

APRENDIZAGEM BASEADA EM INSTÂNCIAS (KNN)

Inteligência Artificial

André Câmara



Slides adaptados do professor Ricardo Prudêncio (Cin-UFPE)

Introdução

- Soluções para novos problemas podem ser definidas, adaptando soluções dadas a problemas similares
- É comum memorizarmos situações e recuperá-las quando necessário
- Procedimento usado no dia-a-dia



Aprendizado Baseado em Instâncias

- Problemas resolvidos no passado são representados como *instâncias*
 - E.g., exemplos de treinamento previamente etiquetados
- Instâncias são *recuperadas* e *adaptadas* para resolver novos problemas



Aprendizado Baseado em Instâncias

- Classe de algoritmos de aprendizado que inclui, por exemplo:
 - K-Vizinhos Mais Próximos
 - Raciocínio Baseado em Casos



ALGORITMO DE K-VIZINHOS MAIS PRÓXIMOS

K-Nearest Neighbors (k-NN)



Algoritmo k-NN

- Todas as instâncias correspondem a *pontos* em um espaço n-dimensional
- Vizinhaça definida por uma função de *distância*, ou por uma função de *similaridade*
 - Menor distância = maior similaridade
- Classe de um novo exemplo é definida a partir dos *vizinhos mais próximos*



Algoritmo k-NN

- Definições:
 - x_i : instância descrita pelo vetor $\langle a_1(x_i), \dots, a_n(x_i) \rangle$
 - $f(x_i)$: classe de x_i
- Treinamento básico:
 - Armazenar exemplos de treinamento $\langle x_i, f(x_i) \rangle$

Algoritmo k-NN

- Dado exemplo x_q a ser classificado,
 - Seja x_1, \dots, x_k as *k instâncias mais similares* a x_q
 - Retorne classe *majoritária* das instâncias recuperadas

$$\hat{f}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

onde

$$\delta(v, f(x_i)) = 1 \quad \text{se} \quad v = f(x_i) \quad e$$

$$\delta(v, f(x_i)) = 0, \quad \text{caso contrário}$$

Algoritmo k-NN

- Algoritmo k-NN usa comumente a *Distância Euclidiana* para definição de vizinhança

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n (a_r(x_i) - a_r(x_j))^2}$$

Algoritmo k-NN

- Atributos de maior *escala numérica* podem dominar função de distância
- Usualmente, os atributos são normalizados para intervalo entre 0 e 1

$$a_{\text{NORM}}(x) = \frac{a_i(x) - \min(a_i(x))}{\max(a_i(x)) - \min(a_i(x))}$$

$$a_{\text{NORM}}(x) = \frac{a(x) - \text{mean}(a_i(x))}{\text{std}(a_i(x))}$$

Algoritmo k-NN

- Boa prática: incluir a normalização dos dados implicitamente no cálculo da distância

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n \frac{(a_r(x_i) - a_r(x_j))^2}{(\max(a_r(x_i)) - \min(a_r(x_j)))^2}}$$

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^n \frac{(a_r(x_i) - a_r(x_j))^2}{(\text{std}(a_r(x_i)))^2}}$$

Algoritmo k-NN

- Distância de Hamming para *atributos categóricos*:
 - Soma 1 para cada atributo cujo valor coincide nas instâncias

$$d_{\text{HAMMING}}(x_i, x_j) = \sum_{r=1}^n \text{dist}(a_r(x_i), a_r(x_j))$$

$$\text{dist}(a_r(x_i), a_r(x_j)) = \begin{cases} 1, & \text{se } a_r(x_i) \neq a_r(x_j) \\ 0, & \text{caso contrário} \end{cases}$$

Distância de Hamming Exemplos

- $d_{HAMMING} ("karolin", "kathrin") = 3.$
- $d_{HAMMING} ("karolin", "kerstin") = 3.$
- $d_{HAMMING} (1011101, 1001001) = 2.$
- $d_{HAMMING} (2173896, 2233796) = 3.$



