CLASSIFICADORES BASEADOS EM DISTÂNCIA

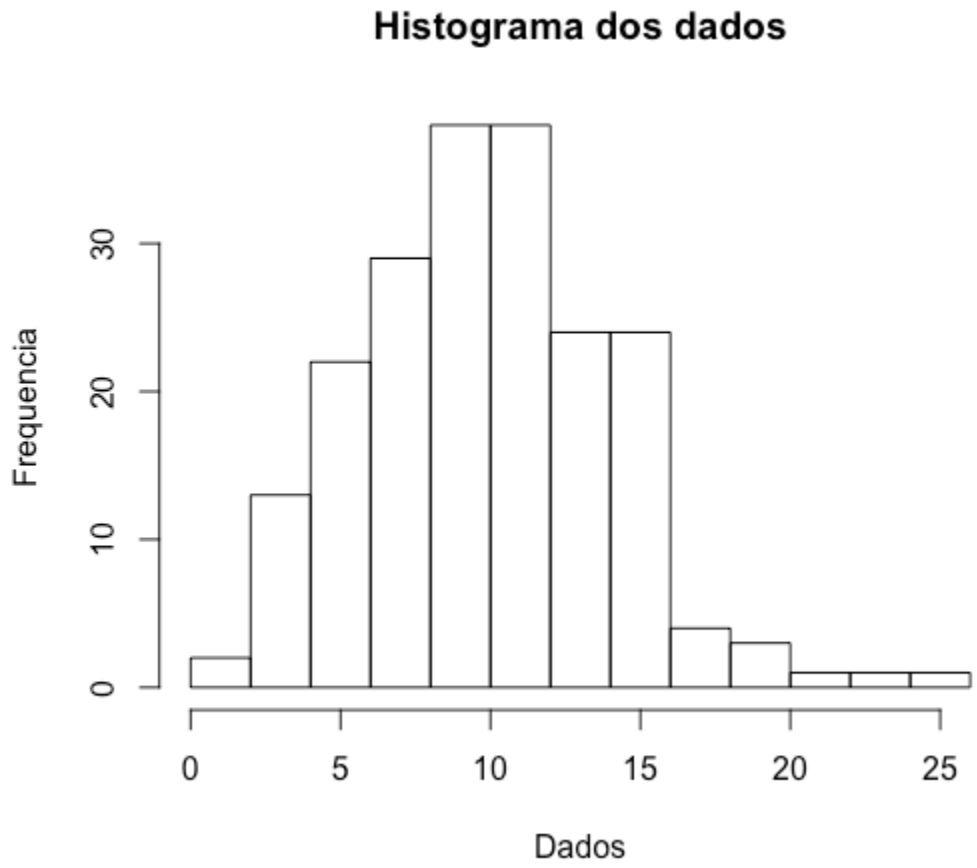
Prof. André Backes | @progdescomplicada

Estimando a densidade

- A função densidade de probabilidade é um conceito fundamental em estatística
  - Permite associar probabilidades a uma variável aleatória *x*
- Especificar a função densidade de uma população nos fornece uma descrição natural da sua distribuição

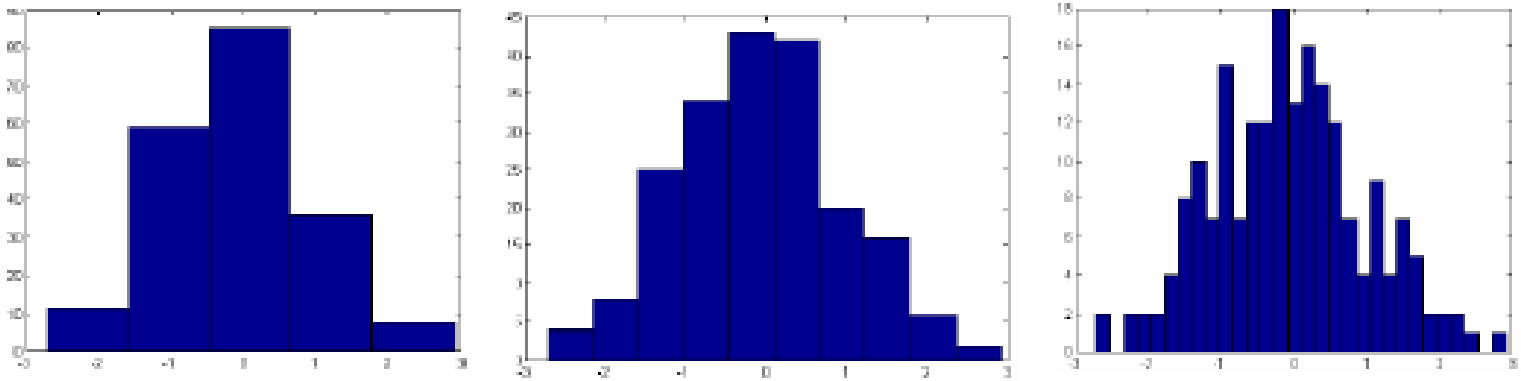
Estimando a densidade

- Histogramas
  - Método mais antigo e simples para estimar a densidade
    - Depende da origem e da largura (*h*) usada para os intervalos
    - *h* controla a granularidade



Estimando a densidade

- Se h é largo
  - Maior confiança na probabilidade no intervalo
    - baseada em um número maior de amostras
  - Detalhes da distribuição podem ser perdidos
- Se h é estreito
  - Preserva-se a estrutura fina (detalhes) da distribuição
  - Menor confiabilidade (menos amostras por intervalo)



Estimando a densidade

- Histogramas
  - Raramente usados em espaços com mais de uma dimensão
    - 1 dimensão: N intervalos
    - 2 dimensões: N^2 intervalos
    - p dimensões: N^p intervalos
  - Histogramas dão uma ideia de estimativa da densidade
    - Necessidade de definir uma abordagem mais formal

Estimando a densidade

- Formalizando a estimativa da densidade
  - A probabilidade de uma amostra *x* estar dentro de uma região *R*, utilizando uma função densidade *p(x)* é

$$P(x) = \int_R p(x)dx$$

Estimando a densidade

- Formalizando a estimativa da densidade
  - Considere a região *R* continua e pequena a tal ponto onde *p(x)* não varia. Então podemos reescrever a formula

$$P(x) = \int_R p(x)dx = p(x) * V$$

  - sendo *V* o volume de *R*

Estimando a densidade

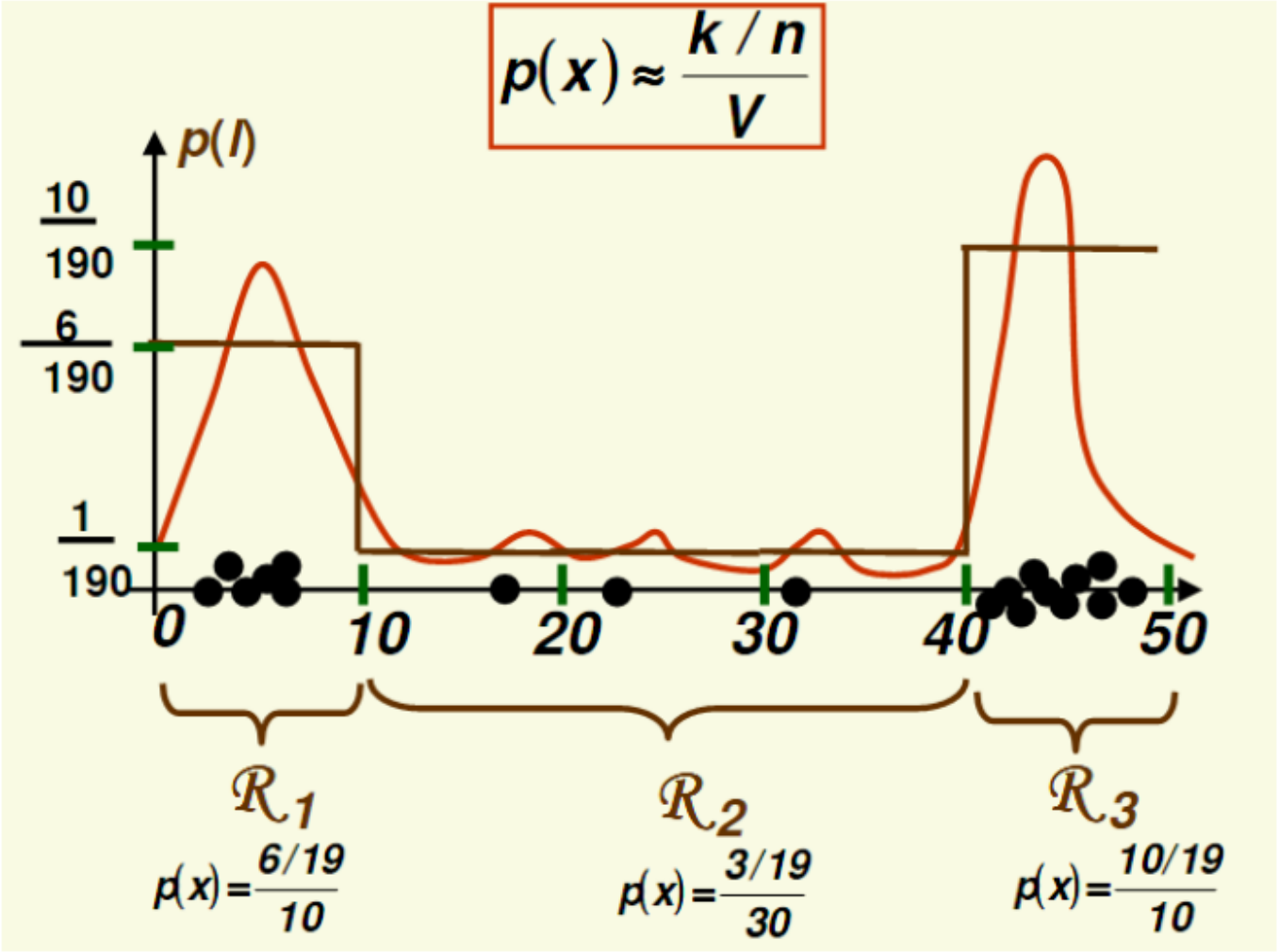
- Formalizando a estimativa da densidade
  - Podemos ainda definir essa probabilidade como sendo

