os conjuntos

Aprendizagem supervisionada

- ***Holdout*** (cont.)

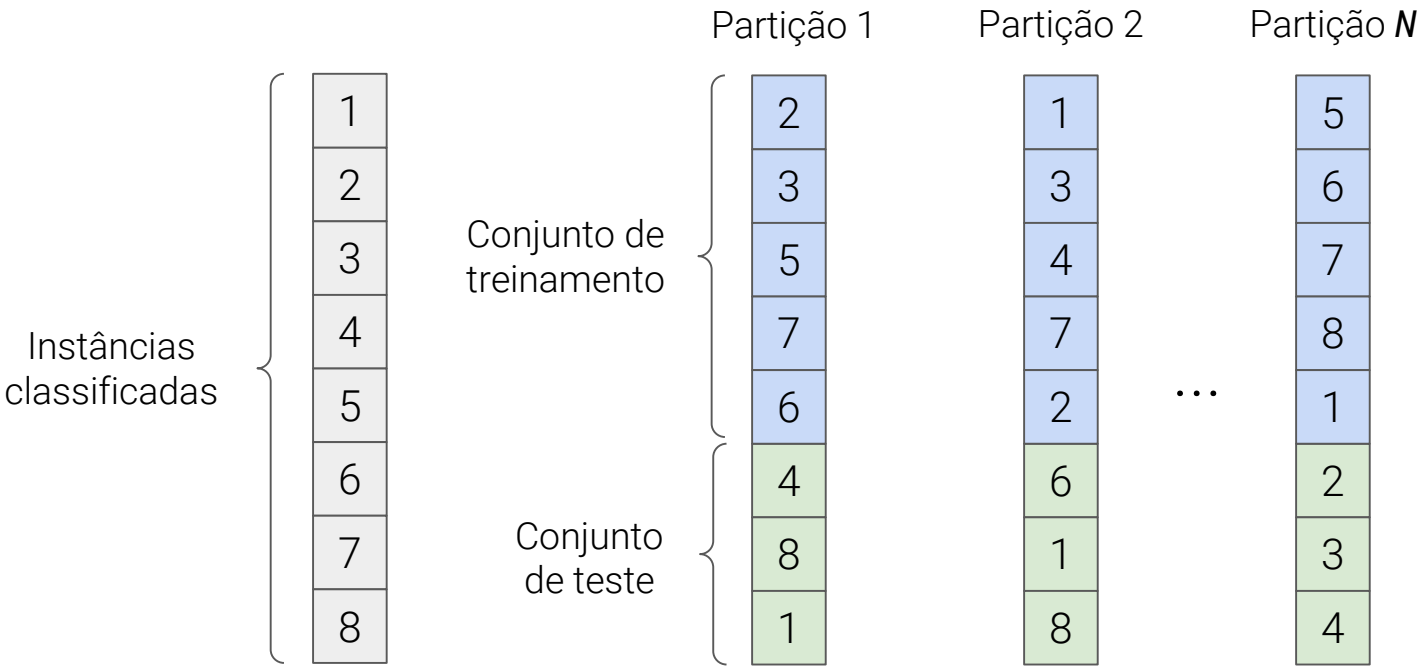


Aprendizagem supervisionada

- *Holdout* repetido
  - Estratégia do procedimento
    - Melhora o ***holdout*** repetindo o processo com diferentes subamostras
    - Cada iteração realiza uma seleção aleatória de uma parcela das instâncias
    - Esta parcela de instâncias são utilizadas o treinamento
    - Ao final das iterações calcula-se a média das taxas de erros das iterações
    - A média representa a taxa geral de erros
  - O ***holdout*** repetido não é um procedimento ótimo
    - Isto é, os diferentes conjuntos de teste podem apresentar sobreposição

Aprendizagem supervisionada

- ***Holdout*** repetido (cont.)



Aprendizagem supervisionada

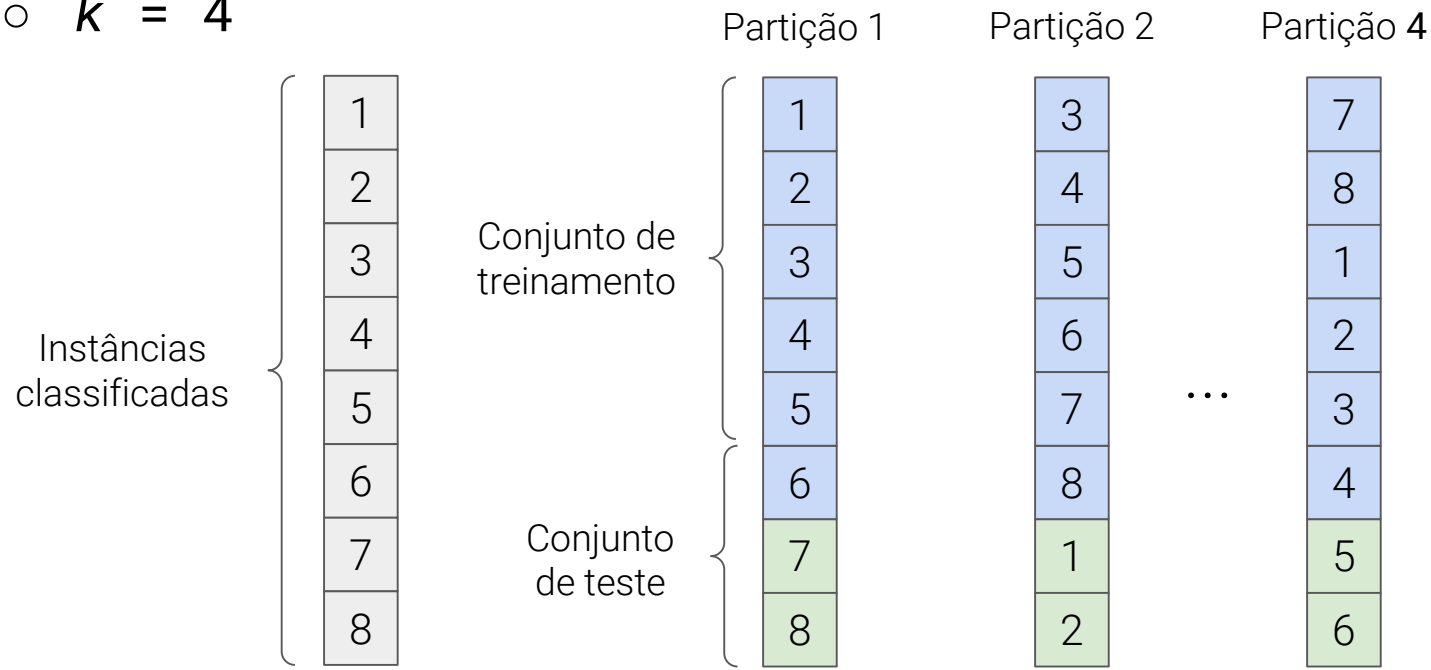
- Validação cruzada
  - Evita sobreposição nos conjuntos de teste
    1. Os dados são divididos em ***k*** subconjuntos de mesmo tamanho
    2. Cada iteração toma um subconjunto para teste e os demais para treinamento

  - Validação cruzada de fator ***k*** (***k-fold***)
    - Normalmente os subconjuntos são estratificados antes do processo
    - A estratificação contribui para reduzir a variância da estimativa
    - Ao final calcula-se a média das estimativas de erro para obter o erro geral

  - Exemplo de validação cruzada com ***k*** `= 10`
    - Validação cruzada estratificada será repetida 10 vezes
    - Experimentos têm demonstrado que 10 é uma boa escolha

Aprendizagem supervisionada

- Validação cruzada (cont.)
  - $k = 4$



Aprendizagem supervisionada

- Medidas para avaliação de classificadores
  - São calculadas a partir da taxa de acerto ou de erro
  - Note que nem sempre a taxa de acerto é a medida mais relevante

- Acurácia
  - Número de instâncias que foram corretamente classificadas
    - Avalia erro nas classes igualmente
  - Pode não ser adequada para dados desbalanceados
    - O motivo é que pode prejudicar desempenho para classe minoritária
    - O qual geralmente é mais interessante que a classe majoritária

Aprendizagem supervisionada

- Acurácia (cont.)
  - $Acc(M)$: % de instâncias de teste que são corretamente classificadas
  - $Err(M): 1 - Acc(M)$

  - Quando a classificação é binária existem dois tipos de erro
    - Falso positivo e falso negativo
    - Representadas pela matriz de confusão

Matriz de confusão	Positivo	Negativo
Positivo	Positivos verdadeiros	Falsos negativos
Negativo	Falsos positivos	Negativos verdadeiros

Aprendizagem supervisionada

- Acurácia (cont.)
  - Classes não balanceadas nem sempre são favoráveis para esta medida
  - Exemplo com diagnósticos de câncer com acurácia de 90%
    - Classe 1 = tem câncer (4 pacientes)
    - Classe 2 = não tem câncer (500 pacientes)
  - Foram classificados corretamente 454 pacientes que não tem câncer
    - Mas não acertou nenhum dos que tem câncer
    - Além disso, neste exemplo os falsos negativos possuem um efeito nocivo
    - Portanto, é considerado um classificador ruim mesmo com acurácia alta

Aprendizagem supervisionada

- Demais medidas utilizadas
  - Sensibilidade ou ***recall***: $VP \ / \ (VP + FN)$
    - % de exemplos positivos classificados como positivos
    - Exemplo: pacientes diagnosticados corretamente com câncer
  - Especificidade: $VN \ / \ (VN + FP)$
    - % de exemplos negativos classificados como negativos
    - Exemplo: pacientes diagnosticados corretamente sem câncer
  - Precisão: $VP \ / \ (VP + FP)$
    - % de exemplos classificados como positivos que são realmente positivos
    - Exemplo: pacientes diagnosticados corretamente com câncer dentre todos os que foram classificados como positivos

Aprendizagem supervisionada

- *Overfitting*
  - Situação em que a hipótese extraída a partir dos exemplos é muito específica para o conjunto de treinamento
    - A hipótese apresenta um bom desempenho para o conjunto de treinamento
    - Mas possui um desempenho ruim para os casos fora desse conjunto