Algoritmo k-NN

- Função de distância considerando *missing values* (Witten, Frank (2000, p.115)):
 - Para atributos categóricos: distância é igual a 1 na presença de valores faltosos
 - Para atributos numéricos:
 - Se os dois valores comparados são faltosos então distância igual a 1
 - Se apenas um dos valores é faltoso, então distância é o maior dentre os seguintes valores:
 - Tamanho normalizado do atributo presente
 - Um (1) menos o tamanho normalizado do atributo presente

Algoritmo k-NN

- Outras funções de distância:

- Distância L1 Normalizada

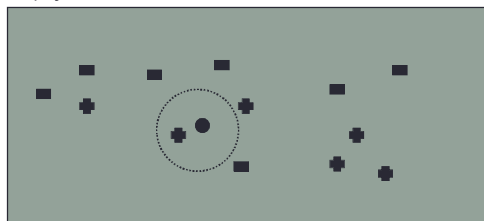
$$d(x_i, x_j) = \sum_{r=1}^n \frac{|a_r(x_i) - a_r(x_j)|}{\max(a_r(x_i)) - \min(a_r(x_j))}$$

- Distância Cosseno Normalizada

$$d(x_i, x_j) = \frac{\sum_{r=1}^n a_r(x_i) * a_r(x_j)}{\sum_{r=1}^n a_r(x_i)^2 * \sum_{r=1}^n a_r(x_j)^2}$$

Algoritmo k-NN - Exemplo

Espaço de instâncias

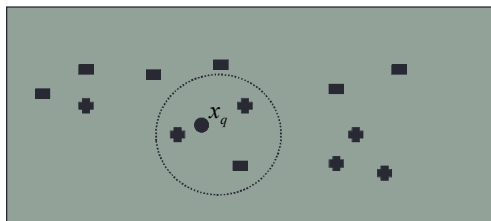


- Exemplos da classe negativa
- ✚ Exemplos da classe positiva
- Exemplos a ser classificado

- Com $k = 1$, exemplo x_q recebe classe positiva

Algoritmo k-NN - Exemplo

Espaço de instâncias

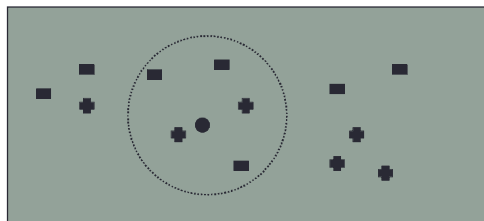


- Exemplos da classe negativa
- ✚ Exemplos da classe positiva
- Exemplos a ser classificado

- Com $k = 3$, exemplo x_q recebe classe positiva

Algoritmo k-NN - Exemplo

Espaço de instâncias

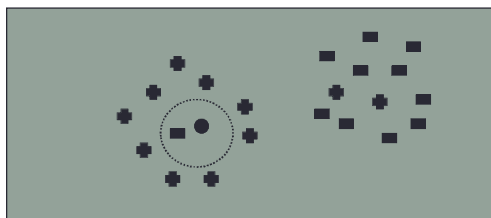


- Exemplos da classe negativa
- ✚ Exemplos da classe positiva
- Exemplos a ser classificado

- Com $k = 5$, exemplo x_q recebe classe negativa

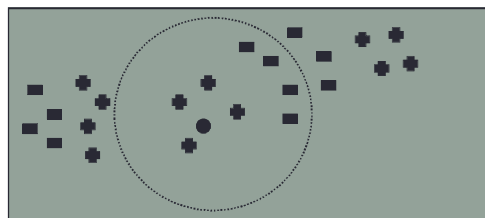
Algoritmo k-NN

- O *dilema* da escolha do parâmetro k
 - Valores muito baixos podem aumentar a contribuição de *exemplos ruidosos*



Algoritmo k-NN

- O *dilema* da escolha do parâmetro k
 - Valores muito altos podem aumentar a contribuição de exemplos *pouco similares*, e assim, *menos relevantes*



Algoritmo k-NN

- O valor do parâmetro k é escolhido comumente através de *tentativa-e-erro*
 - Avaliação empírica com diferentes valores de k
 - *Validação cruzada*

