K-Nearest-Neighbours

K-Nearest-Neighbours (KNN) is a machine learning technique for predicting labels based on the assumption that things tend to be like things that are around. KNN is an elegant and very effective classification algorithm but is limited in its application in cases of dense, non-linearly separable data. The methodology is as follows; For each test point, the KNN assigns the test point a label which is the mode label of the K closest training data points. Because of this, K must always be odd so a clear winner can be determined.

A picture containing clock, object

Description automatically generatedA close up of a clock

Description automatically generated

Fig 1a. 1b.

Within the framework of the algorithm, K is a regularising constant, if K is small, a very heterogeneous spatial distribution of the classification space is produced. If K tends to infinite, a single “winner” label will be assigned to all test points (based on the most common label in the training set). For this task, I will choose to work with the reduced iris dataset (of only petal length and petal width), so as to improve comprehension and reduce the computational complexity.

![A close up of a map

Description automatically generated]()

Fig 2. Scatterplot of petal width vs petal length

With a 20% test 80%, and K=1, a prediction accuracy of 86% can be produced. Increasing to K=3, a 100% prediction accuracy is achieved. Using synthetic data, we can map the decision space of the KNN to get a visual representation of classification boundaries as shown below:

![A picture containing screenshot

Description automatically generated]()

Fig 1a. Classification of synthetic data for K=1 1b. Classification of synthetic data for K=5 1c. Classification of synthetic data for K=11

Whilst this method is very simple and doesn’t generate any meaningful insight other than “things tend to be like things that are like them”, it is very effective in specific circumstances and can be very useful in identifying over-segmentation of your data.

Code

Code 1:

﻿import numpy as np

from sklearn.datasets import load\_iris

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

data, classifier = load\_iris(return\_X\_y=True)

# for ease of visualisation, i reduce the dimenions to 2D

data = np.delete(data,0,1)

data = np.delete(data,0,1)

neigh = KNeighborsClassifier(n\_neighbors=1)

holdout = 0.5;

Xtrain, Xtest, Ytrain, Ytest = train\_test\_split(data, classifier, test\_size=holdout)

neigh.fit(Xtrain,Ytrain)

Ypred = neigh.predict(Xtest)

score = (len(Ytest)-sum(abs(Ytest-Ypred)))/len(Ytest)

Code2:

﻿import numpy as np

from sklearn.datasets import load\_iris

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

data, classifier = load\_iris(return\_X\_y=True)

# for ease of visualisation, I reduce the dimensions to 2D

data = np.delete(data,0,1)

data = np.delete(data,0,1)

neigh = KNeighborsClassifier(n\_neighbors=5)

neigh.fit(data,classifier)

# Explore the separatrix with synthetic data

synth = np.zeros([816,2])

synth\_classifier = np.zeros([816,1])

t0=0;

for k in range(1,69,2):

for l in range(1,25,1):

synth[t0,:] = [0.1\*k,0.1\*l]

synth\_classifier[t0] = neigh.predict([synth[t0,:]])

t0=t0+1;

# Generate the plot

species = ['Setosa','Versicolor','Virginica']

for i in range(0,3):

plt.scatter(synth[np.where(synth\_classifier==i),0],synth[np.where(synth\_classifier==i),1], label=species[i])

plt.legend()

plt.xlabel('Petal Length [cm]')

plt.ylabel('Petal Width [cm]')

plt.title('5-NN of the Iris dataset')

plt.show()