Universidade Federal do Agreste de Pernambuco
Bacharelado em Ciências da Computação

Disciplina: Aprendizagem de Máquina.
Professor: Tiago B. A. de Carvalho.

Exercícios da Semana 02 - k-NN

Aluno(a): Maria Eduarda Deodato Interaminense

1. (20 pontos) Descreva um problema de classificação para o qual seria adequado utilizar o k-NN e descreva um problema de classificação para o qual não seria adequado utilizar este classi cador. Justifique suas escolhas baseado nas vantagens e desvantagens do k-NN. Mostre pelo menos duas vantagens e duas desvantagens para cada exemplo.

R: Um problema de classificação no reconhecimento de padrões em imagens médicas em que o k-NN não seria adequado é a identificação de diferentes tipos de câncer em imagens histopatológicas. Nesse caso, o k-NN apresentaria as seguintes desvantagens:

Alta dimensionalidade: Imagens histopatológicas contêm milhões de pixels, resultando em uma alta dimensionalidade do espaço de características. O k-NN sofre com a "maldição da dimensionalidade", onde a distância entre as amostras se torna menos discriminativa em espaços de alta dimensão. Isso pode levar a classificações imprecisas e afetar negativamente o desempenho do classificador.

Custo computacional: O k-NN requer o cálculo da distância entre a amostra de teste e todas as amostras de treinamento durante a fase de classificação. Em imagens histopatológicas, que são grandes e complexas, o cálculo da distância para cada pixel em cada imagem de treinamento se torna computacionalmente caro e ineficiente.

Em relação às vantagens do k-NN, aqui estão duas que não seriam aplicáveis nesse exemplo:

Simplicidade: Embora o k-NN seja um algoritmo simples e fácil de implementar, a complexidade das imagens histopatológicas exige métodos mais sofisticados de classificação que possam extrair e capturar informações relevantes de forma mais eficiente.

Flexibilidade: Embora o k-NN seja flexível na capacidade de lidar com dados não lineares e complexos, as imagens histopatológicas têm uma estrutura complexa que requer algoritmos de classificação mais avançados capazes de explorar e modelar as relações espaciais e texturais presentes nessas imagens.

Portanto, devido à alta dimensionalidade, ao alto custo computacional e à necessidade de técnicas mais avançadas para lidar com a complexidade das imagens histopatológicas, o k-NN não seria um classificador adequado para esse problema de reconhecimento de padrões em imagens médicas.

2. (35 pontos) Utilizando a base de dados archive.ics.uci.edu/ml/datasets/iris:

(a) Selecione os três exemplos aleatórios de cada classe e construa a matriz de distância entre colocando um exemplo de cada classe como elemento de conjunto de teste e os outros 6 como conjunto de treinamento.

O exercício foi realizado no google sheet -> iris

Treinamento				
sepal width	**sepal length**	**petal width**	**petal length**	**class**
4,9	3,1	1,5	0,1	Iris-setosa
5,1	2,5	3,0	1,1	Iris-versicolor
6,9	3,1	5,4	2,1	Iris-virginica
6,8	3,2	5,9	2,3	Iris-virginica
5,1	3,4	1,5	0,2	Iris-setosa
6,2	2,9	4,3	1,3	Iris-versicolor

Teste				
sepal width	**sepal length**	**petal width**	**petal length**	**class**
5,0	3,4	1,5	0,2	Iris-setosa
4,9	2,4	3,3	1,0	Iris-versicolor
6,0	2,2	5,0	1,5	Iris-virginica

Foi calculada a distância entre os exemplos do conjunto de teste e do conjunto de treinamento.

	Exemplo 1	**Exemplo 2**	**Exemplo 3**	**Exemplo 4**	**Exemplo 5**	**Exemplo 6**
Exemplo X	0,3	2,0	4,7	5,2	0,1	3,3
Exemplo Y	2,1	0,4	3,2	3,6	2,2	1,7
Exemplo Z	4,0	2,2	1,5	1,8	4,0	1,0

(b) Utilizando a matriz de distância explique a classificação dos exemplos de teste utilizando 1-NN.

Com a matriz de distância calculada, encontramos o exemplo mais próximo (1-NN) no conjunto de treinamento para cada exemplo do conjunto de testes. Foi atribuída a classe do exemplo mais próximo como a classe prevista para o exemplo de teste correspondente.