$$P(x) = k/n$$

    - *k:* número de pontos na região *R*
    - *n:* número de pontos visando toda a região *R*
  - Como $P(x) = p(x) * V$ então

$$p(x) \approx \frac{k/n}{V}$$

Estimando a densidade

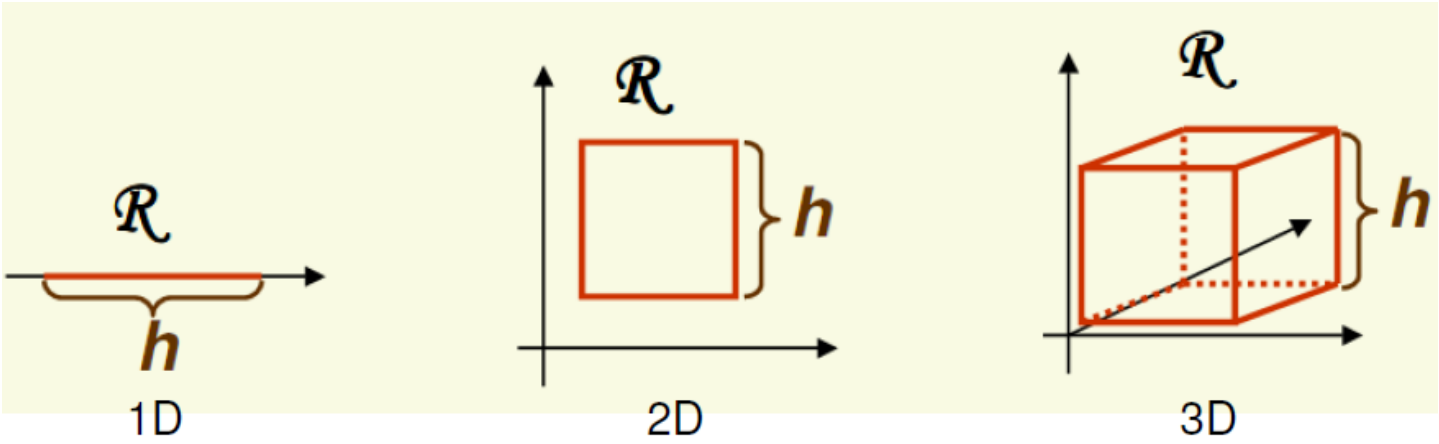


Janela de Parzen

- Técnica de interpolação de dados usada para a estimativa da densidade
  - Nela, fixamos o tamanho da região *R* e o volume *V* para estimar a densidade
  - O valor de *k* é determinado a partir dos dados de aprendizagem.

Janela de Parzen

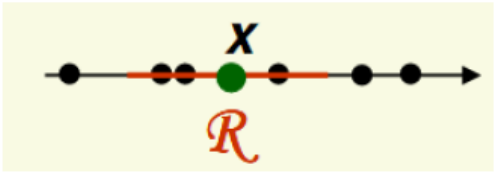
- Assume-se que a região *R* é um hiperplano de tamanho *h* e *d* dimensões
  - *R* é um cubo de *d* dimensões
  - Volume de *R* é h^d



Janela de Parzen

- Para estimar a densidade no ponto *x*
  - Centrar *R* em *x*
  - Contar o número de observações em *R* e substituir na equação

$$p(x) \approx \frac{k/n}{V}$$

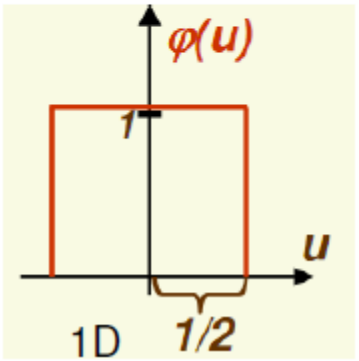


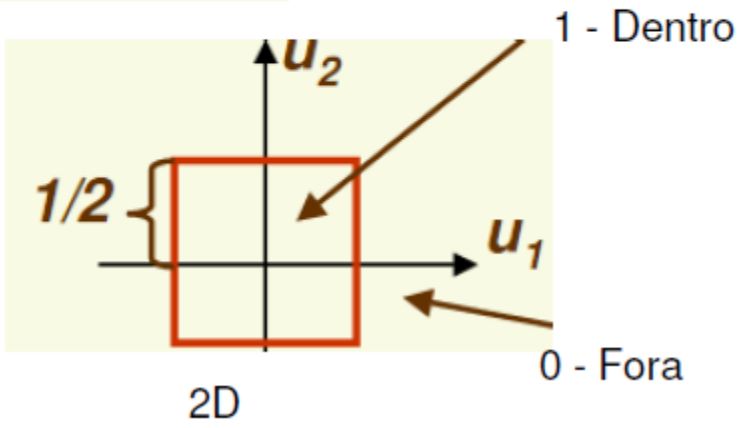
$$p(x) \approx \frac{3/6}{10}$$

Janela de Parzen

- Função de Kernel ($\varphi(u)$)
  - Expressão utilizada para encontrar a quantidade de pontos que caem em R

$$\varphi(u) = \begin{cases} 1 & |u_j| \leq \frac{1}{2} \quad j = 1, \ldots, d \\ 0 & \text{Caso contrário} \end{cases}$$





Janela de Parzen

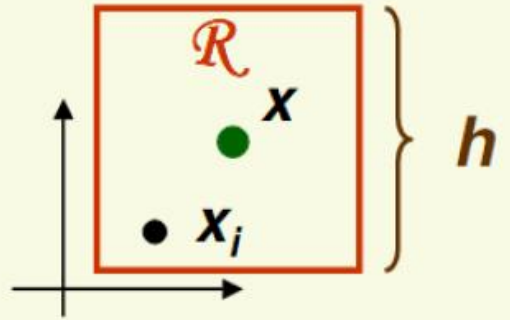
- Existem vários tipos de Kernel
  - I(|u| < 1) retorna 1 se verdade ou 0 se falso

Kernel	$\varphi(u)$
Uniforme	$\frac{1}{2} * I(\|u\| < 1)$
Triangular	$(1-\|u\|) * I(\|u\| < 1)$
Epanechnikov	$\frac{3}{4}*(1 - u^2)^2 * I(\|u\| < 1)$
Quadrático	$15/16 * (1 - u^2)^2 * I(\|u\| < 1)$
Triweight	$35/32 * (1 - u^2)^3 * I(\|u\| < 1)$
Cosseno	$\pi/4 * \cos(\pi/2 * u) * I(\|u\| < 1)$

Janela de Parzen

- Exemplo com Função de Kernel Uniforme
  - Considere $u = (x - x_i)/h$, onde x é o centro da região R e x_i é um ponto qualquer
  - Se x_i estiver dentro do hiperplano
    - φ retorna 1
    - Caso contrário, φ retorna 0

$$\varphi\left(\frac{x - x_i}{h}\right) = \begin{cases} 1 & |x - x_i| \le \frac{h}{2} \quad j = 1, \ldots, d \\ 0 & \text{otherwise} \end{cases}$$



Aprendizado baseado em instâncias

- Também conhecido como aprendizado baseado em memória
  - Algoritmos que comparam novas instâncias do problema com instâncias vistas no treinamento
  - Evitam generalização explícita
- Raciocínio baseado em Casos
- É um tipo de *lazy learning*
  - aprendizado preguiçoso

Aprendizado baseado em instâncias

- *Lazy learning* ou aprendizado preguiçoso
  - Método de aprendizagem em que a generalização além dos dados de treinamento é adiada até que uma consulta seja feita ao sistema
  - Diferente da aprendizagem ansiosa (*eager learning*)
    - Sistema tenta generalizar os dados de treinamento antes de receber uma consulta

Aprendizado baseado em instâncias

- Vantagens
  - Capacidade de adaptar seu modelo a dados nunca vistos
  - São úteis para grandes conjuntos de dados que possuem poucos atributos cada

