Capítulo 5

Classificação e regressão

Neste capítulo serão apresentados os principais conceitos relacionados a AM preditivo, que utilizam dados rotulados para a indução de modelos de classificação ou de regressão.

O presente capítulo está organizado conforme segue. Na Seção 5.2 são apresentados métodos de aprendizado supervisionado baseados em distâncias entre objetos. Na Seção 5.3 são propostos exercícios para fixar os conceitos apresentados.

5.1. Aprendizado de máquina supervisionado

Em AM supervisionado, algoritmos são utilizados para induzir modelos preditivos por meio da observação de um conjunto de objetos rotulados [Von Luxburg e Schölkopf 2008], tipicamente referenciado como conjunto de treinamento. Os rótulos contidos em tal conjunto correspondem a classes ou valores obtidos por alguma função desconhecida. Desse modo, um algoritmo de classificação buscará produzir um classificador capaz de generalizar as informações contidas no conjunto de treinamento, com a finalidade de classificar, posteriormente, objetos cujo rótulo seja desconhecido.

Formalmente, um conjunto de dados de treinamento pode ser definido como uma coleção de tuplas $\{(x_i, y_i)\}_{i=1}^{n}$, onde, em cada tupla, x_i indica um objeto descrito por m características e y_i indica o rótulo correspondente a x_i. Quando os valores de y_i são definidos por uma quantidade limitada de valores discretos, tem-se um problema de *classificação*. Quando tais valores são contínuos, tem-se um problema de *regressão*.

5.2. Métodos baseados em distâncias entre objetos

Métodos baseados em distâncias adotam a ideia de que objetos que pertencem a uma mesma classe possuem uma relação de proximidade no espaço de atributos considerados [Faceli et al. 2011].

5.2.1. *k-Nearest Neighbors* (kNN)

O algoritmo kNN é um dos algoritmos de classificação mais conhecidos e fáceis de se implementar na literatura de aprendizado de máquina e mineração de dados. Sua ideia consiste em, dado um objeto desconhecido, procurar pelos k vizinhos mais próximos a ele em um conjunto de dados previamente conhecido, segundo uma medida de distância pré-estabelecida. A classe do novo objeto será assumida como o voto majoritário entre os

seus k vizinhos.

Devido à sua simplicidade, o kNN possui ampla utilização na solução de diversos problemas do mundo real. Os pseudo-códigos para treinamento e teste do kNN são apresentados, respectivamente, nos Algoritmos 1 e 2.

Algoritmo 1: *Treinamento kNN.*

Entrada: conjunto de treinamento $T = \{x_i, y_i\}_{i=1}^{n}$, valor de k e uma medida de distância $d(\cdot,\cdot)$
Saída : classificador kNN

1 armazenar o conjunto de treinamento e o valor de k;

Algoritmo 2: *Teste kNN.*

Entrada: classificador kNN e um objeto x cuja classe é desconhecida
Saída : classe y atribuída a x

1 buscar pelos k objetos mais próximos a x no conjunto de dados de treinamento do classificador kNN informado;
2 dentre os k vizinhos, determinar y como a classe mais frequente entre eles, resolvendo possíveis empates de maneira arbitrária;

Em Python, pode-se executar o kNN conforme o código que pode ser visto a seguir.

```
import pandas
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

# carrega a base de dados iris
dados = pandas.read_csv('iris.csv')

# para criar um classificador, precisaremos separar
# o DataFrame original em dois numpy.arrays:
# - O primeiro deles, bidimensional, contendo os objetos
#   e os atributos;
# - O segundo deles, unidimensional, contendo apenas as
#   classes representadas por valores inteiros;

# Para obter os objetos e seus atributos, procede-se
# conforme abaixo
colunas = dados.columns.drop('Name')

# obtem numpy.array bidimensional
X = dados[colunas].values

# Para obter as classes como inteiros, utilizamos
# a classe LabelEncoder da scikit-learn
```

```
le = LabelEncoder()
y = le.fit_transform(dados['Name'])

# train_test_split separa o conjunto de dados original
# aleatoriamente em treinamento e teste
# train_size indica a proporcao de objetos presentes
# no conjunto de treinamento (neste caso 70% dos objetos)
# caso deseje-se uma separacao estratificada, deve-se
# informar um parametro adicional stratify=y
X_treino, X_teste, y_treino, y_teste = \
    train_test_split(X, y, train_size=0.7, test_size=0.3)

# instancia um knn
# n_neighbors indica a quantidade de vizinhos
# metric indica a medida de distancia utilizada
knn = KNeighborsClassifier(n_neighbors=5,
                           metric='euclidean')

# treina o knn
knn.fit(X_treino, y_treino)

# testa o knn com X_teste
# y_pred consistira em um numpy.array onde
# cada posicao contem a classe predita pelo knn
# para o respectivo objeto em X_teste
y_pred = knn.predict(X_teste)
```

Exercício 5.2.1. Discuta sobre os possíveis problemas que podem ocorrer com o kNN nos cenários a seguir:

a Quando a escala e intervalo de valores entre os atributos é diferente. Por exemplo, ao considerar atributos que indicam alguma medição em centímetros ou em metros.

b Quando a dimensionalidade dos objetos é alta[1]. Por exemplo, o que pode ocorrer ao buscar pelos k vizinhos mais próximos quando a base de dados possui 50, 100, 200, 300 ou mais atributos?

5.3. Exercícios

1. O kNN pode também ser utilizado para problemas de regressão local [Mitchell 1997, Faceli et al. 2011]. Com a scikit-learn esse tipo de tarefa pode ser realizada por meio da classe `KNeighborsRegressor`[2]. Escreva um código que treine um classificador kNN para regressão considerando os requisitos a seguir:

[1]Este é um problema, em aprendizado de máquina, conhecido como "maldição da dimensionalidade"

[2]`http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html`

- Para o treinamento utilize o arquivo `regressao_treino.csv`.
- Para o teste, utilize o arquivo `regressao_teste.csv`.
- Como medida de distância, considere a distância euclidiana.
- Teste o kNN com ponderação uniforme entre seus vizinhos (ou seja, considerando `weights='uniform'`) e pelo inverso das distâncias (isto é, utilizando `weights='distance'`).
- Teste os seguintes valores para k: 3, 5, 10, 20.

Gere um *scatter plot* para os dados de treinamento utilizando como eixo horizontal os valores dos exemplos e como eixo vertical os rótulos. Gere um *line plot*[3] para cada resultado das possíveis combinações entre ponderação e valor de k. Responda:

a Qual das duas ponderações parece ser mais adequada?

b Note que existem alguns pontos ruidosos no treinamento. Qual tipo de ponderação é mais influenciado por eles? Por que?

c Ao analisar os gráficos obtidos, responda qual função matemática gerou os dados.

Referências

[Faceli et al. 2011] Faceli, K., Lorena, A. C., Gama, J., e Carvalho, A. C. P. L. F. (2011). Inteligência artificial: Uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, 2:192.

[Mitchell 1997] Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877.

[Von Luxburg e Schölkopf 2008] Von Luxburg, U. e Schölkopf, B. (2008). Statistical learning theory: models, concepts, and results. *arXiv preprint arXiv:0810.4752*.

[3] `https://matplotlib.org/users/pyplot_tutorial.html`