- *Underfitting*
  - A hipótese induzida apresenta um desempenho ruim tanto no conjunto de treinamento como de teste, sendo necessário avaliar os motivos
    - Poucos exemplos representativos foram dados ao sistema de aprendizado
    - Os parâmetros dos algoritmos de aprendizagem não estão adequados

Aprendizagem supervisionada

- Consistência e completude
  - Após induzida, a hipótese pode ser avaliada sobre estes dos critérios

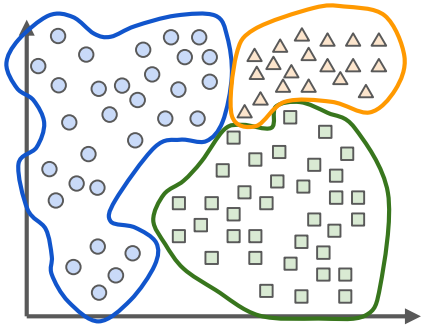
  - Consistência
    - São classificados corretamente os exemplos
    - Vide as medidas já vistas para avaliar a efetividade do classificador

  - Completude
    - São classificados todos os exemplos
    - Ou seja, indica se há hipóteses não cobertas pelo classificador

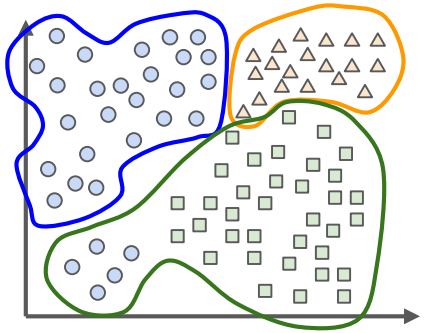
Aprendizagem supervisionada

- Consistência e completude (cont.)

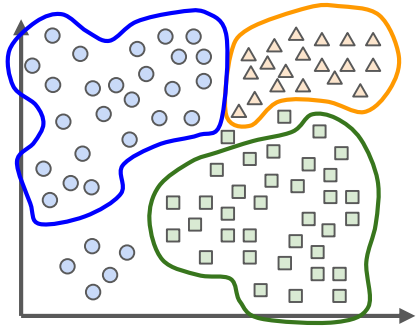
Consistente e completo



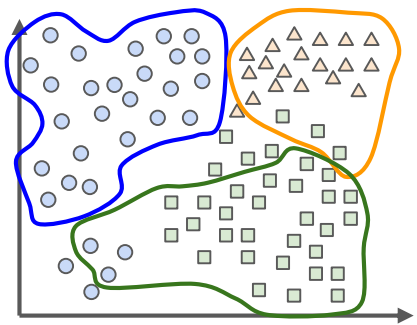
Inconsistente e completo



Consistente e incompleto



Inconsistente e incompleto



Aprendizagem supervisionada

- Categorias de classificadores
  - Também chamados de categorias de sistema de aprendizado
    - Maneira como o conhecimento do modelo é expresso
  - Classificadores não simbólico ou caixa preta
    - Não é facilmente interpretado por humanos
    - Desenvolve sua própria representação de conceitos normalmente numéricas
    - Não fornece uma explicação sobre o processo de classificação
  - Classificadores simbólico ou orientado a conhecimento
    - Cria estruturas simbólicas compreensíveis por seres humanos
    - Geralmente interpretáveis em linguagem natural

Aprendizagem supervisionada

- Algoritmos de aprendizado
  - Também conhecido como indutor ou algoritmo de indução
  - Objetivos do algoritmo de aprendizado
    - Gerar um bom classificador a partir de um conjunto de exemplos rotulados
    - A saída (classificador) é usada para classificar exemplos novos sem rótulos
    - O classificador deve predizer corretamente o rótulo de cada exemplo novo
  - Em seguida o classificador pode ser avaliado considerando
    - Precisão (relação entre acurácia e erro)
    - Capacidade de compreensão
    - Velocidade de aprendizado
    - Outros critérios que determinem o quão apropriado é para a tarefa em questão

Aprendizagem supervisionada

- Paradigmas de aprendizado
  - Abordagem teórica a ser utilizada pelo algoritmo de aprendizado
    - Isto é, descreve a estratégia empregada pelo algoritmo

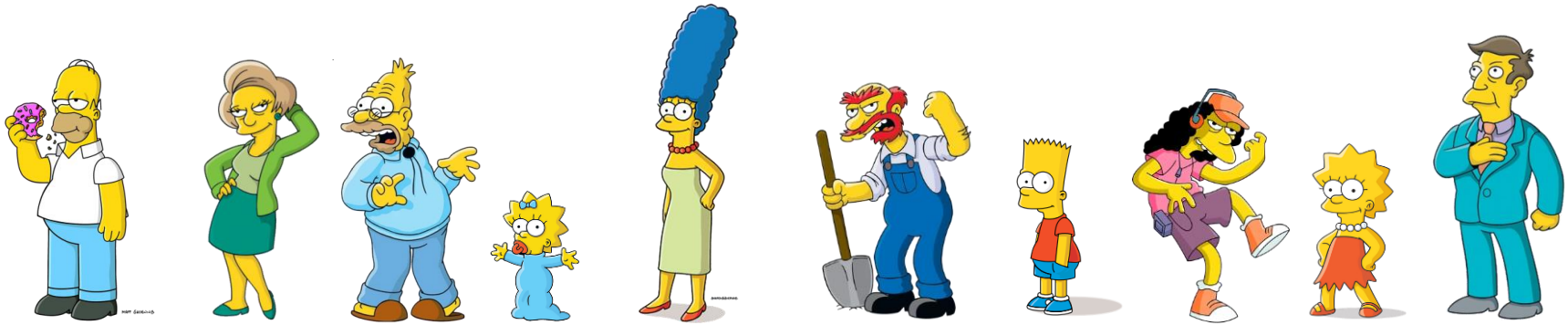
  - Principais paradigmas de aprendizado
    - Simbólico
    - Memorização (***instance-based***)
    - Conexionista
    - Estatístico

Aprendizagem não supervisionada

- Objetivos
  - Analisar instâncias para determinar se existem subconjuntos que podem ser agrupados em certas classes de maneira útil
    - Assume-se que as instâncias não possuem classes relacionadas
  - Exemplos de problemas reais envolvendo agrupamento
    - Perfis de investidores a partir de transações na bolsa de valores
    - Associação entre produtos vendidos a partir dos registros de compras
    - Padrão de atividades de pessoas em redes sociais
    - Agrupar perfis de clientes para melhor resultado em operações de marketing

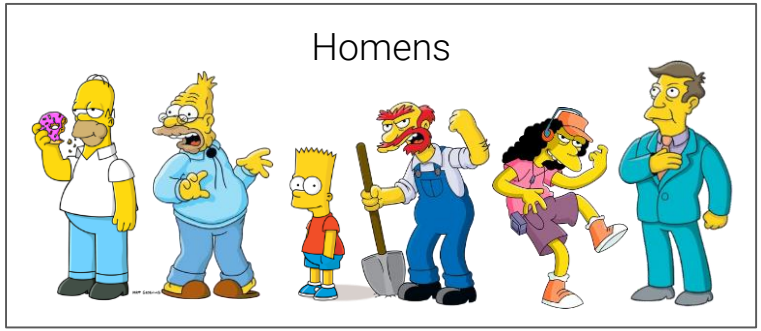
Aprendizagem não supervisionada

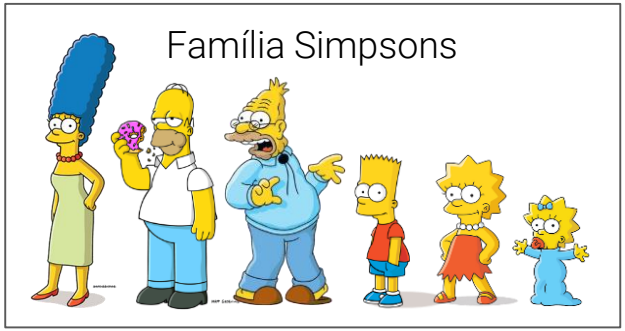
- Objetivos (cont.)
  - Como agrupar estes objetos?

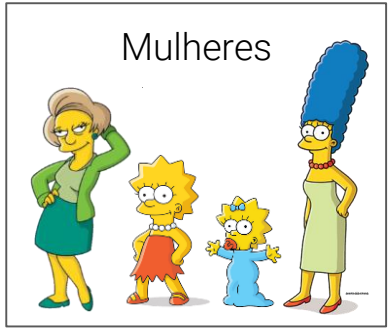


Aprendizagem não supervisionada

- Objetivos (cont.)
  - Como agrupar estes objetos?
  - Agrupamento é algo subjetivo







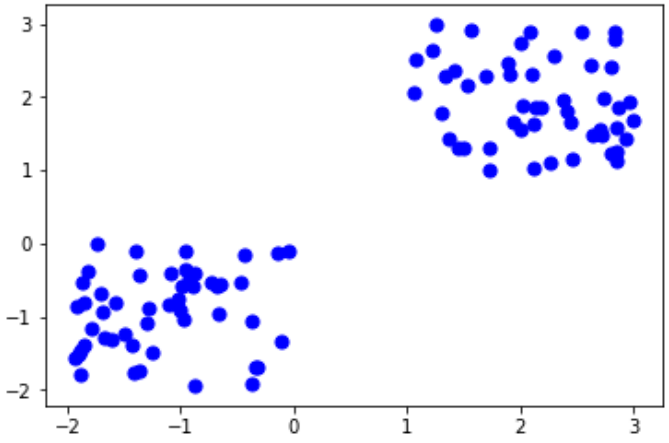
Aprendizagem não supervisionada

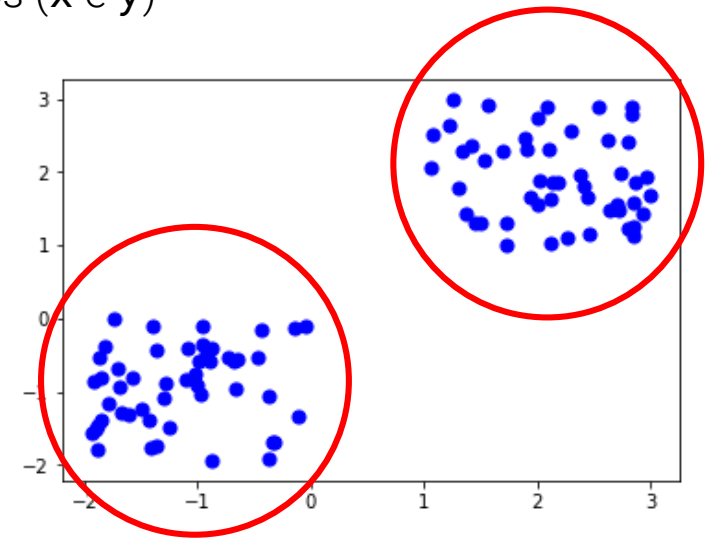
- Agrupamento ou ***clustering***
  - Organização de objetos similares (sob algum aspecto) em grupos
    - O critério de agrupamento é baseado na similaridade em algum nível
    - É medida por uma função de distância ou similaridade