Algoritmo k-NN com Ponderação pela Distância

- A contribuição de cada vizinho pode ser ponderada pela distância com a instância a ser classificada

$$\hat{f}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^k w_i * \delta(v, f(x_i))$$

$$w_i = \frac{1}{d(x_q, x_i)^2} \quad w_i = \frac{1}{d(x_q, x_i)} \quad w_i = 1 - d(x_q, x_i)$$

Algoritmo k-NN com Ponderação pela Distância

- Com ponderação, a escolha adequada de k se tornaria menos importante?
 - Note que instâncias muito distantes teriam pouca contribuição na predição
- “There is no harm in allowing all training examples to have an influence on the classification...” – T. Mitchell (1997, p. 234)
- *Método de Shepard*: k-NN ponderado usando todos os exemplos de treinamento como vizinhos

Algoritmo k-NN para Regressão

- Algoritmo pode ser usado para estimar valores de funções contínuas

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k f(x_i)}{k}$$

- Predição é a *média simples* dos valores alvo armazenados nas instâncias recuperadas

Algoritmo k-NN para Regressão

- Regressão com Ponderação pela Distância

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k w_i f(x_i)}{\sum_{i=1}^k w_i}$$

- Predição é a *média ponderada* dos valores alvo armazenados nas instâncias recuperadas

Algoritmo k-NN

- Discussão

- K-NN é um *método lazy*
 - I.e., não gera um modelo durante o treinamento
- Métodos eager*, como as árvores de decisão, geram modelos de dados
- Consequências para o k-NN:
 - Treinamento rápido
 - Resposta lenta durante uso

Algoritmo k-NN

- Discussão

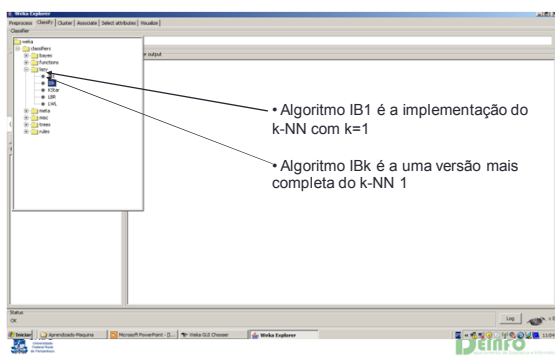
- Vantagens
 - É capaz de gerar boas respostas mesmo com poucos exemplos de treinamento
 - Algoritmos, como árvores de decisão, precisam de mais dados para gerar um bom modelo
 - Fácil de implementar

Algoritmo k-NN

- Discussão

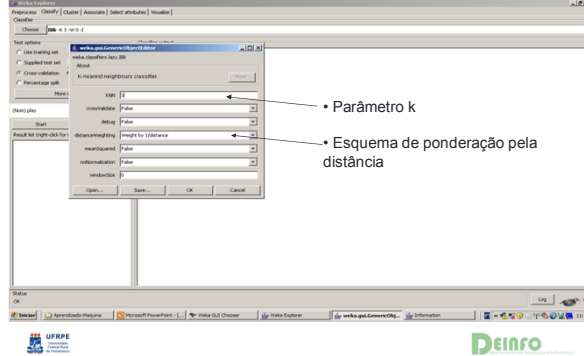
- Desvantagens
 - É muito sensível a presença de atributos irrelevantes e/ou redundantes
 - Curse of Dimensionality*
 - Tempo de resposta em alguns contextos é impraticável
 - Reduzir o número de exemplos de treinamento pode amenizar esse problema
 - Algoritmos baseados em protótipos* também podem ajudar

Algoritmo k-NN no WEKA



- Algoritmo IB1 é a implementação do k-NN com k=1
- Algoritmo IBk é a uma versão mais completa do k-NN 1

Algoritmo k-NN no WEKA



- Parâmetro k
- Esquema de ponderação pela distância

Referências

- T. Mitchell, 1997. *Machine Learning*.
- I. Witten, E. Frank, 2000. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*.
- D. Aha, D. Kibler, M. Albert, ,1991. Instance-based learning algorithms. *Machine Learning*, 6:37--66.