Assignment 3
Clustering of Web Users

Due date: October 18th (23:59)

1 Problem Description

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). In these exercises you are going to use two algorithms on the task of clustering web clients. The aim is to personalize the internet services to different client groups. The clustering is based on previous web requests by the clients, and using the clusters we can reorganize the website and already pre-fetch data (web pages) which the client will probably request the next time he visits. A small portion of the clients will always request more or less the same pages. Note that one client can host several web users with similar preferences. The overall architecture of the clustering and pre-fetching method is well illustrated in the paper of Rangarajan et al. (download it from AI2 Nestor page and read it).

2 The data and Python files

For this exercise we will use the weblogs provided by NASA, which contain all requests made in July and August 1995 (2.5 million weblog entries, 370 MB of data). The data has been pre-processed by extracting the top 70 users who made the most requests (sufficient data about the users must be available in order to be able to perform a good clustering). For these clients, the 200 most requested URLs are retained. A feature vector is generated for each client: it contains 200 0's and/or 1's, a 1 represents that the client has requested that particular URL. These feature vectors will be used further for your clustering algorithms. The data files contain 70 feature vectors (one per client) with 200 dimensions (one per URL request).

Your clustering algorithms should group together web users based on their common preferences (requested URLs). Then, when a particular client visits the website again, the requests specific to its group (cluster) will be pre-fetched. The goal is to minimize the errors by using the smallest number of clusters, while retaining both a good *hitrate* and *accuracy*.

The *hitrate* is the number of requests made from the pre-fetched URLs divided by the number of requests (pre-fetched or not). The *accuracy* is the number of requests made from the pre-fetched URLs divided by the number of pre-fetched URLs (whether useful or not). In general there will always be some 'new' requests that are not pre-fetched. The *hitrate* can be increased by pre-fetching more and more URLs, but the consequence is that most of the pre-fetches will be useless and the *accuracy* goes down. The vice versa is also true. The data necessary for this programming assignment consists of 4 files:

- *train.dat* - containing the training data (70 vectors of dimensionality 200) obtained from the NASA weblog of July 1995, clustering will be applied to this data (training)
- *test.dat* - containing the testing data (70 vectors of dimensionality 200) obtained from the NASA weblog of August 1995, the hitrate and accuracy will be computed on this data (testing)
- *clients.dat* - containing the names (IP addresses) of the top 70 clients
- *requests.dat* - containing the list of top 200 requests (URLs)

clients.dat and *requests.dat* are included because they are part of the original data set, but are not needed to complete this assignment.

A number of files with Python code are provided for you as a starting point in your programming:

- *run_clustering.py* - it is the main file that does the data IO (by reading the 4 data files - they must be present in the current directory), handles the cmd line interaction with the programmer and calls two clustering methods
- *kmeans.py*
- *kohonen.py*

These 2 files contain a framework for the clustering task and it is your job to complete the K-means and Kohonen algorithms following the instructions in this document and the comments in the code.

There are already some data structures in place which are supposed to make your life easier. Three important data structures are:

- self.clusters - In kmeans.py, this is a one-dimensional list containing k clusters. In kohonen.py, it is a two-dimensional list containing $N \times N$ clusters. The class Cluster looks like:

```
class Cluster:
    """This class represents the clusters, it contains the
    prototype (the mean of all it's members) and memberlists with the
    ID's (which are Integer objects) of the datapoints that are member
    of that cluster. You also want to remember the previous members so
    you can check if the clusters are stable."""
    def __init__(self, dim):
        self.prototype = [0.0 for _ in range(dim)]
        self.current_members = set()
        self.previous_members = set()
```

- self.traindata -
- self.testdata - These last 2 data structures are lists containing the training and test data respectively. The data consists of floating point values.

The data is read in by "run_clustering.py", then an instance of the user-defined clustering algorithm is made and it gets a copy of the data vectors and clustering arguments. Your job is to actually implement two clustering algorithms in the framework files "kmeans.py" and "kohonen.py" respectively. More specifically, you must implement the *train()* and *test()* functions in the 2 Python files.

NOTE: You may only import functions from the Python Standard Library:
`https://docs.python.org/3/library/`
So you are **not** allowed to use other libraries such as NumPy, SciPy or Scikit-Learn.

3 K-means

The basic idea of this iterative clustering algorithm is to start with an initial random partition, and then assign patterns to clusters so as to reduce the squared-error. The number of clusters k must be specified beforehand. During training the squared-error is minimized.