- Definição de ***cluster***
  - Propriedades
    - Coleção de objetos que são similares entre si
    - Diferem-se dos objetos pertencentes a outros ***clusters***
  - Necessita de uma medida de similaridade
    - A mais comum é baseada em distância (***distance-based clustering***)

Aprendizagem não supervisionada

- Definição de ***cluster*** (cont.)
  - Exemplo de ***clusters*** baseados em distância
    - Os planos bidimensionais ilustram um conjunto de instâncias dispersas
    - Cada instância possui dois atributos (**x** e **y**)





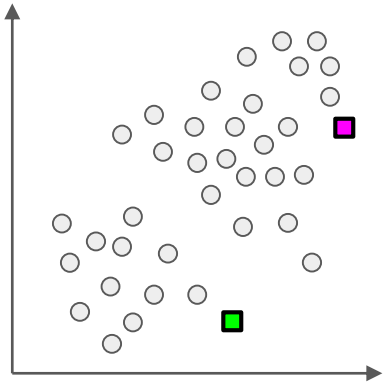
Algoritmo *k-Means*

- Técnica mais simples de aprendizagem não supervisionada
  - Consiste em fixar ***k*** centróides de maneira aleatória
  - Cada centróide está relacionado a um grupo (***cluster***) a ser formado
  - Associa-se cada objeto ao seu centróide mais próximo
  - Depois recalcula-se os centróides com base nos objetos classificados

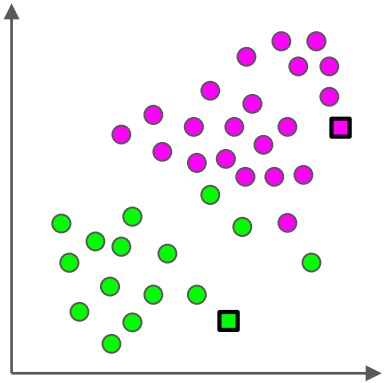
- Passos do algoritmo
  1. Determinar os centróides
  2. Atribuir a cada objeto do grupo o centróide mais próximo
  3. Após atribuir um centróide a cada objeto, os centróides são recalculados
  4. Repetir os passos 2 e 3 até que os centróides não sejam modificados

Algoritmo *k-Means*

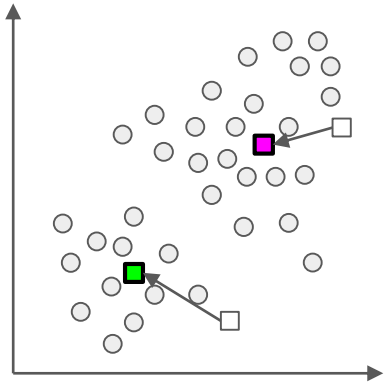
- Passos do algoritmo (cont.)
  - Exemplo



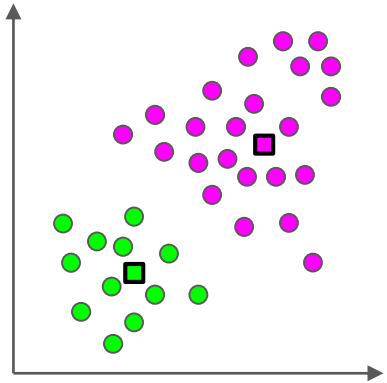
1. Determinar centróides



2. Atribuir objetos



3. Recalcular centróides



4. Repetir passos 2 e 3

Algoritmo *k-Means*

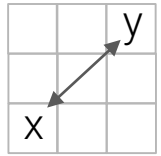
- Importância da inicialização
  - Dependendo onde os centróides iniciam, será necessário mais iterações
  - Quando há noção dos centróides, pode-se melhorar a convergência

- Medidas de distância
  - Distância Euclidiana

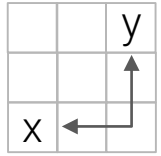
$$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

  - Distância Manhattan

$$d = \sum_{i=1}^{n} |x_i - y_i|$$



Distância Euclidiana



Distância Manhattan

Algoritmo *k-Means*

- Medidas de distância (cont.)
  - Distância Minkowski

$$d = \left(\sum_{i=1}^{n} (x_i - y_i)^r \right)^{1/r}$$

  - Parâmetro r
    - $r = 1$: distância Manhattan
    - $r = 2$: distância Euclidiana

Algoritmo *k-Means*

- Critérios de otimização
  - Qual é o número adequado de objetos em um ***cluster***?
  - Como encontrar ***clusters*** que minimizam/maximizam um dado critério?

  - Definição
    - Dado um conjunto $D = \{x_1, \ldots, x_n\}$ composto de n exemplos
    - Devem ser dividido em c subconjuntos disjuntos $D_1, \ldots, D_c$
    - Cada subconjunto representa um ***cluster***

- Principais critérios de otimização
  - Soma dos erros quadrados
  - Critérios de dispersão

Algoritmo *k-Means*

- Soma dos erros quadrados
  - Critério mais simples e mais usado para otimização em ***clustering***
  - Considera uma noção de tamanho médio dos ***clusters***
    - n_i representa o número de exemplos no cluster D_i
    - m_i representa a média deste ***cluster***

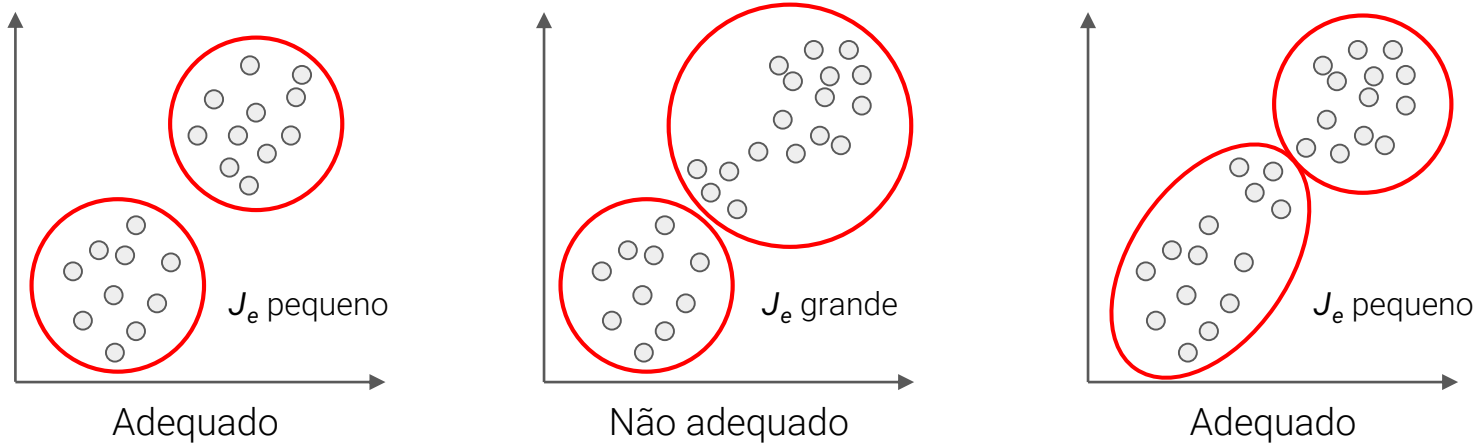
$$m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$

    - J_e representa a soma dos erros quadrados

$$J_e = \sum_{i=1}^{c} \sum_{x \in D_i} \left(x - m_i\right)^2$$

Algoritmo *k-Means*

- Soma dos erros quadrados (cont.)
  - Adequado em casos onde há uma separação natural e homogênea
    - Não adequado em casos onde há dados muito dispersos (***outliers***)
    - Outliers afetam os valores médios dos ***clusters***
  - Exemplo



Algoritmo *k-Means*

- Critérios de dispersão
  - Vetor médio de um cluster **`i`** (m_i)

    $$m_i = \frac{1}{n_i} \sum_{x \in D_i} x$$
  - Vetor médio total (m)

    $$\mathrm{m} = \frac{1}{n} \sum_{D} x$$
  - Dispersão de um cluster **`i`** (S_i)