Aprendizado baseado em instâncias

- Desvantagens
  - O custo de classificação de uma nova instância é alto
  - Necessita de grande espaço para armazenar todo o conjunto de dados de treinamento
  - São geralmente mais lentos para avaliar

Algoritmo K-NN

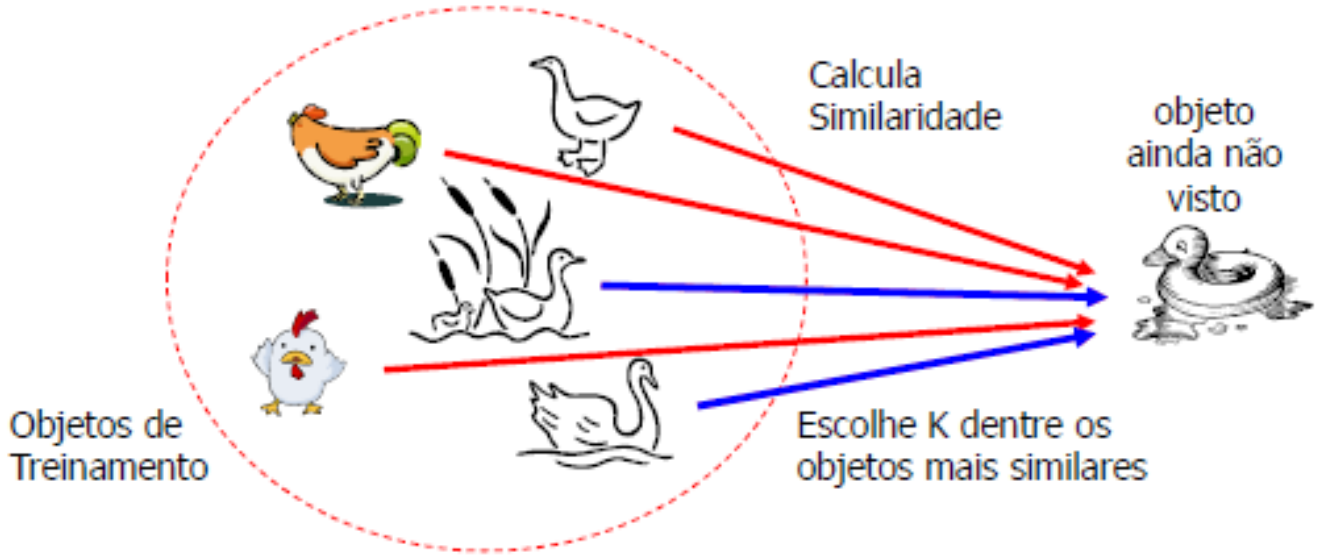
- Algoritmo *K-Nearest Neighbor*
  - Também conhecido com K-Vizinhos-Mais-Próximos
  - É um dos algoritmos mais básicos, simples e bem difundidos do paradigma baseado em instâncias
    - Aprendizagem “*lazy*”
    - Sem etapa de pré-processamento

Algoritmo K-NN

- Algoritmo *K-Nearest Neighbor*
  - Baseado unicamente na combinação de exemplos de treino armazenados.
    - Em geral, é definido em termos da distância Euclidiana
    - Pode ser utilizada também a distância de Mahalanobis
  - Aprendizagem supervisionada, algoritmo não-paramétrico

Algoritmo K-NN

- Idéia básica
  - Usa as *k* instâncias mais similares (vizinhos mais próximos) para classificar
    - Contagem de votos
    - Se anda como um pato, “quacks” como um pato, então provavelmente é um pato



Algoritmo K-NN

- Sua utilização requer 3 coisas
  - A base de dados de treinamento
  - Uma medida de (dis)similaridade
    - Usada entre os objetos desconhecidos e a base (objetos conhecidos)
  - O valor de ***k***
    - Qual a quantidade de vizinhos mais próximos a recuperar

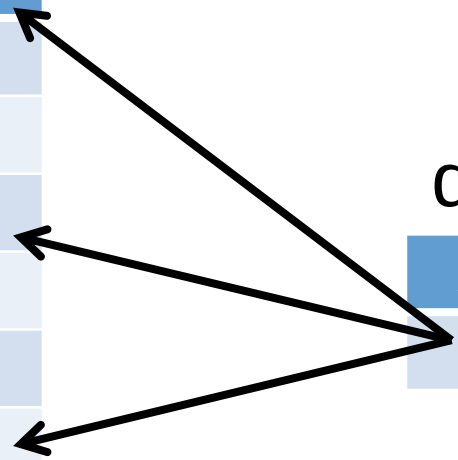
Algoritmo K-NN

- O algoritmo armazena os dados de treinamento em uma tabela
  - Cada instância correspondem a um ponto no espaço n-dimensional, R^n
  - Estes dados são usados para predizer a qual classe pertence uma instâncias desconhecida
    - Cálculo da distância

Algoritmo K-NN

Dados armazenados

At1	...	AtN	Classe
			1
			2
			1
			3
			1
			3
			2



Amostra desconhecida

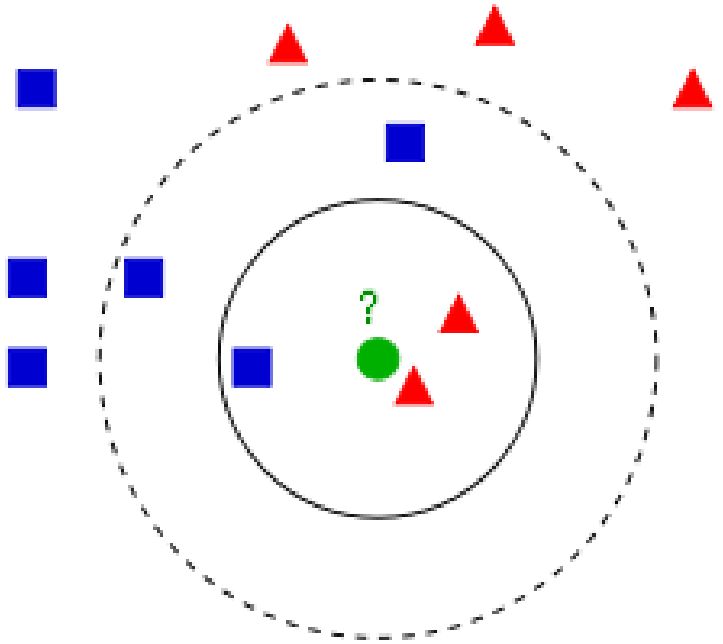
At1	...	AtN

Algoritmo K-NN

- Como classificar um objeto desconhecido?
  - Calcule a distância dele para todos os objetos do treinamento
  - Obtenha os ***k*** objetos do treinamento mais similares (mais próximos)
  - Classifique o objeto desconhecido como pertencente a classe da maioria dos ***k*** vizinhos

Algoritmo K-NN

- Qual a classe do círculo verde?
  - k = 3: triângulo vermelho
  - k = 5: quadrado azul

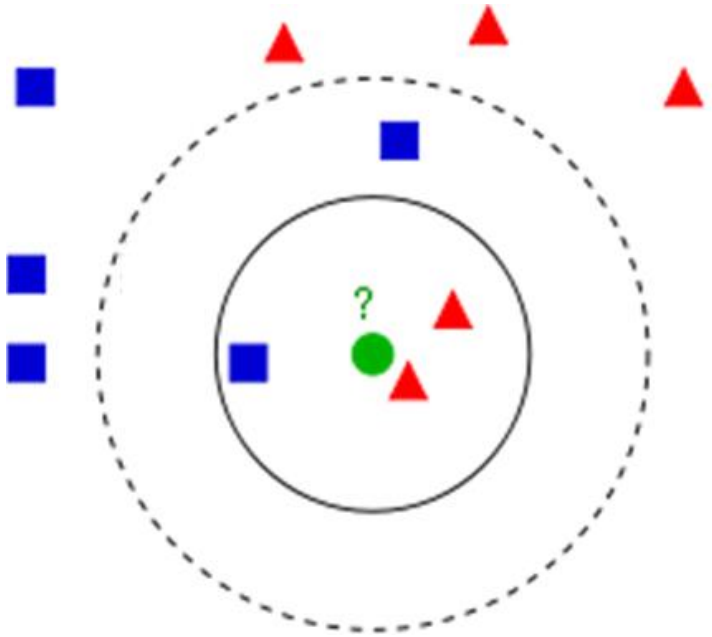


Algoritmo K-NN

- O que fazer em caso de empate entre duas ou mais classes?
  - Considerar apenas os ***k-1*** vizinhos mais próximos.
  - Em caso de novo empate, repetir esse processo
  - Esse processo para quando ama classe for unânime

Algoritmo K-NN

- Qual a classe do círculo verde?
  - k = 4: empate
  - k = 3: triângulo vermelho



Algoritmo K-NN

- Como escolher o valor de ***k***
  - k muito pequeno
    - O algoritmo se torna mais flexível
    - Sensível a ruído, a classificação das amostras pode ser instável
    - Menor gasto computacional