Treinamento							
	Exemplo 1	**Exemplo 2**	**Exemplo 3**	**Exemplo 4**	**Exemplo 5**	**Exemplo 6**	**class**
Exemplo X	0,3	2,0	4,7	5,2	0,1	3,3	iris-setosa
Exemplo Y	2,1	0,4	3,2	3,6	2,2	1,7	Iris-versicolor
Exemplo Z	4,0	2,2	1,5	1,8	4,0	1,0	Iris-versicolor

(c) Utilizando a matriz de distância explique a classificação dos exemplos de teste utilizando 3-NN sem peso.

Com a matriz de distância calculada, encontramos os três exemplos mais próximos (3-NN) no conjunto de treinamento para cada exemplo do conjunto de testes. Com isso, foi atribuída a classe que aparece com mais frequência nos três exemplos mais próximos como a classe prevista para o exemplo de teste correspondente.

A fórmula utilizada para calcular a distância foi a Euclidiana -> $d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d}(x_{ki} - x_{kj})^2},$

A fórmula utilizada na planilha

ex: =RAIZ((B3-H3)^2+(C3-I3)^2+(D3-J3)^2+(E3-K3)^2)

Treinamento							
	Exemplo 1	**Exemplo 2**	**Exemplo 3**	**Exemplo 4**	**Exemplo 5**	**Exemplo 6**	**class**
Exemplo X	0,3	2,0	4,7	5,2	0,1	3,3	Iris-setosa
Exemplo Y	2,1	0,4	3,2	3,6	2,2	1,7	Iris-versicolor
Exemplo Z	4,0	2,2	1,5	1,8	4,0	1,0	Iris-virginica

(d) Utilizando a matriz de distância explique a classificação dos exemplos de teste utilizando 3-NN com peso.

Com a matriz de distância calculada, e com os três exemplos mais próximos (3-NN) no conjunto de treinamento para cada exemplo do conjunto de testes.

É atribuída a classe do exemplo mais próximo com base em uma ponderação inversamente proporcional à distância. Quanto mais próximo o exemplo, maior será o peso atribuído à sua classe na votação final.

A fórmula utilizada para calcular a distância dos exemplos usando o 3NN - > $1/d(\mathbf{x}_i, \mathbf{x}_t)$

A fórmula utilizada na planilha

ex: - > =SOMA(1/0,3)+(1/0,1)

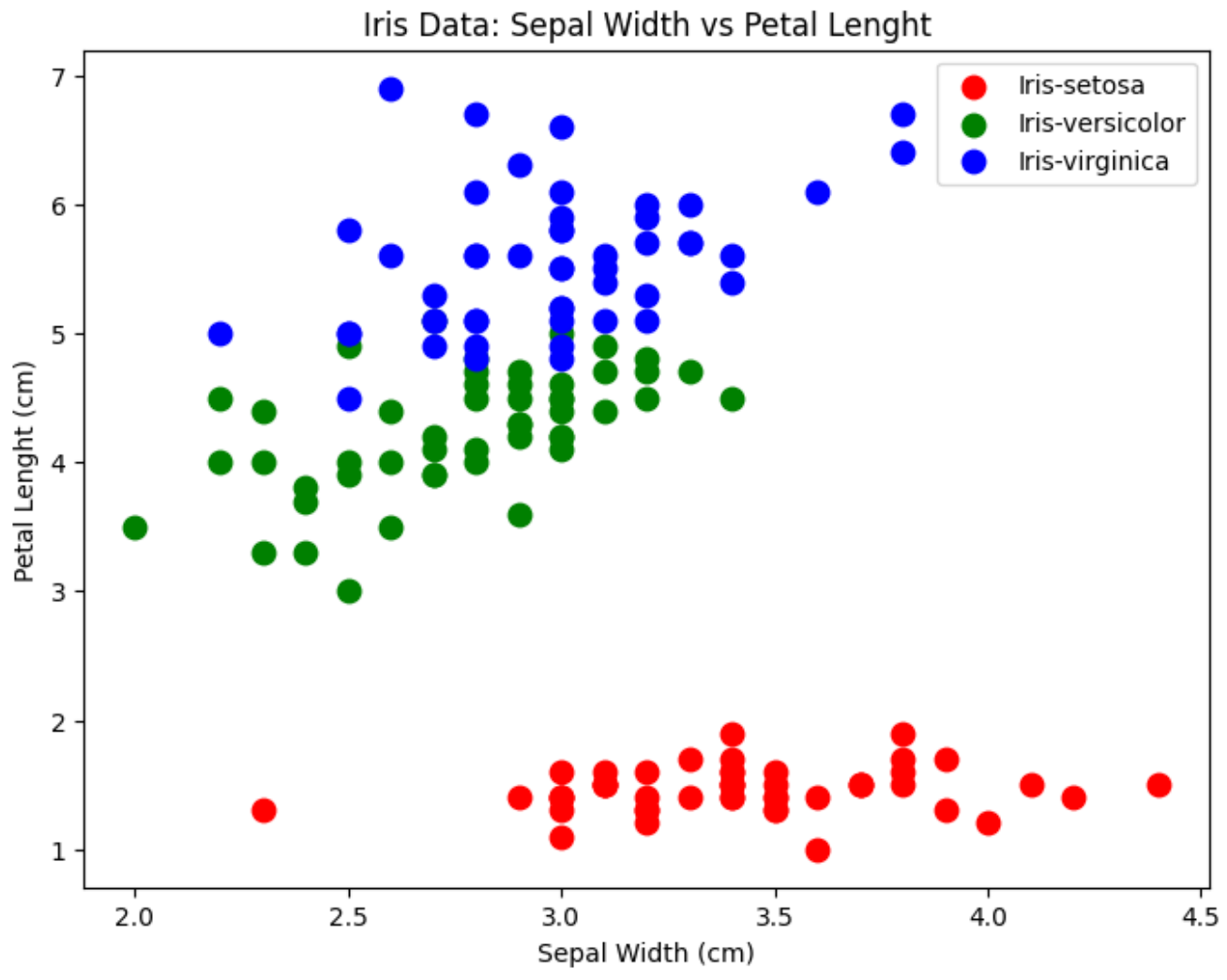
	Exemplo X	**Exemplo Y**	**Exemplo Z**
Iris-setosa	13,33333333	0,4761904762	0
Iris-versicolor	0,5	3,088235294	1
Iris-virginica	0	0	1,222222222