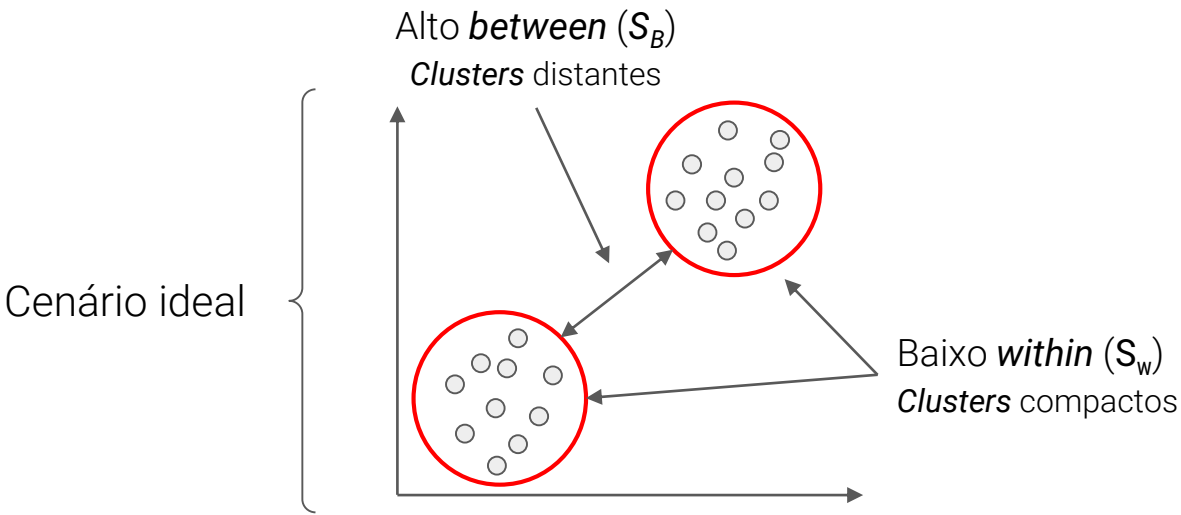
    $$S_i = \sum_{x \in D_i} (x - m_i)(x - m_i)^t$$
  - *Within-cluster* (S_w)

    $$S_w = \sum_{i=1}^{c} S_i$$
  - *Between-cluster* (S_B)

    $$S_B = \sum_{i=1}^{c} n_i (m_i - \mathrm{m})(m_i - \mathrm{m})^t$$

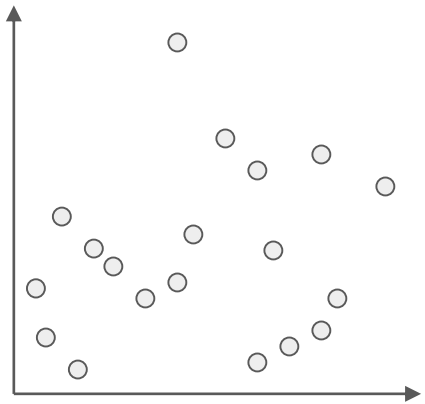
Algoritmo *k-Means*

- Critérios de dispersão (cont.)
  - Relação entre ***within*** e ***between-cluster***
    - ***Within***: leva em consideração a compactação do ***cluster***
    - ***Between***: leva em consideração a distância entre ***clusters***

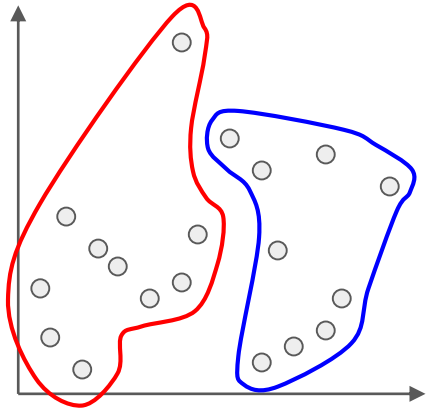


Algoritmo *k-Means*

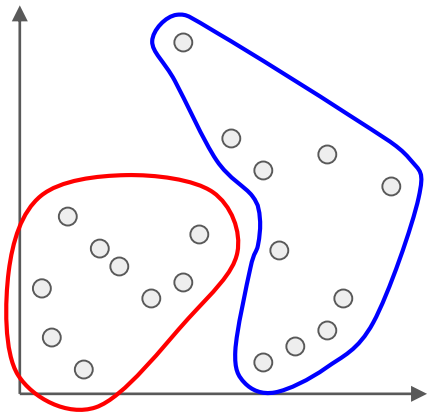
- Exemplos de ***clusters*** com diferentes critérios de otimização
  - Considerando **c = 2**



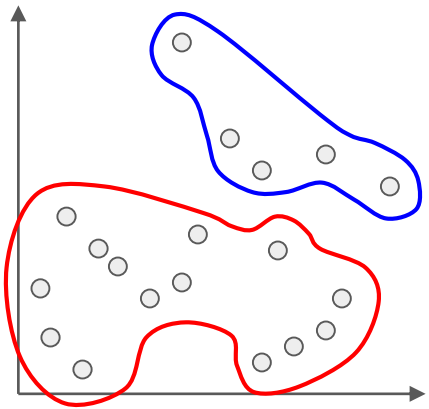
Sem agrupamento



Erro quadrado



Within-cluster



Within e *between-cluster*

Sistemas de aprendizado simbólico

- Características do paradigma
  - Aprender construindo representações simbólicas de um conceito
    - Realizam análise dos exemplos e contraexemplos desse conceito
  - Permitir a compreensão do modelo gerado por humanos

- Representações simbólicas em geral são formas de expressão lógica
  - Árvores de decisão
  - Tabelas de decisão baseada em regras
  - Redes semânticas

- Os algoritmos populares são *ID3*, *C4.5* e *CART*

Sistemas de aprendizado *instance-based*

- Características do paradigma
  - Classificar um exemplo a partir da lembrança de outro similar
    - Onde no exemplo similar é conhecida a classe
    - Assim, assume-se que o novo exemplo terá a mesma classe
  - Sistemas baseados em exemplos que classificam exemplos não vistos
    - Usam exemplos similares conhecidos para isto
    - Esse tipo de sistema de aprendizado é denominado ***lazy***

- Sistemas ***lazy*** não necessitam manter os exemplos na memória
  - Oposto dos sistemas ***eager*** que usam exemplos para induzir um modelo
  - Sistemas ***lazy*** não mantém modelos, são gerados a cada classificação
    - Um algoritmo popular é o KNN (***K Nearest Neighbor***)

Sistemas de aprendizado conexionista

- Características do paradigma
  - Também conhecidas como redes neurais artificiais
    - Construções matemáticas simplificadas
    - Inspiradas no modelo biológico do sistema nervoso
  - A representação envolve inúmeras unidades altamente interconectadas
    - Cada unidade possui uma entrada, uma função de ativação e uma saída
    - Estas unidades são organizadas em camadas
    - Geralmente possui uma camada de entrada, uma escondida e uma de saída
- Os algoritmos populares são ***Multi-layer Perceptron*** e ***Backpropagation***

Sistemas de aprendizado estatístico

- Características do paradigma
  - Usam modelos estatísticos ou probabilísticos
  - Objetiva encontrar uma boa aproximação do conceito induzido
    - Geralmente são métodos são paramétricos
    - Buscam valores apropriados nos parâmetros do modelo a partir de exemplos

  - Nestes sistemas de aprendizado destacam-se os métodos bayesianos
    - Usam modelos probabilísticos baseado no conhecimento prévio do problema
    - Combinam os exemplos de treinamento para determinar a probabilidade final
    - Podem considerar probabilidade causal entre os atributos da base

- Os algoritmos populares são ***Naive Bayes***, ***Redes Bayesianas*** e ***SVM***

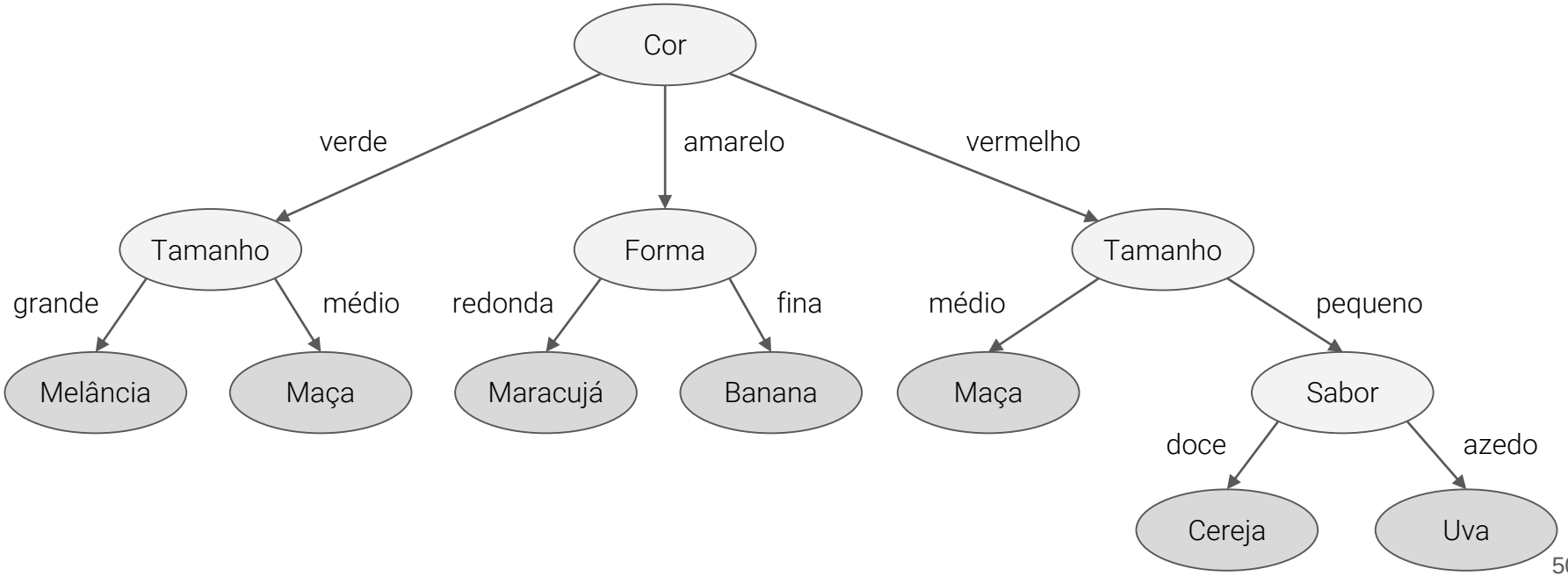
Algoritmo *ID3*

- Aprendizagem em árvores de decisão
  - É um sistema de aprendizado simbólico
  - Método prático de aprendizagem indutiva
    - Aproximação de funções de valor discreto
    - Relativamente robusto a ruídos nos dados

  - A função aprendida é representada por uma árvore de decisão
    - Podem ser representadas com regras **se** <condição> **então** <ação>
    - Classificam instâncias ordenando-as a partir da raiz até alguma folha
    - Cada nó da árvore define o teste de algum atributo da instância
    - Cada ramo saindo de um nó refere-se a um dos valores possíveis do atributo

Algoritmo *ID3*

- Aprendizagem de árvores de decisão (cont.)