Algoritmo K-NN

- Como escolher o valor de ***k***
  - k muito grande
    - Mais robusto a ruído
    - Aumenta a informação sobre a probabilidade de pertencer à classe
    - Vizinhança tende a incluir objetos de outras classes, privilegiando a majoritária
    - Menor flexibilidade
    - Maior gasto computacional

Algoritmo K-NN

- Necessita de normalização dos dados
  - Condicionar os dados de forma apropriada
  - Isso evita que certos atributos dominem completamente a medida de distância
  - Exemplo:
    - Altura de um adulto: 1.4m a 2.1m
    - Peso de um adulto: 50Kg a 130Kg
    - Faixa salarial: R$400 a R$30.000

Algoritmo K-NN

- Vantagens
  - Simples de implementar
  - Não requer uma etapa de treinamento
  - Ideal para conjuntos de dados pequenos ou médios
  - Usa informação local, podendo ser implementado comportamentos adaptativos
  - Pode ser paralelizado

Algoritmo K-NN

- Desvantagens
  - Custo computacional e de armazenamento alto para conjuntos de dados grandes ou com muitos atributos
  - A constante ***k*** usada para definir o número de vizinhos é obtida por tentativa e erro
  - Sua precisão pode ser severamente degradada pela presença de ruído ou atributos irrelevantes

Algoritmo K-NN | Exemplo

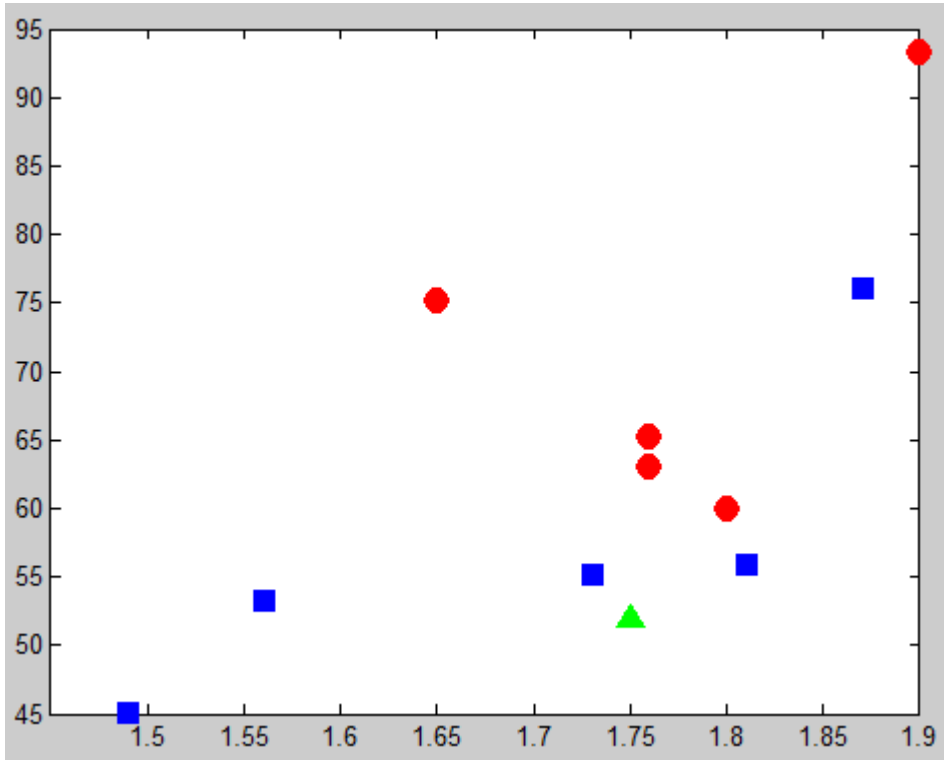
- Qual a classe da amostra, dado o conjunto de treinamento ao lado?

Altura	Peso	Sexo
1,75	52,0	?

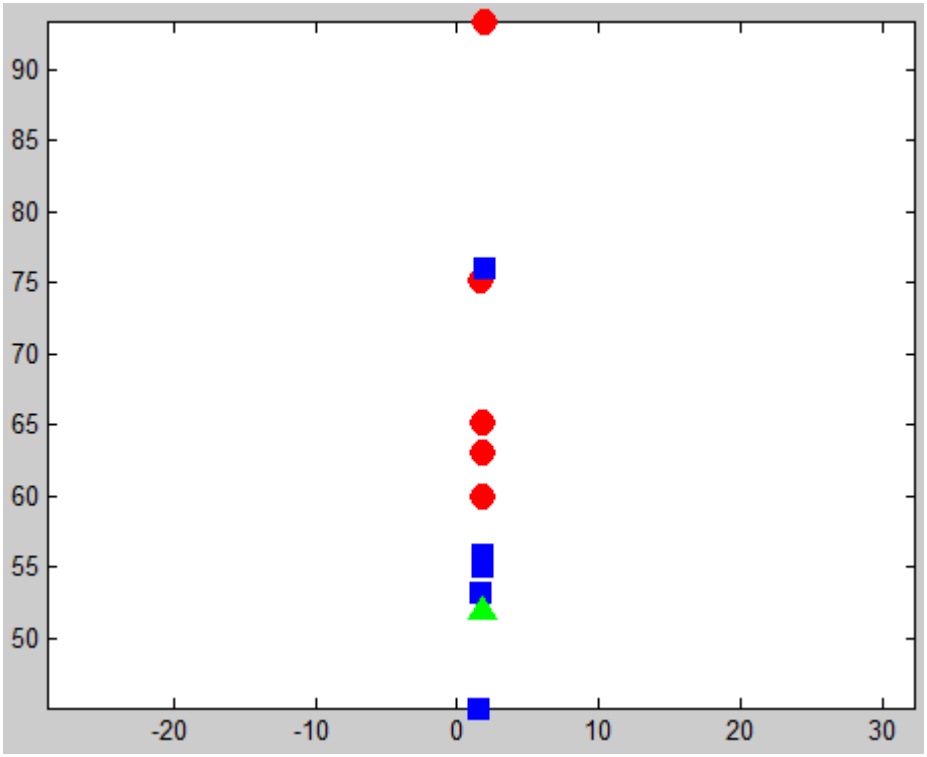
Altura	Peso	Sexo
1,87	76,1	0
1,65	75,2	1
1,80	60,0	1
1,81	55,9	0
1,90	93,3	1
1,74	65,2	1
1,49	45,1	0
1,56	53,2	0
1,73	55,1	0
1,76	63,1	1

Algoritmo K-NN | Exemplo

Com ajuste de escala



Sem ajuste de escala



Algoritmo K-NN | Exemplo

- Primeiro passo
  - Normalizar os valores
    - z-score

Média Altura	Média Peso
1,73	64,22

Desvio Altura	Desvio Peso
0,13	14,01

Altura	Peso	Sexo
1,87	76,1	0
1,65	75,2	1
1,80	60,0	1
1,81	55,9	0
1,90	93,3	1
1,74	65,2	1
1,49	45,1	0
1,56	53,2	0
1,73	55,1	0
1,76	63,1	1

Algoritmo K-NN | Exemplo

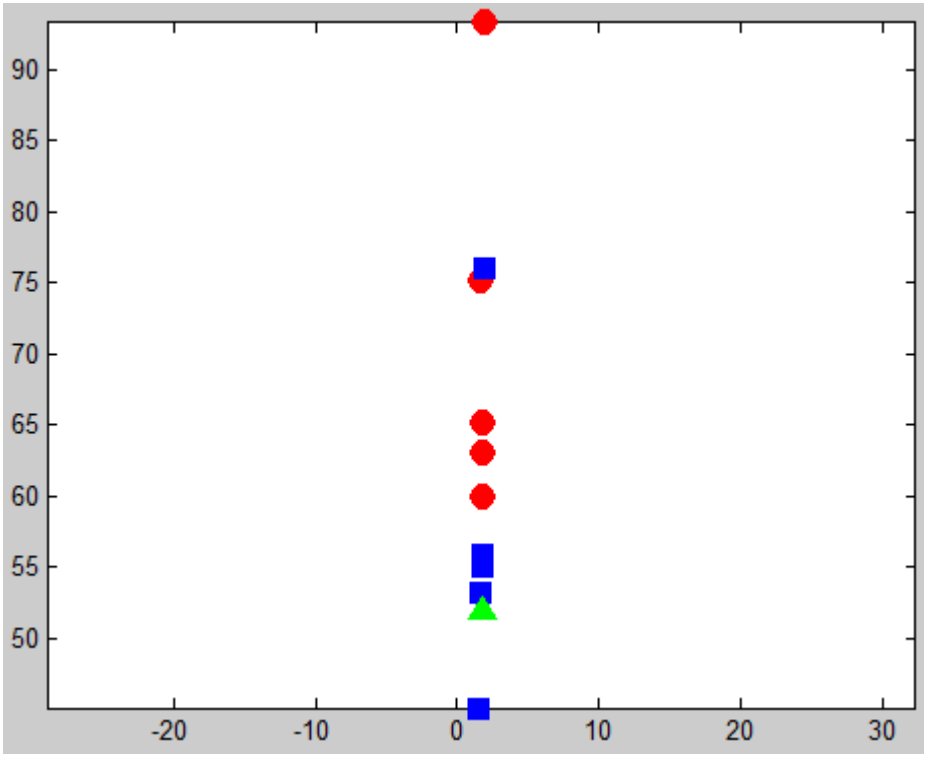
- Primeiro passo
  - Normalizar os valores
    - z-score

Altura	Peso	Sexo
0,14	-0,87	?