Os marcados são as classificações de maior peso.

(e) Selecione duas características da base

Iris construa um diagrama de dispersão colocando símbolos ou cores distintas para cada classe.

```python
# Implementando as bibliotecas necessárias
import pandas as pd
import matplotlib.pyplot as plt

# Define as colunas do conjunto de dados
name = ['sepal lenght','sepal width','petal lenght','petal width', 'class']

# Ler o conjunto de dados
data = pd.read_csv('iris.data', header=None, names = name)

# Definição das colunas selap width e petal lenght
sepal_width = data['sepal width']  # sepal width
petal_lenght = data['petal lenght']  # petal lenght

classe = data['class']

# Criar gráfico de dispersão
# Definindo tamanho do gráfico
plt.figure(figsize=(8, 6))

# Personalização do gráfico
plt.scatter(sepal_width[classe == 'Iris-setosa'], petal_lenght[classe == 'Iris-setosa'], c='red', marker='o', s=80, label = 'Iris-setosa')
plt.scatter(sepal_width[classe == 'Iris-versicolor'], petal_lenght[classe == 'Iris-versicolor'], c='green', marker='o', s=80, label = 'Iris-versicolor')
plt.scatter(sepal_width[classe == 'Iris-virginica'], petal_lenght[classe == 'Iris-virginica'], c='blue', marker='o', s=80, label= 'Iris-virginica')

# Inserindo legenda no gráfico
plt.xlabel('Sepal Width (cm)')
plt.ylabel('Petal Lenght (cm)')
plt.title('Iris Data: Sepal Width vs Petal Lenght')

# Adicionando legenda
plt.legend()

# Exibe o gráfico
plt.show()
```



Dica: esta questão pode ser resolvida inteiramente no Excel ou no LibreOffice Calc.