Algoritmo *ID3*

- Estratégia do algoritmo
  - Construção de uma árvore de decisão de cima para baixo
    - Inicia com a questão: qual atributo deve ser testado na raiz da árvore?
  - Cada atributo da instância é avaliado usando um teste estatístico
    - Determina o quão bem ele sozinho classifica os exemplos de treinamento
    - Ou seja, o nó raiz representa o melhor atributo para esta tarefa
    - Um descendente do nó raiz é criado para cada valor possível do atributo
  - O processo é repetido usando exemplos de treinamento associados com cada nó descendente para selecionar o melhor atributo para testar naquele ponto da árvore

Algoritmo *ID3*

- Estratégia do algoritmo (cont.)
  - O processo de construção da árvore usa uma busca gulosa (***greedy***)
    - O algoritmo nunca recua para reconsiderar escolhas prévias
  - Como identificar os atributos mais úteis para a classificação?

- Ganho de informação
  - Medida quantitativa que mede o quão bem um atributo separa os exemplos do conjunto de treinamento
    - Sempre considerando a classificação alvo

  - A cada divisão, os subconjuntos posteriores ficam cada vez mais puros
    - Um subconjunto dos dados será mais puro conforme contém menos classes
    - A entropia é uma forma de medir a pureza de cada subconjunto

Algoritmo *ID3*

- Ganho de informação dos atributos (cont.)
  - Calculo da entropia

$$entropia(X) = -\sum_{i} p_i \log_2 p_i$$

  - Onde
    - X: um dado atributo do conjunto
    - i: cada uma das classes do conjunto
    - p_i: probabilidade do atributo i da classe acontecer para o atributo

Algoritmo *ID3*

- Ganho de informação dos atributos (cont.)
  - Exemplo de um conjunto ***S*** com 14 exemplos, 9 positivos e 5 negativos

$$entropia(X) = -\sum_i p_i \log_2 p_i$$

  - Entropia da classe (***PlayTennis***)

p_{yes} = 9/14 = 0,64
p_{no} = 5/14 = 0,35
$p_{yes} * log_2(p_{yes})$ = 0,41
$p_{no} * log_2(p_{no})$ = 0,53

entropia(S) = 0,94

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Algoritmo *ID3*

- Ganho de informação dos atributos (cont.)
  - Cálculo do ganho de informação

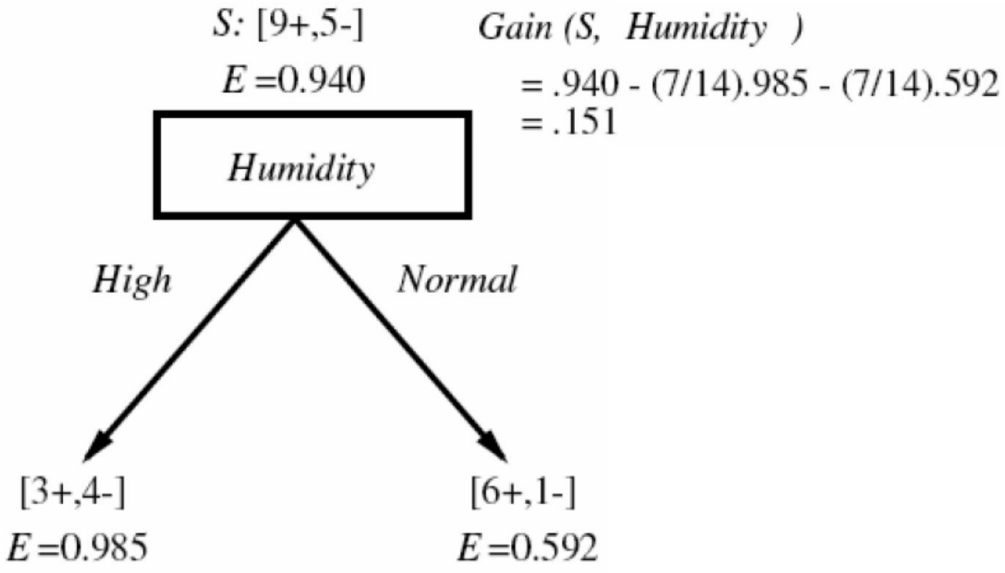
    $$ganho(X) = entropia(S) - entropia(X)$$

  - Onde
    - $entropia(S)$: entropia do conjunto
    - $entropia(X)$: entropia do atributo, sendo representado pela soma ponderada das entropias de suas partições

Algoritmo *ID3*

- Ganho de informação dos atributos (cont.)
  - Exemplo de um conjunto ***S*** com 14 exemplos, 9 positivos e 5 negativos

ganho(Humidity) = entropia(S) - entropia(Humidity)

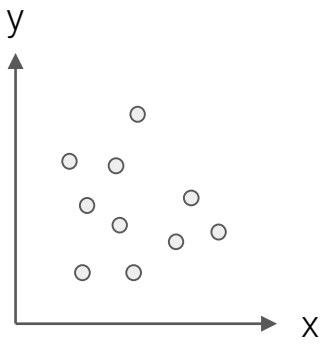


Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

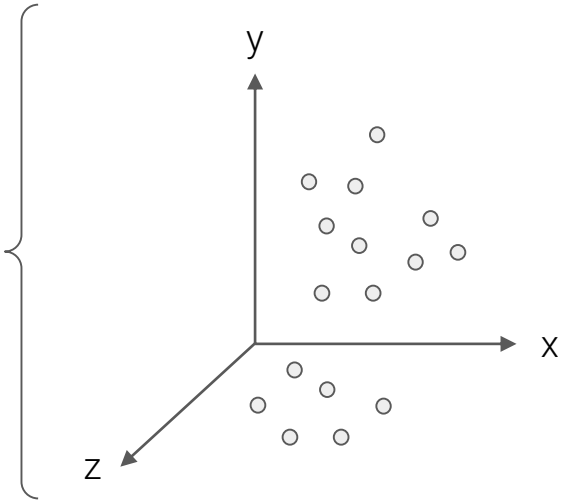
Algoritmo *k-NN*

- Aprendizagem baseada em instâncias
  - As instâncias são representadas como pontos em um espaço Euclidiano
  - Vetores de características ou atributos
    - Cada atributo do conjunto refere-se a uma dimensão do espaço

2-dimensional



3-dimensional



Algoritmo *k-NN*

- Processo de aprendizagem
  - A aprendizagem consiste em armazenar os exemplos de treinamento
    - $<x_1, c_1>, <x_2, c_2>, \ldots, <x_n, c_n>$
    - x_i é um vetor de atributos
    - c_i é a classe para o vetor x_i
  - Para encontrar o valor da classe associado à instância de testes $<x_t, ?>$
    - São buscadas na memória um conjunto de instâncias similares
    - Estas instâncias similares serão utilizadas para classificar a nova instância
  - Qual a distância $d_1, d_2, \ldots, d_n$ entre $x_1, x_2, \ldots, x_n$ e x_t?
    - Quais são as instâncias de exemplo mais similares a x_t?
    - A classe de x_t será a classe associada ao conjuno de treinamento mais similar

Algoritmo *k-NN*

- Estratégia do algoritmo
  - O algoritmo ***k–NN*** (***k Nearest Neighbor***) assume que
    - Todas as instâncias correspondem a pontos em um espaço n–dimensional
    - Vizinhos mais próximos são definidos em termos da distância Euclidiana
  - Estratégia de classificação
    - Atribuir a classe mais frequente dentre as ***k*** amostras mais próximas
    - Utiliza um esquema de votação para esta tarefa
  - É o método de aprendizagem baseado em instâncias mais elementar
    - Constrói aproximações para a função alvo para cada instância de teste

Algoritmo *k-NN*

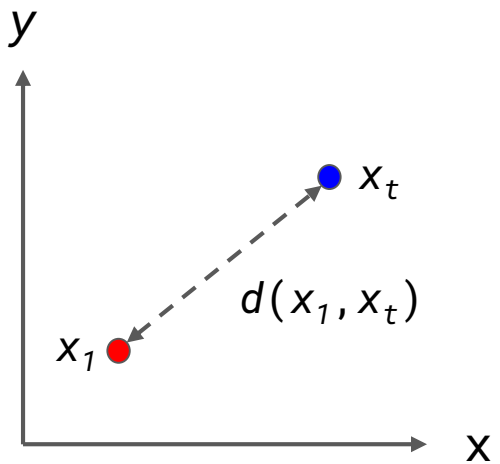
- Estratégia do algoritmo
  - Distância Euclidiana

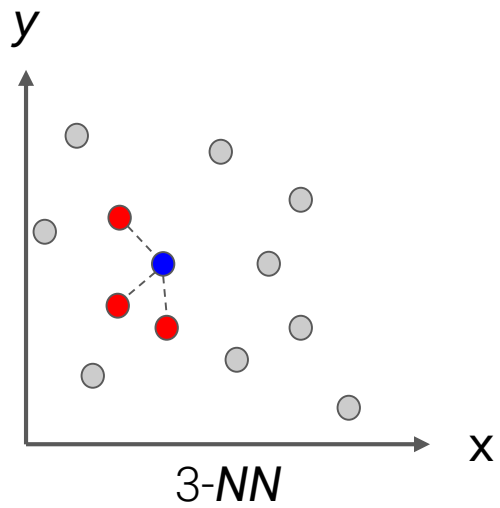
$$d(x, y) = \sqrt{\sum_{i=i}^{n} (x_i - y_i)^2}$$

  - Onde
    - x é o vetor de atributos de uma instância do conjunto de treinamento
    - y é o vetor de atributos a instância a ser classificada
    - x_i é um valor de um atributo de x
    - y_i é um valor de um atributo de y

Algoritmo *k-NN*

- Estratégia do algoritmo
  - Exemplo com $k = 3$ (3-*NN*)