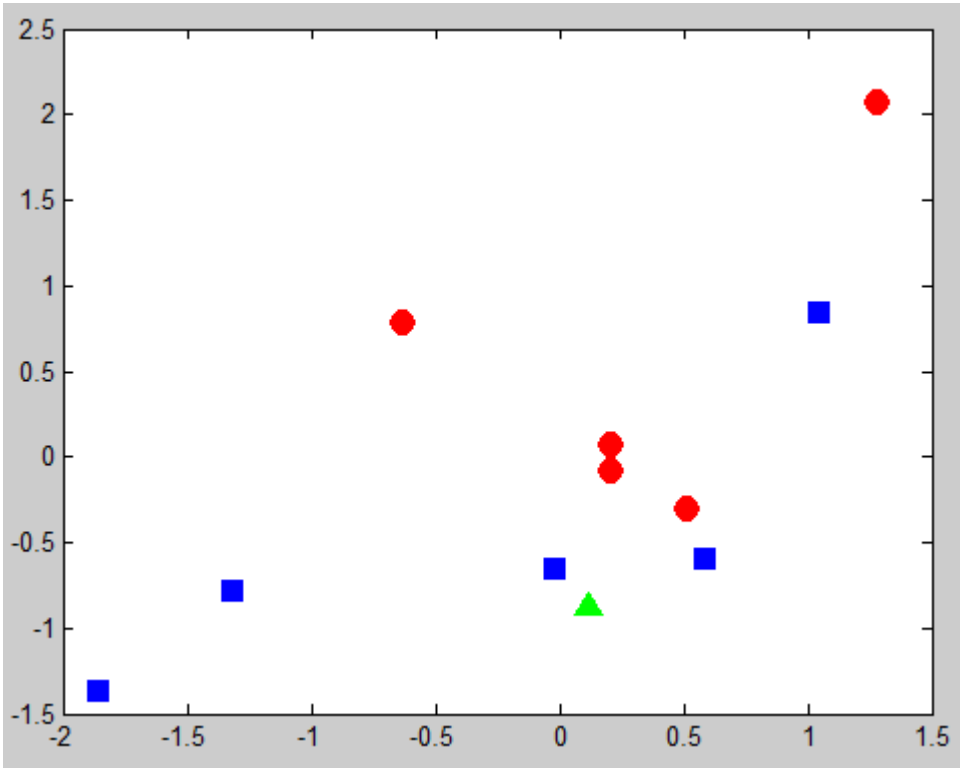
Altura	Peso	Sexo
1,04	0,84	0
-0,63	0,78	1
0,51	-0,30	1
0,58	-0,59	0
1,27	2,07	1
0,20	0,06	1
-1,85	-1,36	0
-1,32	-0,78	0
-0,02	-0,65	0
0,20	-0,07	1

Algoritmo K-NN | Exemplo

Sem ajuste de escala



Z-score



Algoritmo K-NN | Exemplo

- Segundo passo
  - Calcular as distâncias da amostra desconhecida para as conhecidas

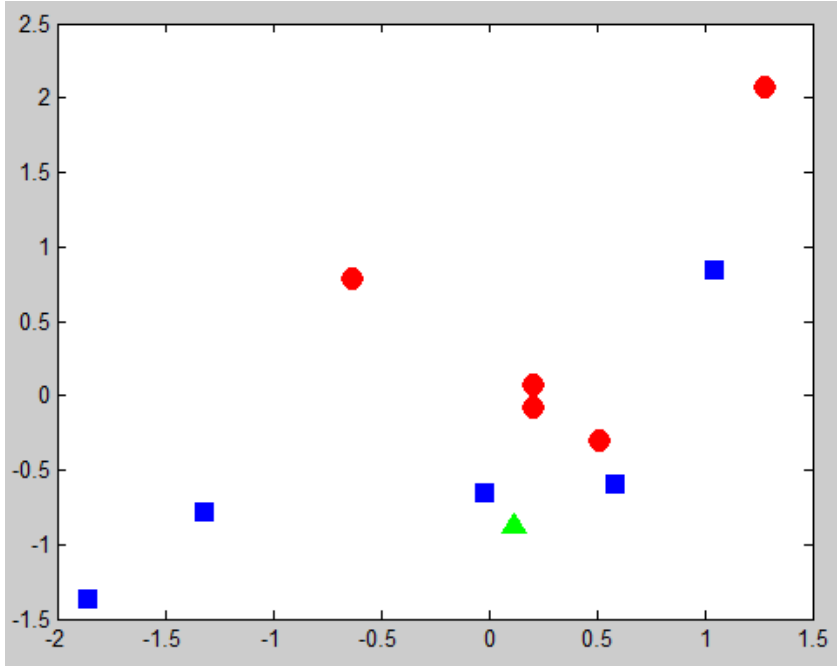
Altura	Peso	Sexo
0,14	-0,87	?

Altura	Peso	Sexo	D
1,06	0,84	0	1,95
-0,62	0,78	1	1,82
0,53	-0,30	1	0,69
0,60	-0,59	0	0,54
1,29	2,07	1	3,16
0,07	0,07	1	0,94
-1,84	-1,36	0	2,05
-1,31	-0,79	0	1,46
-0,01	-0,65	0	0,27
0,22	-0,08	1	0,79

Algoritmo K-NN | Exemplo

- Terceiro passo
  - Classificação: k = 3

Altura	Peso	Sexo
0,12	-0,87	? = 0

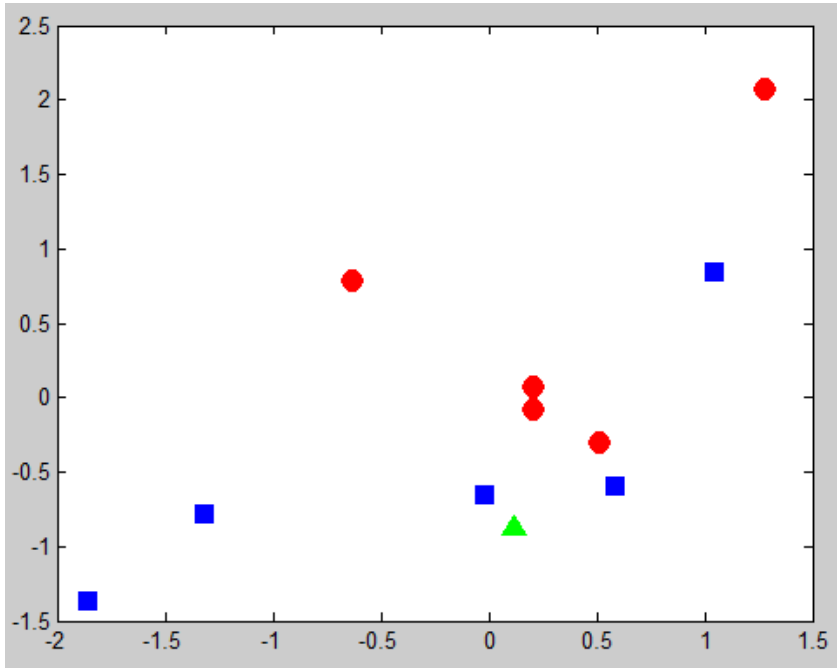


Altura	Peso	Sexo	D
1,06	0,84	0	1,95
-0,62	0,78	1	1,82
0,53	**-0,30**	**1**	**0,69**
0,60	**-0,59**	**0**	**0,54**
1,29	2,07	1	3,16
0,07	0,07	1	0,94
-1,84	-1,36	0	2,05
-1,31	-0,79	0	1,46
-0,01	**-0,65**	**0**	**0,27**
0,22	-0,08	1	0,79

Algoritmo K-NN | Exemplo

- Terceiro passo
  - Classificação: k = 5

Altura	Peso	Sexo
0,12	-0,87	? = 1



Altura	Peso	Sexo	D
1,06	0,84	0	1,95
-0,62	0,78	1	1,82
0,53	**-0,30**	**1**	**0,69**
0,60	**-0,59**	**0**	**0,54**
1,29	2,07	1	3,16
0,07	**0,07**	**1**	**0,94**
-1,84	-1,36	0	2,05
-1,31	-0,79	0	1,46
-0,01	**-0,65**	**0**	**0,27**
0,22	**-0,08**	**1**	**0,79**

