K-nearest neighbor implementation

Tiina Nokelainen

In [1]:
```
import pandas as pd
import numpy as np
import math
import operator
from sklearn.utils import shuffle
from collections import Counter
```

In [13]:
```
# Euclidean distance
# summarizes the distances to distance-variable and returns squareroot of the distance
def distance(trainX, predX):

    distance = 0

    for n in range(1,len(trainX)):
        distance += math.pow((trainX[n] - predX[n]), 2)

    return math.sqrt(distance)
```

In [12]:
```
# Split the dataset to training data and testing data
# split is the size of the training data
def splitData(dataSet, split, trainSet=[], testSet=[]):


    index = math.floor(len(dataSet)*split)

    for i in range(index):
        trainSet.append(dataSet[i])

    for i in range(index, len(dataSet)):
        testSet.append(dataSet[i])
```

In [16]:
```
# returns k number of nearest neighbors
def getNeighbors(trainSet, testInstance, k):

    distances = []

    for i in range(len(trainSet)):
        # adds all the distances to the distances-variable
        distances.append((trainSet[i][0], distance(trainSet[i], testInstance)))

    # sorts the distances by the distance-value (which is the second column)
    distances.sort(key=operator.itemgetter(1))

    neighbors = []

    # stores k nearest neighbors to neighbors
    for i in range(k):
        neighbors.append(distances[i][0])

    return neighbors
```

In [15]:
```
# returns the accuracy of predictions
def accuracy(predictions, testSet):

    correct = 0

    for i in range(len(testSet)):
        if predictions[i]==testSet[i][0]:
            correct += 1

    return (round(correct / len(testSet)*100, 2))
```

In [14]:
```
def kNN(data,split,times):

    testSet = []
    trainSet = []

    splitData(data, split, trainSet, testSet)

    # tests the data with k values from 1 to times
    for x in range(1,times+1):

        k=x

        predictions = []

        for i in range(len(testSet)):
            neighbors = getNeighbors(trainSet, testSet[i], k)
            label = Counter(neighbors).most_common()[0][0]
            predictions.append(label)


        a = accuracy(predictions, testSet)


        print("k =", k, ":" , a)
```

Cross-validation

In [7]:
```
def cv(k,data,n):

    # splits the data to n folds
    splitted = np.array_split(data,n)
    accur = []

    for i in range(n):

        testset = splitted[i]

        trainset = np.concatenate(np.delete(splitted, i,0), axis=0)

        preds = []

        for j in range(len(testset)):
            neighbors = getNeighbors(trainset, testset[j], k)
            label = Counter(neighbors).most_common()[0][0]
            preds.append(label)

        accur.append(accuracy(preds, testset))

    print("k =", k, "accuracy is", round(np.mean(accur),2))
```

In [ ]:
```
# data preparations

wineData = np.loadtxt('wine.data', dtype=np.float, delimiter=',')
np.random.shuffle(wineData)

irisdata = pd.read_csv('iris.data', header=None)
irisdata = shuffle(irisdata)
irisdata.reset_index(drop=True)

irisdata = irisdata.reset_index(drop=True)

irisdata = irisdata[[4,0,1,2,3]]

irisDataSet = irisdata.values
```

In [11]:
```
# main

#print("WINE DATA:")
#kNN(wineData, 0.7, 10)

#print("IRIS DATA:")
#kNN(irisDataSet, 0.66, 10)

print("WINE DATA with leave-one-out cross validation:")
for i in range(1,10):
    cv(i,wineData,len(wineData))


print("IRIS DATA with leave-one-out cross validation:")
for i in range(1,30):
    cv(i,irisDataSet,len(irisDataSet))
```

```
WINE DATA with leave-one-out cross validation:
k = 1 accuracy is 76.97
k = 2 accuracy is 76.97
k = 3 accuracy is 74.16
k = 4 accuracy is 73.03
k = 5 accuracy is 71.91
k = 6 accuracy is 71.35
k = 7 accuracy is 69.66
k = 8 accuracy is 74.16
k = 9 accuracy is 70.79
IRIS DATA with leave-one-out cross validation:
k = 1 accuracy is 96.0
k = 2 accuracy is 96.0
k = 3 accuracy is 96.0
k = 4 accuracy is 96.0
k = 5 accuracy is 96.67
k = 6 accuracy is 96.0
k = 7 accuracy is 96.67
k = 8 accuracy is 96.67
k = 9 accuracy is 96.67
k = 10 accuracy is 96.0
k = 11 accuracy is 97.33
k = 12 accuracy is 96.67
k = 13 accuracy is 96.67
k = 14 accuracy is 97.33
k = 15 accuracy is 97.33
k = 16 accuracy is 97.33
k = 17 accuracy is 97.33
k = 18 accuracy is 97.33
k = 19 accuracy is 98.0
k = 20 accuracy is 97.33
k = 21 accuracy is 98.0
k = 22 accuracy is 97.33
k = 23 accuracy is 96.67
k = 24 accuracy is 96.67
k = 25 accuracy is 96.67
k = 26 accuracy is 96.67
k = 27 accuracy is 96.67
k = 28 accuracy is 96.0
k = 29 accuracy is 95.33
```

Best k-value for iris.data is when k = 19 (accuracy 98 %) with leave-one-out cross validation (special cade of k-fold cv with k being n). This sounds kind of weird, I think there is something wrong with my CV or kNN or possibly both.