K-means stepwise:

- Step 1: Make a random partition of the data over the k clusters. Different initial partitions can lead to different final clusterings because clustering algorithms based on squared-error converge to local minima. For simplicity an initial random partitions is chosen.
- Step 2: Generate a new partition by assigning each data vector (pattern) to its closest cluster center. Compute the Euclidean distance between the data vector X and all the cluster prototypes P and select the minimum:

$$d(X, P) = \sqrt{\sum_i (x_i - p_i)^2} \tag{1}$$

- Step 3: Compute new cluster centers as the centroid of the data vectors assigned to the considered cluster. Compute the mean by summing over all cluster members and then dividing by their number.
- Step 4: Repeat step 2 and 3 until the membership of the patterns no longer changes.

4 Kohonen Self Organising Feature Maps (SOM)

SOMs were invented by professor Teuvo Kohonen, a member of the Academy of Finland, and they provide a way of representing multidimensional data in much lower dimensional spaces (1D, 2D, 3D). Reducing the dimensionality of vectors is essentially a data compression technique known as vector quantisation. In addition, the Kohonen technique creates a map that stores information in such a way that any topological relationships within the training set are also maintained (to some degree). In this assignment we use the self organising properties of the Kohonen map to form clusters of web users.

Kohonen stepwise:

- Step 1: Each cluster center is randomly initialized.
- Step 2: For all vectors from the set: The vector is presented to the map, and
- Step 3: All nodes in the map (which represent cluster centers) are examined to find the closest to the input vector in terms of the Euclidean distance. The winning node is known as the Best Matching Unit (BMU).
- Step 4: The radius r of the neighbourhood of the BMU is calculated. This is a value that starts large, typically set to half the "size" of the map, but diminishes with each training epoch. All the nodes found within this radius are inside the BMU's neighbourhood.
- Step 5: Each node in BMU's neighborhood (the nodes found in step 4) is adjusted to make it more like the input vector:
$$P_{NEW} = (1 - \eta) * P_{OLD} + \eta * X \tag{2}$$
- Step 6: Repeat steps 2, 3, 4, 5 running over the whole training dataset several times ($tmax$) and at each training epoch decrease the radius of the neighborhood r and the learning rate η.

The algorithm requires you to specify the size of the map N (a square map with $N \times N$ nodes will be created) and the number of training epochs $tmax$ (the total number of runs over the training data, e.g. 100).

$$\eta = 0.8(1 - \frac{t}{tmax}) \tag{3}$$

$$r = \frac{N}{2}(1 - \frac{t}{tmax}) \tag{4}$$

In this exercise it will suffice to implement a simple Kohonen SOM with a linearly decaying learning rate η and neighbourhood size r (t is the training epoch counter). Also the neighbourhood "shape" is rectangular and the learning rate is the same for all the neighbours (i.e. not distance related).

5 Testing your algorithms

After programming the two described algorithms, their performance must be tested. After clustering, the prototypes (cluster centers) will be vectors with real values between 0 and 1. A decision must be made whether or not to prefetch the corresponding URL and a threshold must be set. This "prefetch_threshold" is an important parameter that directly influences the hitrate and the accuracy. Using your implemented algorithms, you must study the following problems:

- the influence of the "prefetch_threshold" on the performance (*hitrate* and *accuracy*)
- the influence of the number of clusters on the performance (*hitrate* and *accuracy*)
- the influence of the clustering algorithm on the performance (*hitrate* and *accuracy*)

For the prefetch threshold and number of clusters, do a parameter sweep where you compare several values. Include the results regarding the above three points in your report.

Observation: Put clear comments in the code to explain your implementation of the algorithms. Mark your comments with two pound signs (i.e. this sign: ##) at the beginning of the lines to make them distinguishable.

6 Final Questions (to be answered individually)

1. Describe the influence of the three parameters (prefetch_threshold, number of clusters and clustering algorithm) on the performance of the system (hitrate and accuracy).
2. Compare the two (K-means and Kohonen-SOM) clustering algorithms in: the way they work, their computational complexity, their final performance.

7 What to hand in

1. Your code in a zip file. Please include the data as well so that it can be ran "out of the box".
2. A pdf report, generated from LaTeX, with (at least) the following:
   (a) Explanation of your implementation and design choices, for both clustering algorithms and their respective training and testing stages.
   (b) The results of the testing of both algorithms in terms of hitrate and accuracy, with respect to the prefetch threshold, number of clusters and algorithm. This can be done as a group, as long as the actual discussion of these results (i.e. the individual questions) are done individually.
   (c) For each team member: the answers to the individual questions.