Algoritmo 1-NN

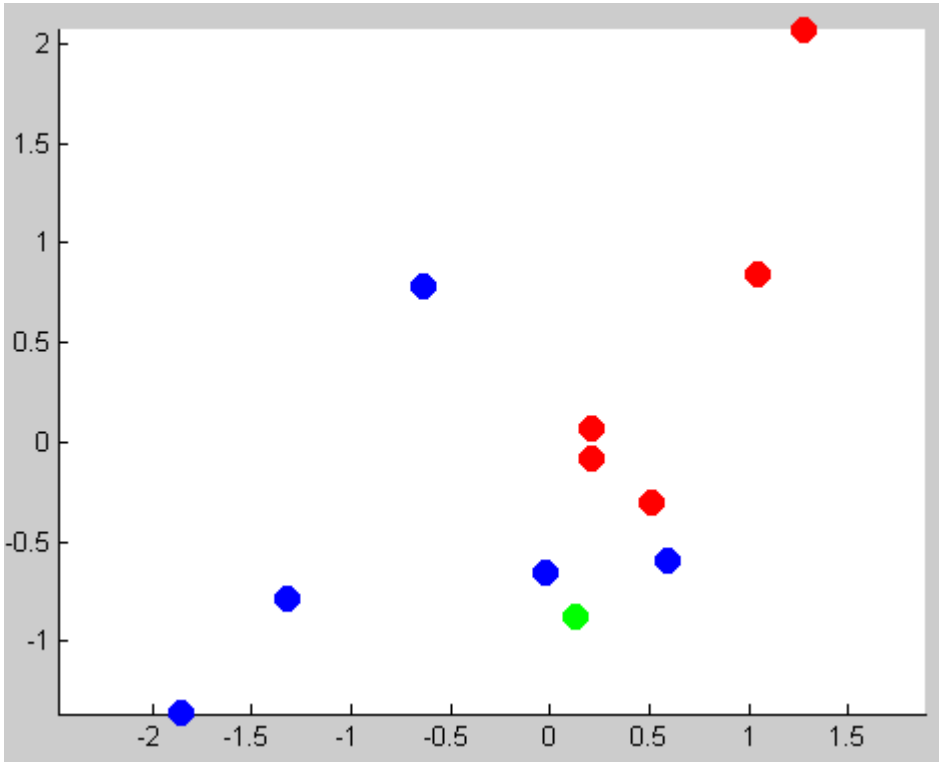
- Uma amostra desconhecida é considerada como pertencente a mesma classe da amostra conhecida que apresentar a menor distância até ela
  - É um caso particular do algoritmo K-NN
    - Nesse caso, k = 1
  - Qual a classe do círculo verde?

Nearest Prototype Classifier

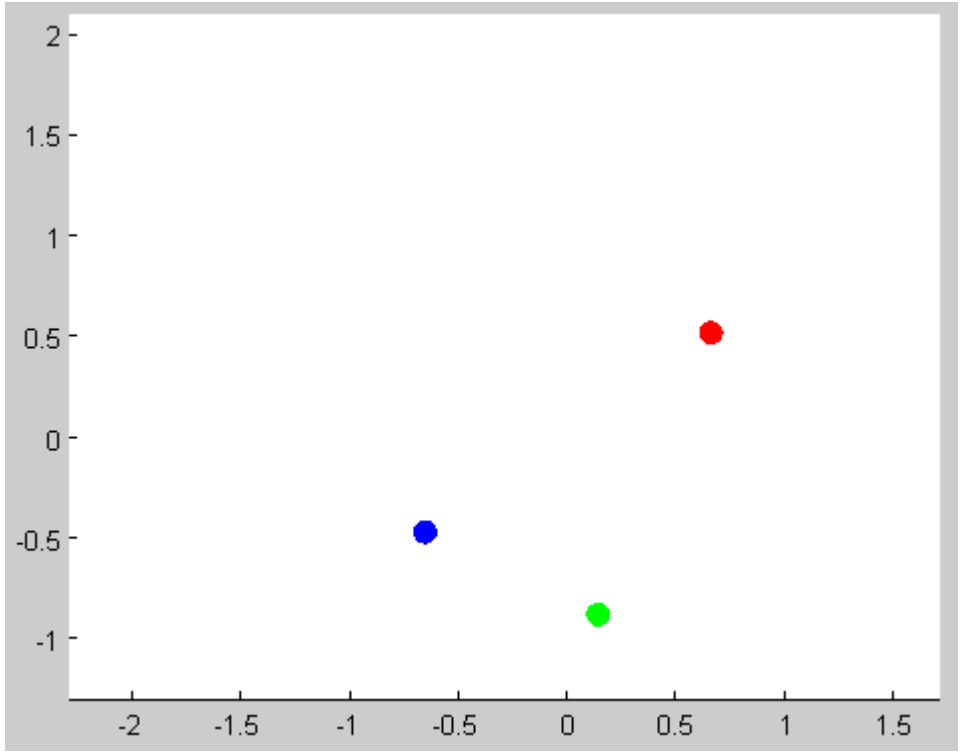
- Também conhecido como algoritmo do *centroide mais próximos*
  - Similar ao algoritmo 1-NN
  - Atribui uma observação à classe de amostras de treinamento cujo centroide (média) se encontra mais próximo

Nearest Prototype Classifier

Z-score



Centroides



Distância Euclidiana | Limitações

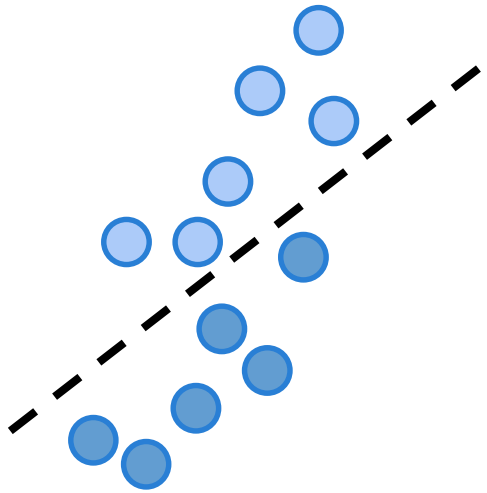
- Se um classificador baseado em distância mínima tem um elevado número de classificações corretas, não há razão para procurar classificadores mais complexos
  - No entanto, é frequente ocorrer um baixo desempenho

Distância Euclidiana | Limitações

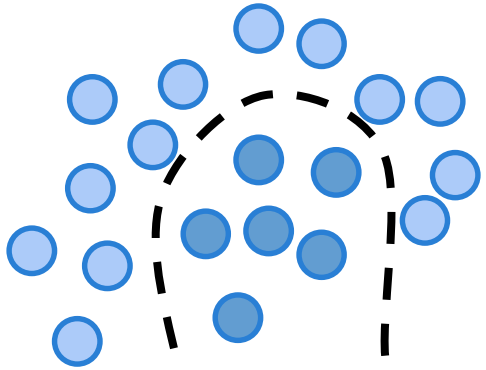
- Por que isso ocorre?
  - Atributos inadequados
  - Atributos correlacionados
  - As superfícies de decisão podem ser não-lineares
  - Existência de subclasses distintas
  - Espaço de padrões muito complexo

Distância Euclidiana | Limitações

- Atributos correlacionados

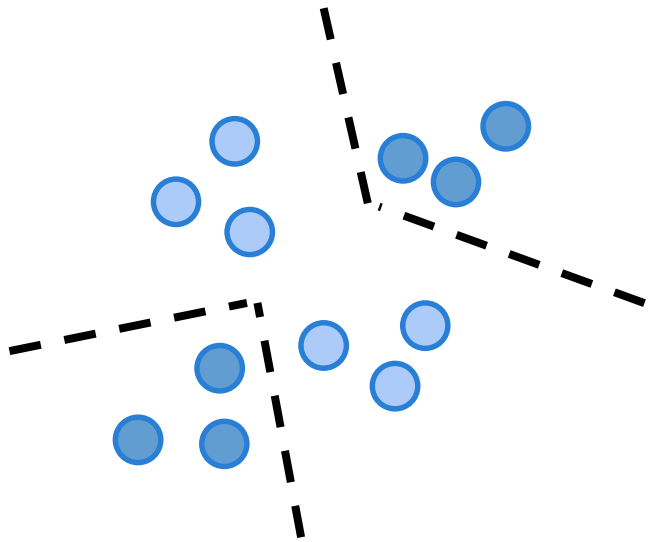


- As superfícies de decisão podem ser não-lineares

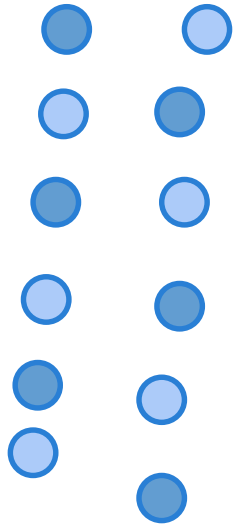


Distância Euclidiana | Limitações

- Existência de subclasses distintas



- Espaço de padrões muito complexo



Maldição da dimensionalidade

- Suponha o seguinte problema
  - Um conjunto de dados é descrito por 20 atributos
  - Destes, apenas 2 são relevantes
    - Os demais são atributos ruins ou correlacionados
  - O resultado será um mau desempenho na classificação
- O algoritmo K-NN é normalmente enganado quando o número de atributos é grande

Maldição da dimensionalidade

- Maldição da dimensionalidade (ou *Curse of dimensionality*)
  - Termo que se refere a vários fenômenos que surgem na análise de dados em espaços com muitas dimensões (atributos)
    - Muitas vezes com centenas ou milhares de dimensões
  - Basicamente, adicionar características não significa sempre melhora no desempenho de um classificador

Maldição da dimensionalidade

- Teorema do patinho feio (de Watanabe)
  - Caso haja um conjunto suficientemente grande de características em comum, sem uma outra referência previamente estabelecida, é possível fazer com que dois padrões arbitrários sejam considerados similares.
  - Um cisne e um pato e um par de cisnes podem ficar igualmente similares

Maldição da dimensionalidade

- Um grande número de atributos tende a gerar informação redundante
  - Isso prejudica o desempenho do sistema
- Classificar dados significa encontrar grupos com propriedades similares
  - Em espaços com muitas dimensões as amostras se tornan esparsas e pouco similares
  - Isso impossibilita estratégias comuns de organização dos dados de serem eficiente.

