Lab Solutions

k-NN Nearest Neighbour Prediction

Q1

- Three examples are shown below from the Iris dataset:
  - Each example represented by 4 numeric features.
  - Example *x1*: Class A
  - Example *x2*: Class B

Example: *x1*	
Sepal length	4.4
Sepal width	2.9
Petal length	1.4
Petal width	0.2
Class	A

Example: *x2*	
Sepal length	5.6
Sepal width	3.0
Petal length	4.5
Petal width	1.5
Class	B

Query: *q*	
Sepal length	6.1
Sepal width	3.0
Petal length	4.6
Petal width	1.4
Class	???

a. What type of distance function might be appropriate for comparing the examples above?

b. Use this distance function to calculate the distances between the query example *q* and the labelled examples. Which class label would a 1-NN classifier assign to the query based on the distances?

Q1

- Euclidean distance: Calculate square root of sum of squared differences for each feature *f* representing a pair of examples.

$$\mathrm{ED}(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{f \in F} (q_f - p_f)^2}$$

ED(q, x1)

$$\sqrt{(6.10 - 4.40)^2 + (3.00 - 2.90)^2 + (4.60 - 1.40)^2 + (1.40 - 0.20)^2} = 3.82$$

ED(q, x2)

$$\sqrt{(6.10 - 5.60)^2 + (3.00 - 3.00)^2 + (4.60 - 4.50)^2 + (1.40 - 1.50)^2} = 0.52$$

- Distance to example *x2* is smaller

➡ Assign query to Class "B"

Q2

- Three examples from a system for predicting whether a person is over or under the drink driving limit.
  - Gender, Weight, Amount of alcohol in units, Meal type, Duration of drinking session.

Example: *x1*

Gender	female
Weight	60
Amount	4
Meal	full
Duration	90
Class	over

Example: *x2*

Gender	male
Weight	75
Amount	2
Meal	full
Duration	60
Class	under

Query: *q*

Gender	male
Weight	70
Amount	1
Meal	snack
Duration	30
Class	???

a. Normalise all numeric features to the range [0,1].

b. Propose an appropriate global distance function for comparing examples such as the above.

c. Use your proposed distance function to calculate the distances between the query example and the two labelled examples. Which class label would a 1NN classifier assign to the query based on the distances?

Q2a

a. Normalise all numeric features to the range [0,1]

- Min-max normalisation:
  Use min and max values for a given feature to rescale to the range [0,1]

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

  - Weight: numeric, range [50,150]
  - Amount: numeric, range [1,16]
  - Duration: numeric, range [20,230]

Example: x1

Weight	(60-50)/(150-50) = 0.1
Amount	(4-1)/(16-1) = 0.2
Duration	(90-20)/(230-20) = 0.333

Example: x2

Weight	(75-50)/(150-50) = 0.25
Amount	(2-1)/(16-1) = 0.067
Duration	(60-20)/(230-20) = 0.19

Query: q

Weight	(70-50)/(150-50) = 0.2
Amount	(1-1)/(16-1) = 0
Duration	(30-20)/(230-20) = 0.048

Q2b

b. Propose an appropriate global distance function for comparing the examples.

Ordinal features: the distance can be the absolute difference between the two positions in the ordinal list of possible values.

- Meal: {None, Snack, Lunch, Full} = {1, 2, 3, 4}

e.g. `d(Snack,Full) = |2-4| = 2`

In practice, we often normalise with respect to ordinal list length *n*. (Note: we can also normalise with respect to the range *n-1*).

e.g. `|2-4|/4 = 0.5`
or `|2-4|/3 = 0.66`

Feature	Type	Local Distance Function
Gender	Categorical	Overlap function
Weight	Numeric	Absolute difference (after normalisation)
Amount	Numeric	Absolute difference (after normalisation)
Meal	Ordinal {None, Snack, Lunch, Full}	Absolute relative rank difference (norm)
Duration	Numeric	Absolute difference (after normalisation)

Q2c

c. Use your proposed distance function to calculate the distances between the query example and the two labelled examples.

Sum over local distance on each feature:
Gender + *Weight* + *Amount* + *Meal* + *Duration*

D(x1,q)

Feature	Difference
Gender	1
Weight	|0.1-0.2| = 0.1
Amount	|0.2-0| = 0.2
Meal	|2-4|/4 = 0.5
Duration	|0.333-0.048| = 0.285

D(x1,q)
= 1 + 0.1 + 0.2 + 0.5 + 0.285
= 2.085

D(x2,q)

Feature	Difference
Gender	0
Weight	|0.25-0.2| = 0.05
Amount	|0.067-0| = 0.067
Meal	|2-4|/4 = 0.5
Duration	|0.19-0.048| = 0.142

D(x2,q) = 0 + 0.05 + 0.067 +
0.5 + 0.142
= 0.759

➡ Label *q* with same class as *x2* (i.e. "under")

Q3a

- Pairwise distances between 9 labelled training examples and a new query example **q**, for the system described in Question 2.

a. What class would a 3-NN classifier assign to **q**?

Example	Class	Distance to q
x1	over	1.5
x2	under	2.8
x3	over	1.8
x4	under	2.9
x5	under	2.2
x6	under	3.0
x7	under	2.4
x8	over	3.2
x9	over	3.6

Example	Class	Distance to q
x1	over	1.5
x3	over	1.8
x5	under	2.2
x7	under	2.4
x2	under	2.8
x4	under	2.9
x6	under	3.0
x8	over	3.2
x9	over	3.6

Sort by distance, smallest first

- Over = 2 votes
- Under = 1 vote

➡ Label **q** as 'over'

Q3b

- Pairwise distances between 9 labelled training examples and a new query example **q**, for the system described in Question 2.

b. What class would a 4-NN classifier assign to **q**?