3. (10 pontos) Utilize o classi cador pelo vizinho mais próximo (1-NN) com distância euclidiana. Avalie este classi cador fazendo metade dos exemplos de cada classe da base Iris como conjunto de teste e o restante como conjunto de treinamento. Calcule o número acertos no conjunto de testes. Dica: você pode utilizar o sklearn scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

```python
# Importação das bibliotecas necessárias
# 'pandas' usada para manipular e análisar dados
import pandas as pd
# Implementa o algoritmo do KNN
from sklearn.neighbors import KNeighborsClassifier

# Define as colunas do conjunto de dados
name = ['sepal lenght','sepal width','petal lenght','petal width', 'class']

# Ler o conjunto de dados de treinamento e teste
data_treinamento = pd.read_csv('iris.treinamento.txt', header=None, names= name)
data_teste = pd.read_csv('iris.test.txt', header=None, names= name)

# Imprime o conjunto de dados de treinamento e teste
print(data_treinamento)
print(data_teste)

# Instanciação dos classificadores KNN, com o 1 vizinho mais próximo
knn = KNeighborsClassifier(n_neighbors=1)

# Previsão dos conjuntos
knn.fit(data_treinamento[name[:-1]], data_treinamento['class'])
knn.predict(data_teste[name[:-1]])

# Calcular taxas de acerto
score = knn.score(data_teste[name[:-1]], data_teste['class'])

# Imprime taxas de acerto
print('Taxa de acerto: ',score)
```

```
    sepal lenght  sepal width  petal lenght  petal width          class
0            5.0          3.0           1.6          0.2    Iris-setosa
1            5.0          3.4           1.6          0.4    Iris-setosa
2            5.2          3.5           1.5          0.2    Iris-setosa
3            5.2          3.4           1.4          0.2    Iris-setosa
4            4.7          3.2           1.6          0.2    Iris-setosa
..           ...          ...           ...          ...            ...
68           6.7          3.0           5.2          2.3  Iris-virginica
69           6.3          2.5           5.0          1.9  Iris-virginica
70           6.5          3.0           5.2          2.0  Iris-virginica
71           6.2          3.4           5.4          2.3  Iris-virginica
72           5.9          3.0           5.1          1.8  Iris-virginica

[73 rows x 5 columns]
    sepal lenght  sepal width  petal lenght  petal width          class
0            5.1          3.5           1.4          0.2    Iris-setosa
1            4.9          3.0           1.4          0.2    Iris-setosa
2            4.7          3.2           1.3          0.2    Iris-setosa
3            4.6          3.1           1.5          0.2    Iris-setosa
4            5.0          3.6           1.4          0.2    Iris-setosa
..           ...          ...           ...          ...            ...
72           5.6          2.8           4.9          2.0  Iris-virginica
73           7.7          2.8           6.7          2.0  Iris-virginica
74           6.3          2.7           4.9          1.8  Iris-virginica
75           6.7          3.3           5.7          2.1  Iris-virginica
76           7.2          3.2           6.0          1.8  Iris-virginica

[77 rows x 5 columns]
Taxa de acerto:  0.948051948051948
```

4. (15 pontos) Utilize os classificadores 7-NN com e 7-NN sem peso e avalie os classi cadores utilizando metade dos exemplos de cada classe da base Speaker Accent Recognition como conjunto de teste e a outra metade como conjunto de treinamento. Base: archive.ics.uci. edu/ml/datasets/Wine, arquivo accent-mfcc-data-1.csv. Dica: você pode utilizar o skle arn archive.ics.uci.edu/ml/datasets/Speaker+Accent+Recognition.

```python
# Importação das bibliotecas necessárias
# 'pandas' usada para manipular e análisar dados
import pandas as pd
# Implementa o algoritmo do KNN
from sklearn.neighbors import KNeighborsClassifier
# Divide os dados em conjuntos de treinamento e teste
from sklearn.model_selection import train_test_split


name = ['class', 'X1', 'X2' , 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9', 'X10',
        'X11', 'X12']

# Ler os dados
data = pd.read_csv('accent-mfcc-data-1.csv', header=0, names= name)

# Divide os dados em conjuntos de treinamento e teste
dataX_train, dataX_test, datay_train, datay_test = train_test_split(data[name[1:]],
data['class'], test_size=0.5)

# Imprime os dados dos conjuntos de treinamento e teste
print(dataX_train)
print(dataX_test)

# Instanciação dos classificadores KNN, com os 7 vizinhos mais próximos
knn = KNeighborsClassifier(n_neighbors=7)
# Instanciação dos classificadores KNN, indica que os vizinhos mais próximos
# terão pesos baseados na distância
knn_weight = KNeighborsClassifier(n_neighbors=7, weights = 'distance')
```

```python

# Treinamento dos classificadores
knn.fit(dataX_train,  datay_train)
knn_weight.fit(dataX_train,  datay_train)

# Previsão dos conjuntos
knn.predict(dataX_test)
knn_weight.predict(dataX_test)

# Calcular taxas de acerto
score = knn.score(dataX_test, datay_test)
score_weights = knn_weight.score(dataX_test, datay_test)

# Imprime as taxas de acerto
print('7NN sem peso | Taxa de acerto: ', score)
print('7NN com peso | Taxa de acerto: ', score_weights)
```

```
7NN sem peso | Taxa de acerto:  0.6606060606060606
7NN com peso | Taxa de acerto:  0.703030303030303
```

