K-nearest neighbors and cross-validation

In the k-nearest neighbor algorithm, the training set is retained as part of the model. So, it is an extreme case instance-based algorithm.

The k-nearest algorithm is one of the simplest algorithms to implement. The parameters of this algorithm are the **training set**, the **test set** and **k** that represents the closest neighbors to the test instance.

The **training set** is used for creating the predictions and the **test set** is used for checking the accuracy of those predictions.

Notice that the k-nearest neighbor algorithm can be used in regression or classification problems. In regression problems, we trained to predict the real number, and in classification problems, we trained to predict one type or class.

Notice that the Euclidean distance is used to calculate the closest neighbors between the test instance and the neighbors of the training data.

Note:

In this exercise, the iris data was used. It can be found here: http://archive.ics.uci.edu/ml/datasets/Iris .

KNearestNeighbors implementation:

```
"""
Calculates the accuracy of a kNearestNeighbors classifier.
"""
def kNearestNeighbors(trainingData, testData, k = 1):

    predictions = []
    for testInstance in testData:
        neighbors = getNeighbors(trainingData, testInstance, k)

        prediction = getResponse(neighbors)

        predictions.append(prediction)

    return getAccuracy(testData, predictions)
```

Once the k-nearest neighbor algorithm is implemented, it is necessary to evaluate the **prediction performance** by calculating the **classification error** or **regression error**.

The cross-validation will divide the data set into **folds** of the same size. Depending on the number of folds we will carry **1-leave out cross validation**, 2-leave out cross validation, n-leave out cross-validation.

The objective is to get the best possible validation.

The basic idea is to partition the data set into k folds of equal size (K). After that, there is an iteration/rotation process in which in each iteration a new test fold is selected.

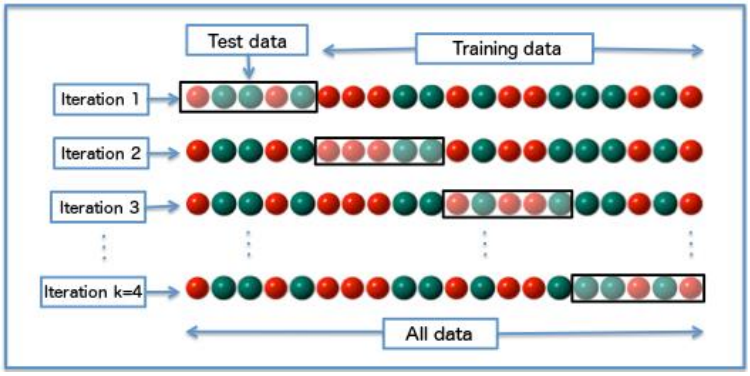

Figure 1. https://en.wikipedia.org/wiki/Cross-validation_(statistics)

The accuracy or error obtained on each iteration is stored and the mean is obtained.

This way, the assessment of the learning algorithm will be more accurate because **we are using all our data for training and all our data for testing.**

Cross-Validation implementation:

```
"""
Evaluation of the predictionPerformance using crossValidation.

samples: independent data set
"""
def crossValidation(samples, foldSize, predictionMethod, neighborNumber = 1):

    k = len(samples) // foldSize

    sumAccuracy = 0

    for fold in range(0,k):
        beginningFold = fold * foldSize
        endFold = beginningFold + foldSize

        trainingData = samples[:beginningFold] + samples[endFold:]
        testData = samples[beginningFold:endFold]

        # get the accuracy of the prediction for that testData
        iterationAccuracy = predictionMethod(trainingData,testData,neighborNumber)

        sumAccuracy += iterationAccuracy

    return sumAccuracy / k
```

Results:

Using cross-validation in the validation of the KNearestNeighbors algorithm, it was obtained an accuracy in the classification of **96%.** Which it is a quite good result.

```
samples = getSamples(r"C:\Users\agazor\agazorDropBox\Dropbox\UTU\APPLICATION_OF_DATA_ANALYSIS\EXERCISE_1\iris.data.txt")
foldSize = 1
neighborNumber = 1
crossValidation(samples, foldSize, kNearestNeighbors, neighborNumber)
```

```
96.0
```

If we find the best number of neighbors for the KNearestNeighbors algorithm, the accuracy of the classification is **98%**.

```
"""
Which k-neighbor gives the best Accuracy.
"""
maxAccuracy = -1
maxNeighbor = -1
foldSize = 1

for neighbor in range(1,len(samples)-1):
    accuracy = crossValidation(samples, foldSize, kNearestNeighbors, neighbor)
    if accuracy > maxAccuracy:
        maxNeighbor = neighbor
        maxAccuracy = accuracy

print ("NeighborNumber: " + str(maxNeighbor) + " = " + str(maxAccuracy))
```