Maldição da dimensionalidade

- É possível evitar isso?
  - O número de amostras de treinamento deve aumentar exponencialmente com o aumento do número de atributos
  - Isso nem sempre é possível na prática
    - De onde tirar novos dados?

Maldição da dimensionalidade

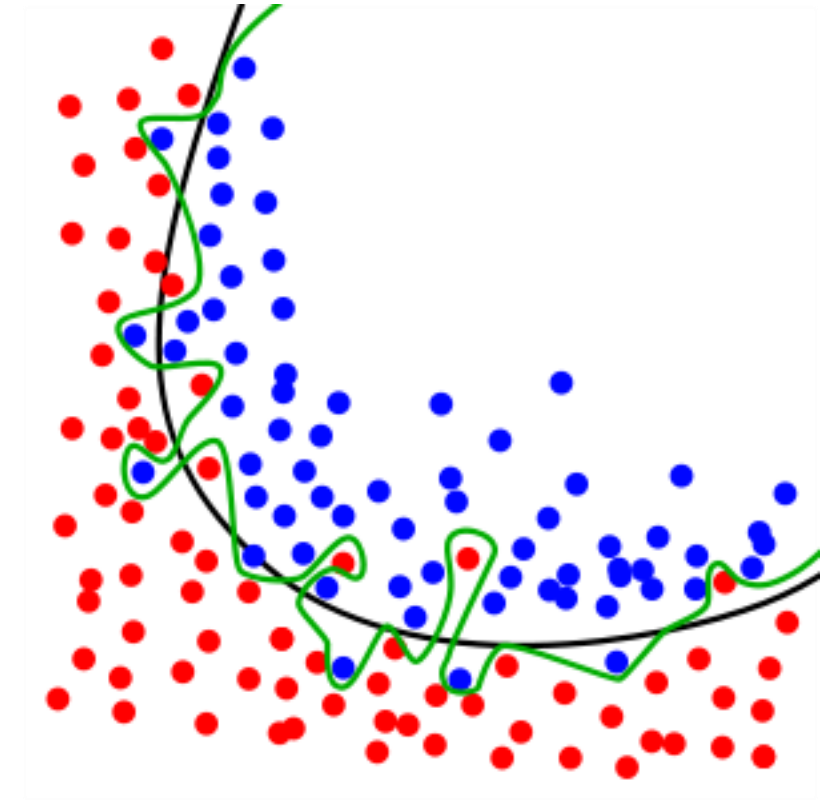
- É possível evitar isso?
  - Podemos reduzir o número de dimensões do espaço de características
    - Etapa importante no projeto de um sistema de classificação
    - Pode ser feita selecionando e/ou compondo as características mais adequadas
    - Uso de técnicas de seleção e combinação de características

Overfitting

- Fenômeno que ocorre quando o modelo estatístico se ajusta em demasiado ao conjunto de dados/amostra
  - Também conhecido com **sobrejuste** ou **over-training**
  - Ao invés de aprender o padrão, o modelo estatístico aprende suas “esquisitices”

Overfitting

- É comum haver desvios causados por erros de medição ou fatores aleatórios nas amostras
  - No *overfitting*, o modelo se ajusta a estes desvios e se esquece do comportamento geral dos dados

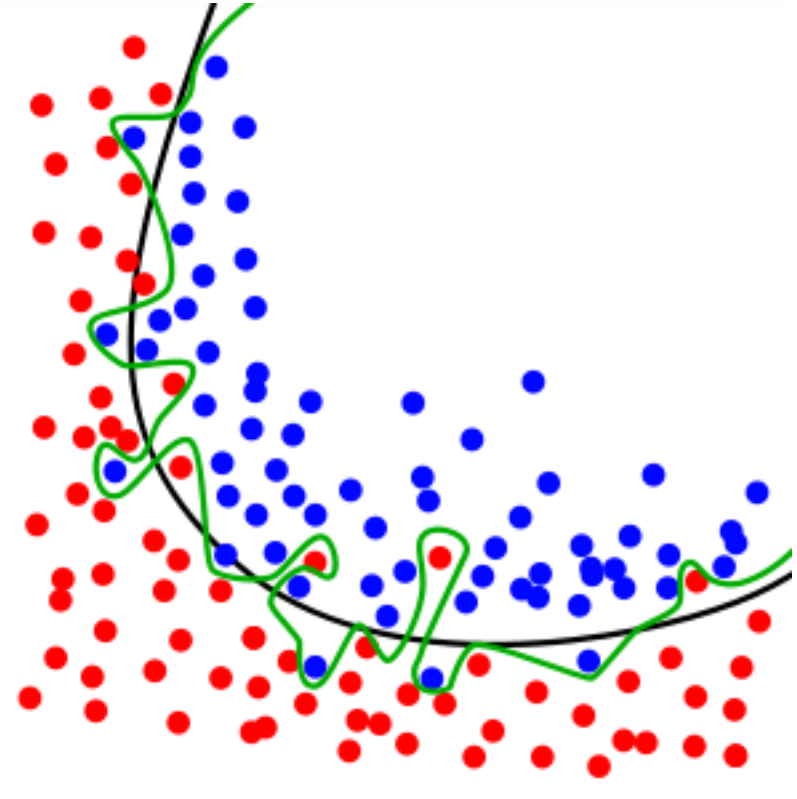


Overfitting

- Modelo com *overfitting*
  - Se degenera e se especializa no conjunto de treinamento
    - Alta precisão quando testado com seu conjunto de treinamento
  - Esse modelo não representa a realidade e deve ser evitado
    - Não tem capacidade de generalização.

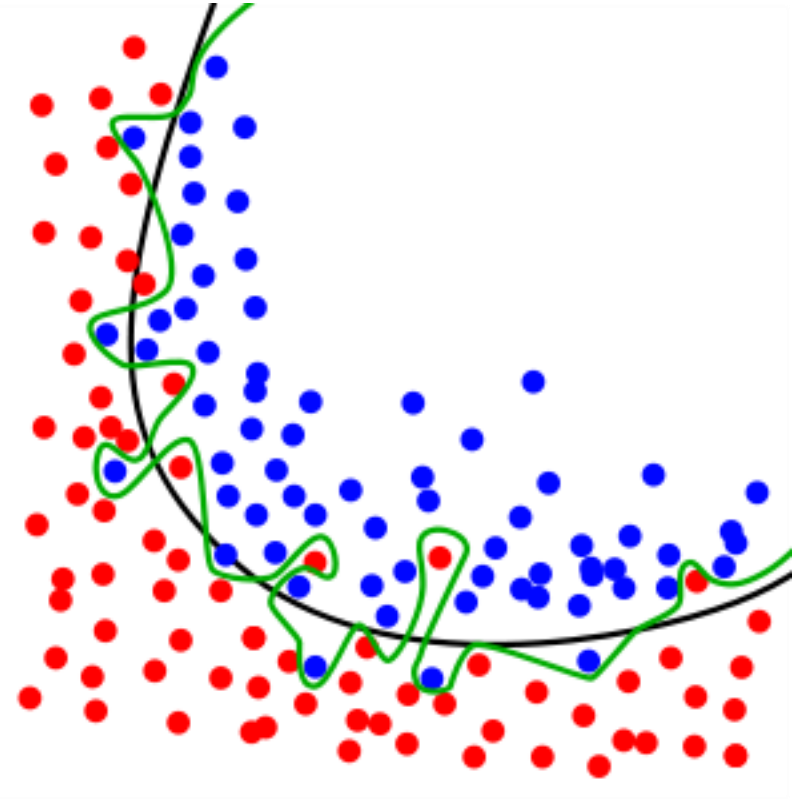
Overfitting

- Simplicidade é a resposta
  - Ajuda a proteger contra *overfitting*
    - É preferível um classificador que erre os dados estranhos no pressuposto de que eles são, de fato, estranhos e sem valor preditivo.