5. (10 pontos) Faça o mesmo da questão anterior para a base Wine archive.ics.uci.edu/ml/datasets/Wine.

```
# Importação das bibliotecas necessárias
# 'pandas' usada para manipular e análisar dados
import pandas as pd
# Implementa o algoritmo do KNN
from sklearn.neighbors import KNeighborsClassifier

# Define as colunas do conjunto de dados
name = ['class','Alcohol','Malic acid','Ash','Alcalinity of ash', ' Magnesium',
        ' Total phenols', 'Flavanoids', 'Nonflavanoid phenols','Proanthocyanins',
        'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline']

# Ler o conjunto de dados de treinamento e teste
data_treinamento = pd.read_csv('wine.data', header=None, names= name)
data_teste = pd.read_csv('wine.teste', header=None, names= name)

# Imprime o conjunto de dados de treinamento e teste
print(data_treinamento)
print(data_teste)

# Instanciação dos classificadores KNN, com os 7 vizinhos mais próximos
knn = KNeighborsClassifier(n_neighbors=7)
# Instanciação dos classificadores KNN, indica que os vizinhos mais próximos
# terão pesos baseados na distância
knn_weight = KNeighborsClassifier(n_neighbors=7, weights = 'distance')

# Treinamento dos classificadores
knn.fit(data_treinamento[name[1:]], data_treinamento['class'])
knn_weight.fit(data_treinamento[name[1:]], data_treinamento['class'])
```

```python
# Previsão dos conjuntos
knn.predict(data_teste[name[1:]])
knn_weight.predict(data_teste[name[1:]])

# Calcular taxas de acerto
score = knn.score(data_teste[name[1:]], data_teste['class'])
score_weights = knn_weight.score(data_teste[name[1:]], data_teste['class'])

# Imprime as taxas de acerto
print('7NN sem peso | Taxa de acerto: ', score)
print('7NN com peso | Taxa de acerto: ', score_weights)
```

```
7NN sem peso | Taxa de acerto:  0.9213483146067416
7NN com peso | Taxa de acerto:  1.0
```

6. (10 pontos) Faça o mesmo da questão anterior removendo a última coluna da base Wine

```python
# Importação das bibliotecas necessárias
# 'pandas' usada para manipular e análisar dados
import pandas as pd
# Implementa o algoritmo do KNN
from sklearn.neighbors import KNeighborsClassifier

# Define as colunas do conjunto de dados
name = ['class','Alcohol','Malic acid','Ash','Alcalinity of ash', ' Magnesium',
        ' Total phenols', 'Flavanoids', 'Nonflavanoid phenols','Proanthocyanins',
        'Color intensity', 'Hue', 'OD280/OD315 of diluted wines', 'Proline']

# Ler o conjunto de dados de treinamento e teste
data_treinamento = pd.read_csv('wine.data', header=None, names= name)
data_teste = pd.read_csv('wine.teste', header=None, names= name)

# Imprime o conjunto de dados de treinamento e teste
print(data_treinamento)
print(data_teste)

# Instanciação dos classificadores KNN, com os 7 vizinhos mais próximos
knn = KNeighborsClassifier(n_neighbors=7)
# Instanciação dos classificadores KNN, indica que os vizinhos mais próximos
# terão pesos baseados na distância
knn_weight = KNeighborsClassifier(n_neighbors=7, weights = 'distance')

# Treinamento dos classificadores
knn.fit(data_treinamento[name[1:-1]], data_treinamento['class'])
knn_weight.fit(data_treinamento[name[1:-1]], data_treinamento['class'])
```

```python
# Previsão dos conjuntos removendo a ultima coluna
knn.predict(data_teste[name[1:-1]])
knn_weight.predict(data_teste[name[1:-1]])

# Calcular taxas de acerto
score = knn.score(data_teste[name[1:-1]], data_teste['class'])
score_weights = knn_weight.score(data_teste[name[1:-1]], data_teste['class'])

# Imprime as taxas de acerto
print('7NN sem peso | Taxa de acerto: ', score)
print('7NN com peso | Taxa de acerto: ', score_weights)
```

```
[89 rows x 14 columns]
7NN sem peso | Taxa de acerto:  0.8651685393258427
7NN com peso | Taxa de acerto:  1.0
```