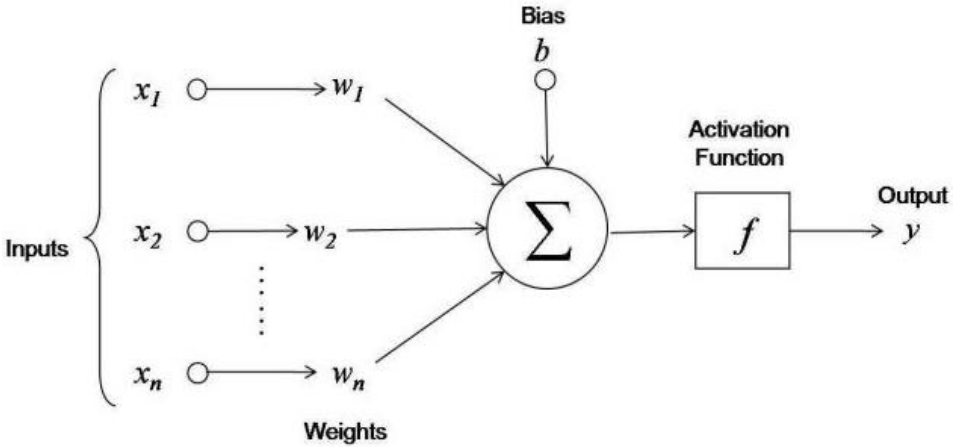


Algoritmo *Multi-Layer Perceptron*

- Aprendizagem baseada em redes neurais artificiais
  - Fornecem um método prático para a aprendizagem de funções de valor real e de valor discreto a partir de exemplos
    - É robusta a erros e ruídos nos dados de treinamento
    - Embora os algoritmos de treinamento podem consumir muito tempo
  - O valor do conceito alvo (classe) pode ser
    - Valores discretos
    - Valores reais
    - Vetores de valores discretos e reais
  - A avaliação da função alvo aprendida é rápida
    - Uma vez que a rede esteja treinada, a avaliação de novas instâncias é rápida

Algoritmo *Multi-Layer Perceptron*

- *Perceptron*
  - Rede neural elementar baseada em uma unidade chamada *Perceptron*
  - Funcionamento de um *Perceptron*
    1. Recebe um vetor de entradas de valor real
    2. Calcula uma combinação linear destas entradas
    3. Fornece uma saída (1 ou -1)



Activation function

$$\begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$$

Algoritmo *Multi-Layer Perceptron*

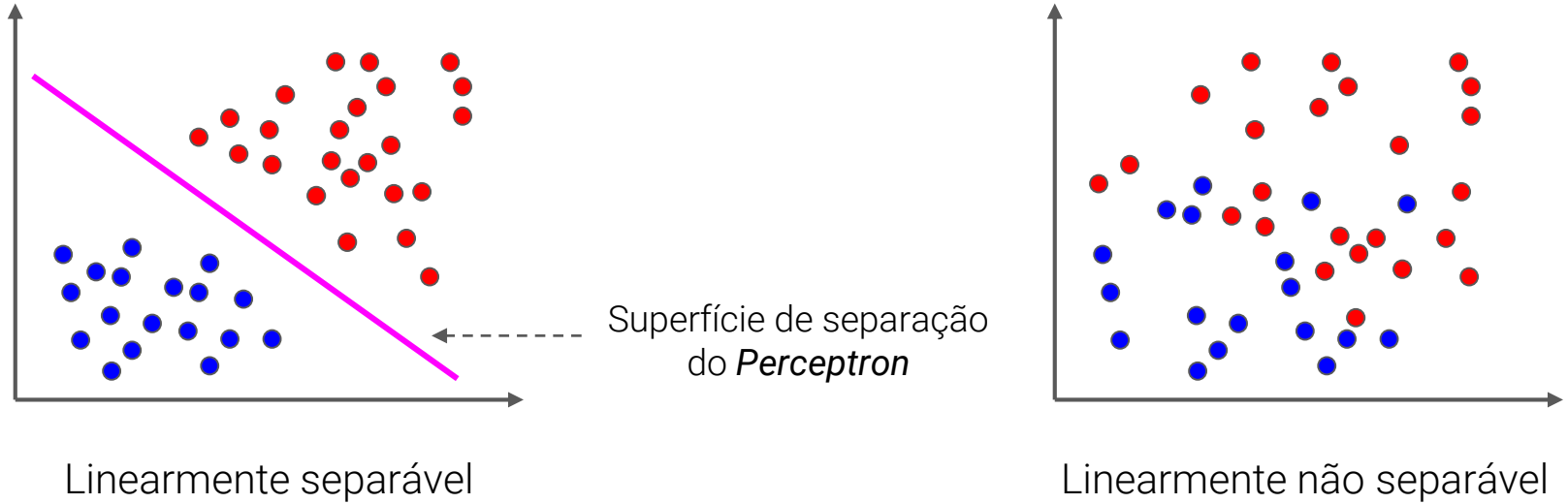
- *Perceptron* (cont.)
  - Cada elemento w_i é uma constante de valor real, ou peso
    - Determina a contribuição da entrada x_i na saída do *Perceptron*
    - A aprendizagem do perceptron envolve a escolha dos valores dos pesos a w_n
  - A camada de entrada possui uma unidade especial chamada ***bias***
    - Termo constante que não depende de nenhum valor de entrada
    - O ***bias*** altera a posição da superfície de separação

Algoritmo *Multi-Layer Perceptron*

- Superfície de separação
  - Um ***Perceptron*** atua como uma superfície de separação
    - Considerando um espaço n-dimensional de instâncias
  - A saída do ***Perceptron*** representa
    - 1 para instâncias dispostas em um lado do hiperplano
    - −1 para instâncias dispostas no outro lado
  - Um único ***Perceptron*** consegue separar somente conjuntos de exemplo que são linearmente separáveis

Algoritmo *Multi-Layer Perceptron*

- Superfície de separação (cont.)



Linearmente separável

Linearmente não separável

Algoritmo *Multi-Layer Perceptron*

- Treinamento de um *Perceptron*
  - Determinar um vetor de pesos de modo que
    - Permita o *Perceptron* produzir a saída correta (**–1** ou **+1**)
    - Considere cada um dos exemplos de treinamento
  - Uma alternativa iterativa
    - Começar com um vetor de pesos aleatórios
    - Aplicar iterativamente a regra perceptron para cada exemplo de treinamento
    - Modificar os pesos cada vez que ele classificar um exemplo erroneamente
  - O processo é repetido várias vezes até que o perceptron classifique todos os exemplos de treinamento corretamente

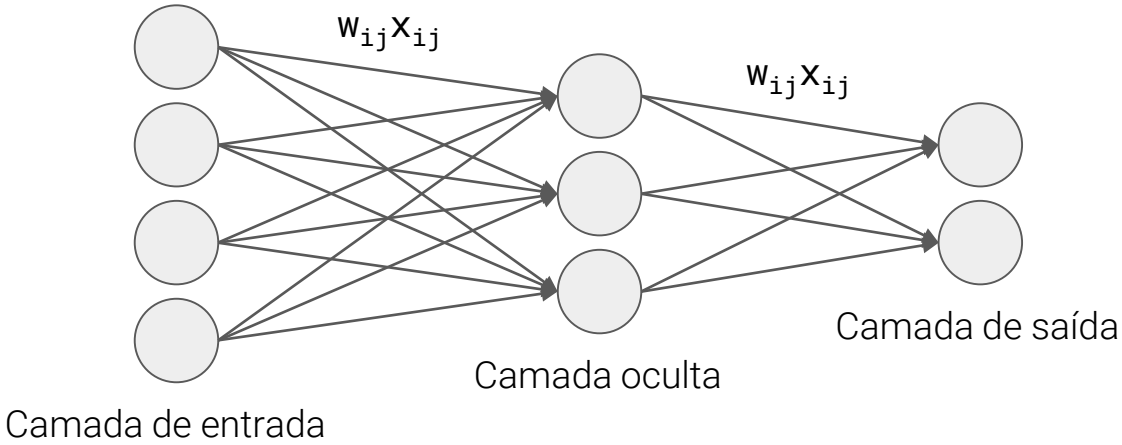
Algoritmo *Multi-Layer Perceptron*

- Treinamento de um ***Perceptron*** (cont.)
  - Os pesos do ***Perceptron*** são modificados a cada iteração
    - Considera uma constante (parâmetro) para ajuste dos pesos
    - Esta constante é um parâmetro chamado taxa de aprendizagem

- *Multi-Layer Perceptron*
  - ***Perceptrons*** expressam somente superfícies de decisão linear
    - Redes multicamadas podem representar superfícies de decisão não lineares
  - Unidade cuja saída seja uma função não-linear de suas entradas
    - Unidade sigmoidal permitem valores entre 0 e 1

Algoritmo *Multi-Layer Perceptron*

- *Multi-Layer Perceptron* (cont.)
  - Requer o treinamento dos pesos a partir de um algoritmo especializado
  - Algoritmo ***Backpropagation***
    - Emprega uma técnica chamada “descida do gradiente”
    - Minimiza o erro entre os valores dos pesos e o esperado para o conceito



Algoritmo *Naive-Bayes*

- Aprendizagem baseada no teorema de Bayes
  - É uma abordagem probabilística para aprendizagem que supõe que
    - Quantidades de interesse são reguladas por distribuições de probabilidade
  - Distribuição de probabilidade
    - Descreve a probabilidade de uma variável aleatória assumir certos valores
  - Conhecimento ***a priori*** pode ser combinado com os dados observados para determinar a probabilidade de uma hipótese
    - Conhecimento ***a priori*** é um conhecimento ou justificação independente
    - Conhecimento ***a posteriori*** é dependente, é uma probabilidade condicionada

Algoritmo *Naive-Bayes*

- Aprendizagem baseada no teorema de Bayes (cont.)
  - $P(c|X) = P(X|c) * P(c) / P(X)$

  - Onde
    - $P(c|X)$: probabilidade da classe c dado o vetor X
    - $P(X|c)$: probabilidade do vetor X dada a classe c
    - $P(c)$: probabilidade a priori da classe c
    - $P(X)$: probabilidade a priori do vetor de treinamento X

  - $P(c|X)$ é a probabilidade a *posteriori* de c
    - Reflete a confiança que c se mantenha após observado o vetor X
    - Em outras palavras, reflete a influência de treinamento X
    - Em contraste, a probabilidade a *priori* $P(c)$ é independente de X