Overfitting

- Simplicidade é a resposta
  - Navalha de Occam
    - "*Se em tudo o mais forem idênticas as várias explicações de um fenômeno, a mais simples é a melhor*“ - William de Ockham (século XIV)



Overfitting

- Como contornar esse problema?
  - Regularização
    - Manter todos os atributos, mas reduzir a magnitude/valor dos deles
    - Penalizar atributos pela imposição de restrições de suavidade ao modelo de aproximação

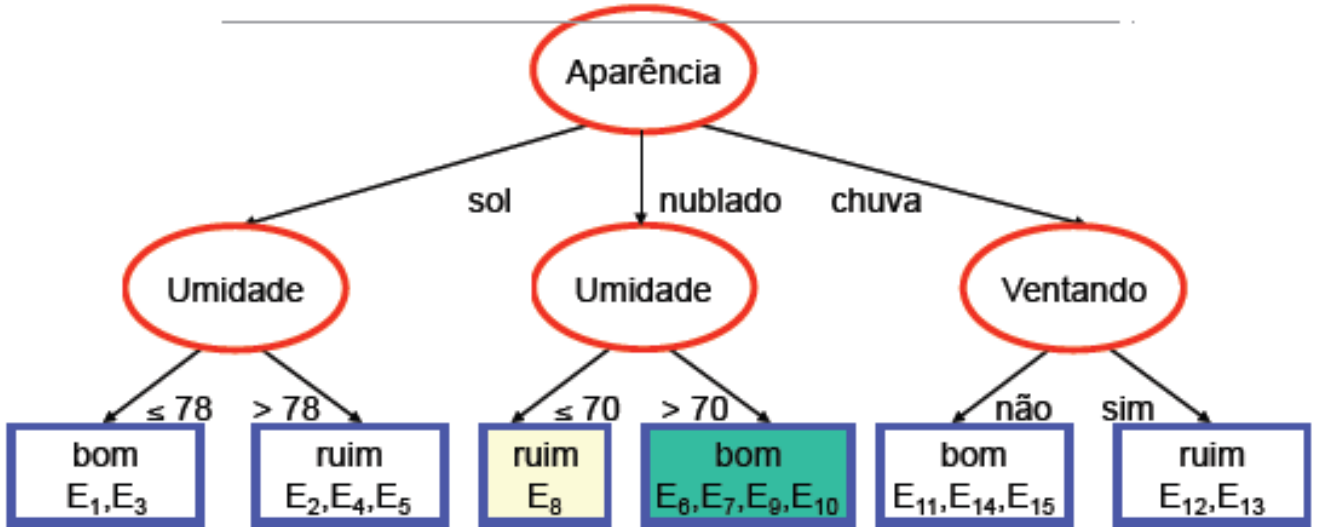
Overfitting

- Como contornar esse problema?
  - Métodos de poda (*pruning*)
    - Voltados para árvores de decisão
    - Muitas das arestas ou sub-árvores podem refletir ruídos ou erros.
    - Necessidade de detectar e excluir essas arestas e sub-árvores
    - Smplifica a árvore e facilita sua interpretabilidade por parte do usuário

Overfitting

- Árvore de Decisão
  - Sem poda

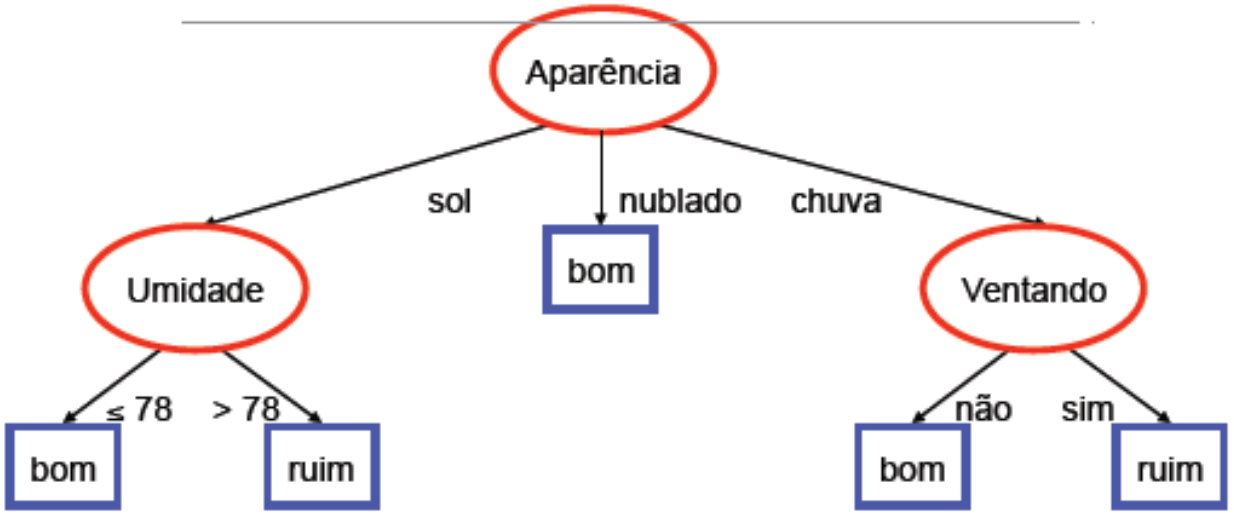
Exemplo	Aparência	Temperatura	Umidade	Ventando	Viajar
E_1	sol	25	72	sim	bom
E_2	sol	28	91	sim	ruim
E_3	sol	22	70	não	bom
E_4	sol	23	95	não	ruim
E_5	sol	30	85	não	ruim
E_6	nublado	23	90	sim	bom
E_7	nublado	29	78	não	bom
E_8	nublado	19	65	sim	ruim
E_9	nublado	26	75	não	bom
E_{10}	nublado	20	87	sim	bom
E_{11}	chuva	22	95	não	bom
E_{12}	chuva	19	70	sim	ruim
E_{13}	chuva	23	80	sim	ruim
E_{14}	chuva	25	81	não	bom
E_{15}	chuva	21	80	não	bom



Overfitting

- Árvore de Decisão
  - Depois da poda

Exemplo	Aparência	Temperatura	Umidade	Ventando	Viajar
E_1	sol	25	72	sim	bom
E_2	sol	28	91	sim	ruim
E_3	sol	22	70	não	bom
E_4	sol	23	95	não	ruim
E_5	sol	30	85	não	ruim
E_6	nublado	23	90	sim	bom
E_7	nublado	29	78	não	bom
E_8	nublado	19	65	sim	ruim
E_9	nublado	26	75	não	bom
E_{10}	nublado	20	87	sim	bom
E_{11}	chuva	22	95	não	bom
E_{12}	chuva	19	70	sim	ruim
E_{13}	chuva	23	80	sim	ruim
E_{14}	chuva	25	81	não	bom
E_{15}	chuva	21	80	não	bom

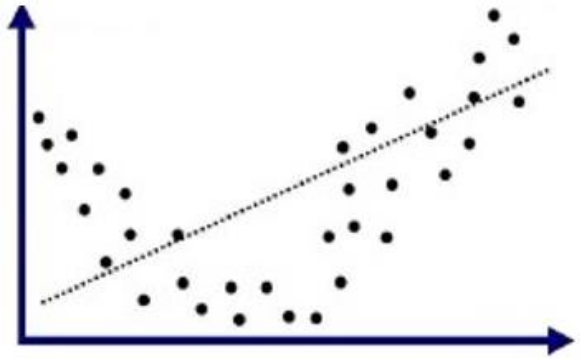


Overfitting

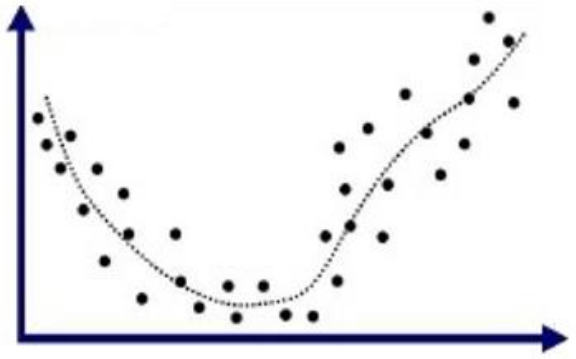
- Como contornar esse problema?
  - *Cross-validation*
    - Consiste em separar os dados em Treinamento e Teste
    - Essa divisão dos dados em subconjuntos ajuda a evitar que o modelo aprenda as particularidades dos dados

Underfitting

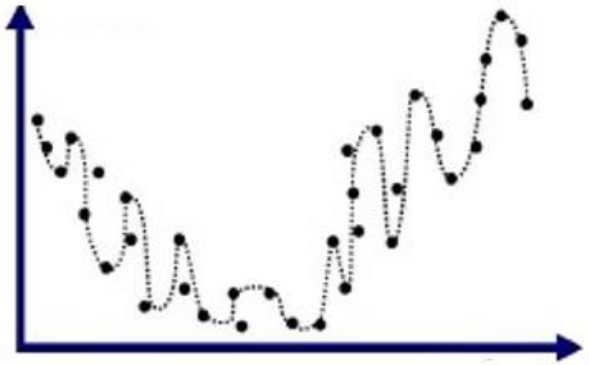
- *Underfitting* ou sub-ajuste
  - O modelo não consegue aprender o suficiente
  - O modelo pode ser muito simples



Underfitting: não aprendeu!



Modelo adequado



Overfitting: decorou o treinamento!

Agradecimentos

- Agradeço aos professores
  - Guilherme de Alencar Barreto - Universidade Federal do Ceará (UFC)
  - Prof. Ricardo J. G. B. Campello – ICMC/USP
- pelo material disponibilizado