Example	Class	Distance to q
x1	over	1.5
x2	under	2.8
x3	over	1.8
x4	under	2.9
x5	under	2.2
x6	under	3.0
x7	under	2.4
x8	over	3.2
x9	over	3.6

Example	Class	Distance to q
x1	over	1.5
x3	over	1.8
x5	under	2.2
x7	under	2.4
x2	under	2.8
x4	under	2.9
x6	under	3.0
x8	over	3.2
x9	over	3.6

Sort by distance, smallest first

- Over = 2 votes
- Under = 2 votes

➡ Tie!

Note top-ranked examples are both 'over'

Q3c

- Pairwise distances between 9 labelled training examples and a new query example **q**, for the system described in Question 2.

c. What class would a weighted 4-NN classifier assign to **q**?

Example	Class	Distance to q	Weight
x1	over	1.5	1/1.5 = 0.666
x3	over	1.8	1/1.8 = 0.555
x5	under	2.2	1/2.2 = 0.454
x7	under	2.4	1/2.4 = 0.417
x2	under	2.8	...
x4	under	2.9	...
x6	under	3.0	...
x8	over	3.2	...
x9	over	3.6	...

- Over = 0.666 + 0.555 = 1.221
- Under = 0.454 + 0.417 = 0.871

➡ Label **q** as 'over'

Sort by distance, smallest first.
Calculate weight as inverse distance.

Q4

- Two examples from a Case-based reasoning (CBR) system for estimating the price of second-hand cars are described by 6 features:

Example: ***x1***

Manufacturer	Ford
Model	Fiesta
Engine Size	1,100
Fuel	Petrol
Mileage	65,000
Condition	Excellent
Price	€3,100

Example: ***x2***

Manufacturer	Citroen
Model	BX
Engine Size	1,800
Fuel	Diesel
Mileage	37,000
Condition	Fair
Price	€4,500

a. Normalise all numeric features to the range [0,1]. Assume that the feature ranges are: Engine Size 1,000 to 3,000; Mileage 1,000 to 100,000.

b. Propose a suitable global distance function. Assume that Condition is an ordinal feature that has the possible values {Poor, Fair, Good, Excellent},

c. Use this measure to calculate the distance between *x1* and *x2*.

Q4a

a. Normalise all numeric features to the range [0,1]. Note that you can assume that the feature ranges for: Engine Size is 1,000 to 3,000; Mileage is 1,000 to 100,000.

- Min-max normalisation:
  Use min and max values for a given feature to rescale to the range [0,1]

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Example: *x1*

Manufacturer	Ford
Model	Fiesta
Engine Size	**(1100-1000)/ (3000-1000) = 0.05**
Fuel	Petrol
Mileage	**(65000-1000)/ (100000-1000) = 0.646**
Condition	Excellent

Example: *x2*

Manufacturer	Citroen
Model	BX
Engine Size	**(1800-1000)/ (3000-1000) = 0.4**
Fuel	Diesel
Mileage	**(37000-1000)/ (100000-1000) = 0.364**
Condition	Fair

Q4b

Feature	Type	Local Distance Function
Manufacturer	Categorical	Overlap function
Model	Categorical	Overlap function
Engine Size	Numeric	Absolute difference (after normalisation)
Fuel	Categorical	Overlap function
Mileage	Numeric	Absolute difference (after normalisation)
Condition	Ordinal {Poor, Fair, Good, Excellent}	Absolute relative rank difference (normalised)

Sum over local distance on each feature:
Manufacturer + Model + Engine Size + Fuel + Mileage + Condition

Q4c

Example: *x1* (Normalised)

Manufacturer	Ford
Model	Fiesta
Engine Size	0.05
Fuel	Petrol
Mileage	0.646
Condition	Excellent

Example: *x2* (Normalised)

Manufacturer	Citroen
Model	BX
Engine Size	0.4
Fuel	Diesel
Mileage	0.364
Condition	Fair

Calculate *D(x1,x2)*

Feature	Difference
Manufacturer	1
Model	1
Engine Size	|0.05-0.4| = 0.35
Fuel	1
Mileage	|0.646-0.364| = 0.282
Condition	|4-2|/4 = 0.5

```
Dist = 1 + 1 + 0.35 + 1
+ 0.282 + 0.5 = 4.132
```

* subject to rounding

Q5

- Change the metric used by k-NN to correlation to see if it will predict the other class.

```
house_C_kNN = KNeighborsClassifier(n_neighbors=1, metric='correlation')
house_C_kNN.fit(X,y)
print('Query is classified as',house_C_kNN.predict([q])[0] )

Query is classified as C1
```

Q6

- In the Data Normalisation example in the 02-kNN Notebook replace the N(0,1) scaler with a min-max scaler.

```
athlete = pd.read_csv('AthleteSelection.csv',index_col = 'Athlete')
y = athlete.pop('Selected').values
X = athlete.values
names = athlete.index
q = [5.0,7.5]

In [20]:
from sklearn import preprocessing
mm_scaler = preprocessing.MinMaxScaler().fit(X)
X_scaled = mm_scaler.transform(X)
q_scaled = mm_scaler.transform([q])
q_scaled
```

Q6

- In the Data Normalisation example in the 02-kNN Notebook replace the N(0,1) scaler with a min-max scaler.

```
athlete = pd.read_csv('AthleteSelection.csv',index_col = 'Athlete')
y = athlete.pop('Selected').values
X = athlete.values
names = athlete.index
q = [5.0,7.5]

In [20]:
from sklearn import
mm_scaler = preproce
X_scaled = mm_scaler
q_scaled = mm_scaler
q_scaled
```