Algoritmo *Naive-Bayes*

- Estratégia do algoritmo
  - Adota uma suposição simples de que os valores dos atributos são condicionalmente independentes dado o valor alvo (classe)
    - A probabilidade de observar a conjunção de atributos $x_1, x_2,\ldots,x_n$ é somente o produto das probabilidades para os atributos individuais
    - $P(x_1, x_2, \ldots, x_n | c_j) = P(x_1 | c_j) * P(x_2 | c_j) * \ldots * P(x_n | c_j)$
  - Escolhe-se o valor alvo que maximiza esta probabilidade ingênua
    - Os termos $P(c_j)$ e $P(x_i | c_j)$ são estimados baseado nas suas respectivas frequências no conjunto de treinamento

Algoritmo *Naive-Bayes*

- Estratégia do algoritmo (cont.)
  - Exemplo de um conjunto *S*
    - 14 exemplos
    - 9 positivos
    - 5 negativos

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

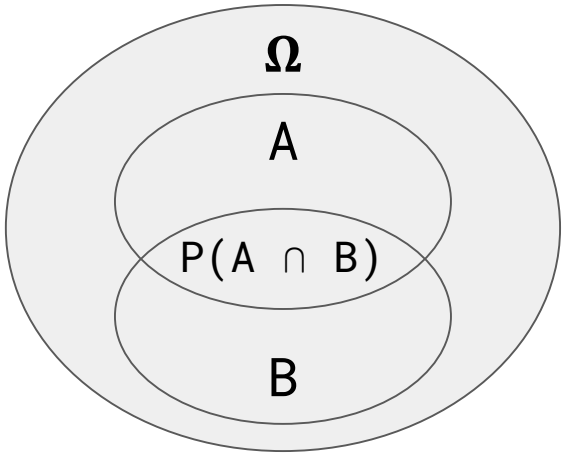
  - Probabilidades a priori
    - $P(PlayTennis=yes) = 9/14 = 0.64$
    - $P(PlayTennis=no) = 5/14 = 0.36$
  - Probabilidades condicionais
    - $P(Wind=strong|PlayTennis=yes) = 3/9 = 0.33$
    - $P(Wind=strong|PlayTennis=no) = 3/5 = 0.60$

Algoritmo *Redes Bayesianas*

- Aprendizagem baseada em *Redes Bayesianas*
  - Redes Bayesianas são aplicadas em cenários de incerteza quando
    - Sabem-se certas probabilidades (condicionais)
    - Buscam-se probabilidades desconhecidas para situações específicas
  - Variáveis aleatórias discretas
    - Considere A como uma variável aleatória discreta
    - Esta variável indica um evento, o qual não se tem certeza se acontece ou não
    - A probabilidade de A é igual à proporção de resultados em que é A verdadeiro
  - Intuição sobre probabilidade
    - Ω é o conjunto de todos os resultados possíveis e sua área é $\Omega = 1$
    - A probabilidade da variável aleatória $P(A)$ deve ser $0 \leqslant P(A) \leqslant P(\Omega)$

Algoritmo *Redes Bayesianas*

- Aprendizagem baseada em ***Redes Bayesianas*** (cont.)
  - Probabilidade condicional
    - $P(A \cap B)$ é a proporção do espaço em que **A** é verdadeiro e **B** também
  - Definição formal
    - $P(B|A) = P(A \cap B)/P(A)$



Algoritmo *Redes Bayesianas*

- Aprendizagem baseada em ***Redes Bayesianas*** (cont.)
  - Eventos independentes
    - A e B são independentes se e somente se $P(A \cap B) = P(A)P(B)$
    - A independência de A e B implica em:
    - $P(A|B) = P(A)$, se $P(B) \neq 0$
    - $P(B|A) = P(B)$, se $P(A) \neq 0$

  - Independência condicional
    - A e B são condicionalmente independentes dado C se e somente se $P(A \cap B|C) = P(A|C)P(B|C) \equiv I(A,B|C)$
    - $P(A|B,C) = P(A|C)$, se $P(B|C) \neq 0$
    - $P(B|A,C) = P(B|C)$, se $P(A|C) \neq 0$

Algoritmo *Redes Bayesianas*

- Aprendizagem baseada em ***Redes Bayesianas*** (cont.)
  - A tarefa de classificação consiste em determinar $P(H|A)$
    - Probabilidade a ***posteriori***
    - Probabilidade da hipótese ser verdadeira para os dados observados em A
  - Porém, a classificação parte da probabilidade inicial $P(H)$
    - Probabilidade a ***priori***
- Definição de uma ***Rede Bayesiana***
  - Utiliza a dependência condicional para tratar atributos correlacionadas
  - Em outras palavras, levam em conta a dependência entre atributos
    - Probabilidade condicional

Algoritmo *Redes Bayesianas*

- Definição de uma ***Rede Bayesiana*** (cont.)
  - A probabilidade condicional de atributos pode ser mapeada em dígrafos
    - Mais especificamente por uma DAG (***Directed Acyclic Graph***)

- Elementos de uma ***Rede Bayesiana***
  - DAG representando relações de dependência entre conjunto de variáveis
    - Cada nó do grafo representa uma variável (atributo do conjunto)
    - Cada aresta estabelece uma relação de dependência entre pares de variáveis

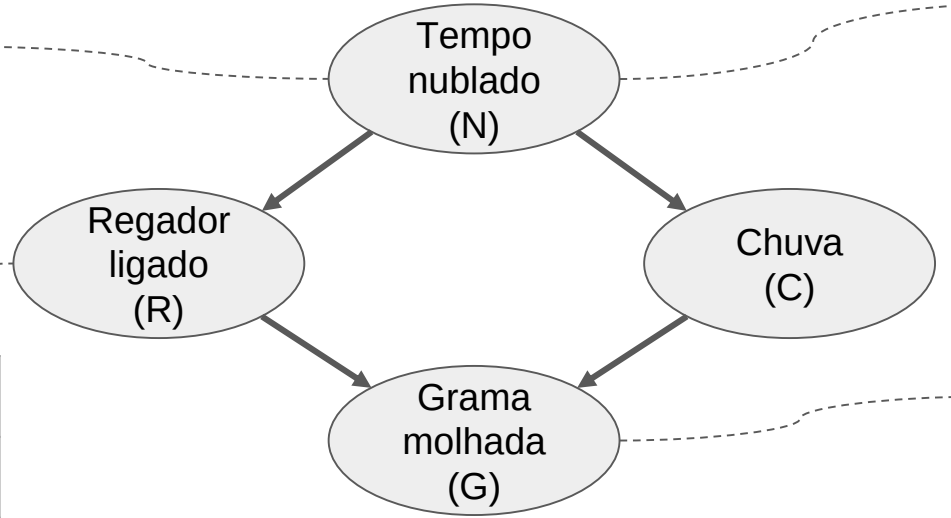
  - Tabela de probabilidades ligando cada nó com seus nós pais imediatos
    - Portanto, a topologia da rede combinada com a tabela de probabilidade codifica a relação de dependência condicional

Algoritmo *Redes Bayesianas*

- Elementos de uma ***Rede Bayesiana*** (cont.)
  - Exemplo de uma ***Rede Bayesiana***

P(N=T)	P(N=F)
0.5	0.5

N	P(R=T)	P(R=F)
T	0.8	0.2
F	0.2	0.8



N	P(R=T)	P(R=F)
T	0.1	0.9
F	0.5	0.5

N	C	P(G=T)	P(G=F)
T	T	0.99	0.01
T	F	0.9	0.1
F	T	0.9	0.1
F	F	0	1

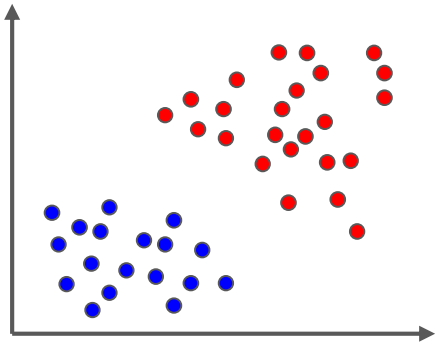
Algoritmo *Redes Bayesianas*

- Estratégia do algoritmo
  - Inicia a inferência a partir das variáveis independentes
  - Calcula a probabilidade das demais variáveis dependentes
    - Este cálculo é realizado percorrendo sucessivamente os nós da DAG
  - Ao aplicar este processamento para o conjunto de testes tem-se
    - A probabilidade de um dado valor alvo (classe) a partir de um exemplo
    - Um modelo menos suscetível ao ***overfitting*** dada a natureza probabilística

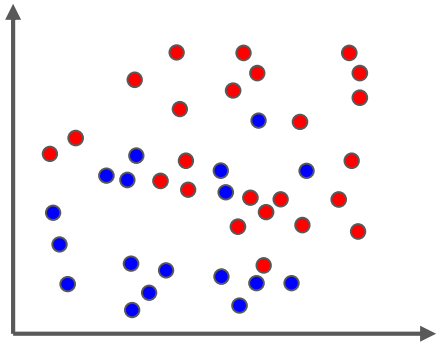
- Relação entre ***Naive Bayes*** e ***Redes Bayesianas***
  - Mais robusto para lidar com problemas envolvendo probabilidade causal
  - Construir uma rede demanda um esforço razoável
    - Porém, acrescentar novas variáveis é praticamente imediato

Algoritmo *SVM*

- Aprendizagem baseada em probabilidade
  - Cada instância do conjunto pode ser visto como um ponto em um plano
    - O aprendizado consiste em dividir as instâncias neste espaço euclidiano
  - Porém, e se o espaço não for linearmente separável?
    - Isto eleva significativamente o desafio do aprendizado



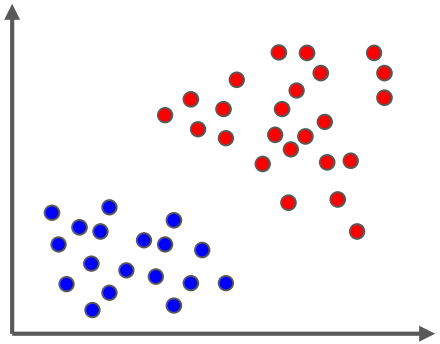
Linearmente separável



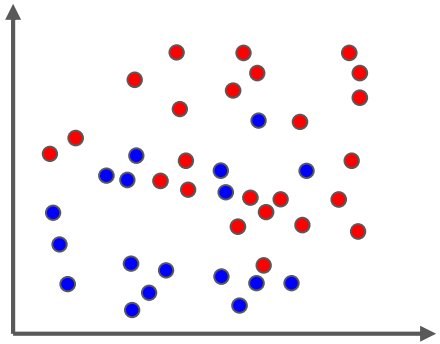
Linearmente não separável

Algoritmo *SVM*

- Aprendizagem baseada em probabilidade (cont.)
  - Suponha que o problema abaixo tem por objetivo classificar cogumelos entre comestíveis e venenosos
    - Cada ponto do plano representa uma amostra de cogumelo
    - O eixo **X** representa a largura e o eixo **Y** a altura do cogumelo
    - As amostras azuis são comestíveis e as vermelhas são venenosos



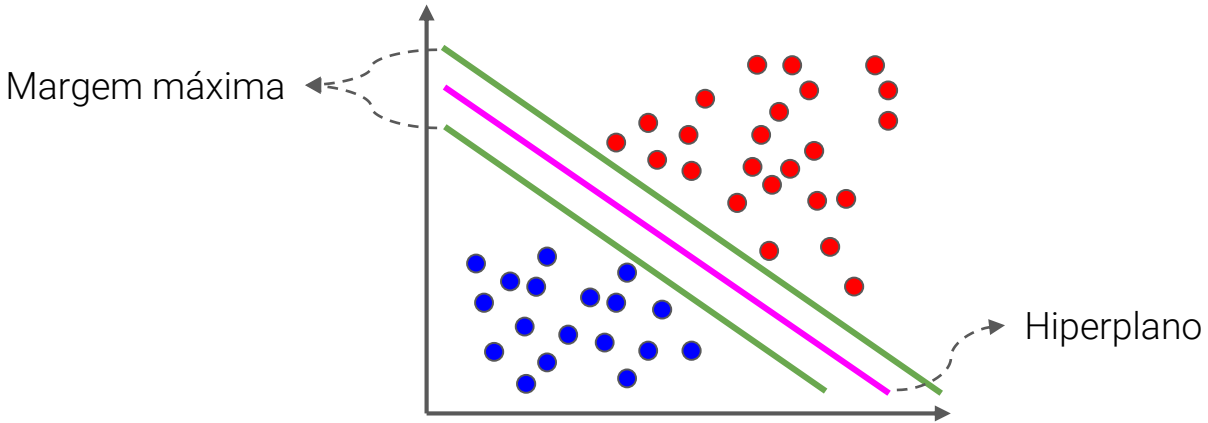
Linearmente separável



Linearmente não separável

Algoritmo *SVM*

- Estratégia do algoritmo
  - Características do *SVM* (*Support Vector Machine*)
    - Realiza uma busca por um hiperplano que divida duas classes com maior margem possível
    - Capaz de realizar a separação do plano em problemas linearmente separáveis ou não linearmente separáveis

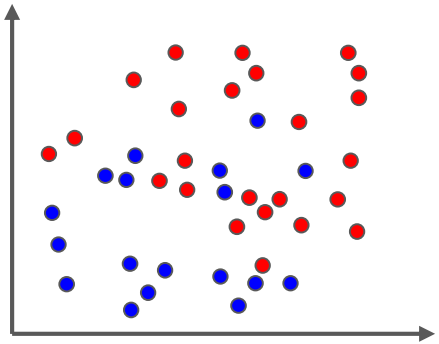


Algoritmo *SVM*

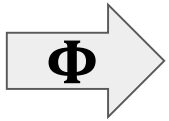
- Estratégia do algoritmo (cont.)
  - Espaços linearmente separáveis
    - Resolvidos com variáveis de relaxamento
    - Estas variáveis suavizam as restrições impostas pelo hiperplano
  - Espaços linearmente não separáveis
    - Utilizadas funções de mapeamento $\mathbf{\Phi}$
    - Estas funções geram um novo espaço de atributos
    - O novo espaço de atributos é chamado de espaço de características
  - Em casos não linear, a dimensão do novo espaço de características deve ser maior que o original
    - Isto aumenta a probabilidade das amostras serem linearmente separáveis

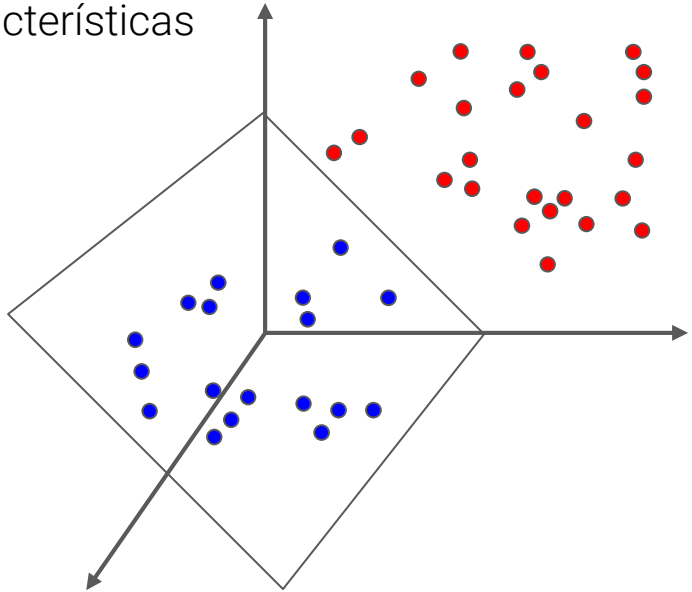
Algoritmo *SVM*

- Estratégia do algoritmo (cont.)
  - Transformação do espaço de estados
    - Adição de um novo espaço de características



Linearmente não separável





Linearmente separável

Algoritmo *SVM*

- Funções de ***kernel***
  - Diferentes funções podem ser utilizadas para a geração do novo espaço de características
    - A escolha estão relacionadas às características do domínio do problema
  - Os ***kernels*** mais populares são
    - Polinomial
    - Gaussiano
    - Sigmoidal
  - Alguns ***kernels*** suportam diferentes tipos de problemas como
    - Regressão
    - Multiclasse
    - Uma classe

Algoritmo *SVM*

- Regressão
  - Lida com problemas envolvendo a predição de valores contínuos
  - Ao invés de classes discretas

- Multiclasse
  - Lida com problemas envolvendo mais de duas classes
  - Um contra todos
    - São gerados K SVMs, onde K é o número de classes
    - Em cada SVM, uma classe é considerada positiva e as outras negativas
  - Todos contra todos
    - São gerados $K\ (K-1)\ /\ 2$ SVMs, onde K é o número de classes
    - Cada SVM compara um par de classes
    - Saída é esperada é quem recebeu mais indicações

Algoritmo *SVM*

- Uma classe (*One-class*)
  - Lida com problemas envolvendo uma única classe
    - Podem ser utilizados para remover outliers do conjunto de dados
    - Exemplos: detecção placas de veículos e identificação de manuscritos
  - Consiste encontrar uma área no plano que
    - Concentre grande parte dos exemplos do conjunto de treinamento
    - Considere somente exemplos positivos

Exercícios

1. Calcule o ganho de informação para cada atributo do conjunto de dados abaixo. Em seguida, determine qual atributo deve ser mantido na raiz da árvore de decisão.

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

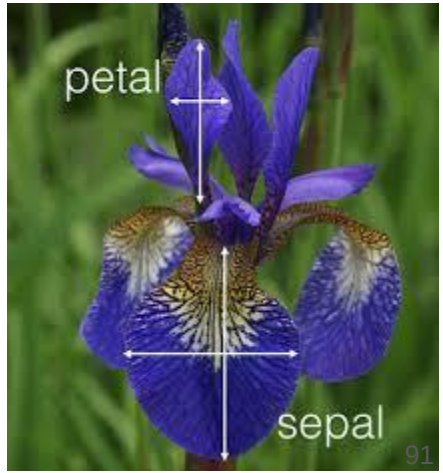
Exercícios

2. Calcule a probabilidade condicional cada atributo do conjunto de dados abaixo. Em seguida, determine a probabilidade para jogar tênis em um dia com:
   - *Outlook*: *Rain*
   - *Temperature*: *Mild*
   - *Humidity*: *Normal*
   - *Wind*: *Weak*

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Exercícios

3. Utilizando a ferramenta Weka, compare o desempenho dos algoritmos ***K-NN***, ***C4.5***, ***Multi-Layer Perceptron***, ***Naive Bayes*** e ***SVM*** utilizando as informações abaixo:
   - Base de dados: https://archive.ics.uci.edu/ml/datasets/Iris
   - Objetivo da base
     - A partir das características da flor Iris, o problema consiste é classificar qual é o tipo dessa flor dentre os três possíveis: versicolor, setosa e virginica
   - Observação
     - A base possui 3 clases
     - Uma classe é linearmente separável das outras duas
     - Uma classe não é linearmente separável das demais
   - Discussão
     - Identificar qual indutor obteve a melhor acurácia
     - Identificar quais indutores geraram modelos orientados ao conhecimento
     - Descrever o procedimento de treinamento dos classificadores e parâmetros utilizados nos indutores



Exercícios

4. O termo dimensionalidade refere-se ao número de características (atributos) de um conjunto de dados, ou seja, a dimensão do espaço de características. A redução da dimensionalidade tem por objetivo eliminar características do conjunto de dados sem causar perdas significativas na eficácia da tarefa de aprendizado. As duas principais razões para que a dimensionalidade seja a menor possível está ligada ao custo de medição e a precisão do classificador. A partir da base abaixo, utilize um algoritmo genético para otimizar a tarefa de classificação pelo algoritmo ***K-NN***. O objetivo do algoritmo genético é otimizar o número de atributos da base de dados, de modo a maximizar a acurácia do algoritmo ***K-NN***.
    - Base de dados: https://archive.ics.uci.edu/ml/datasets/Ionosphere
    - Objetivo da base
        - Contém registros de leituras de dados da ionosfera, compostas pela incidência de íons, plasma ionosférico, níveis de radiação, dentre outras
        - O propósito consiste em identificar níveis de elétrons e classificá-los entre bons (***good***) e ruins (***bad***)
    - Use o algoritmo ***K−NN*** com o parâmetro $K = 1$

unisociesc
centro universitário

Aprendizagem de máquina

Inteligência artificial
Prof. Claudinei Dias
Prof. Maicol

Colaboração: Allan Rodrigo Leite