Artificial Intelligence (CS 4253/6613)

Project 1: Adaptation (Nearest neighbor, Genetic Algorithm& Simulated Annealing

In this project you will write Python code to implement a simple supervised learning scheme as well as two local search and optimization procedures. You will work individually and turn in commented code, runs on some test data to show your implementations work as expected, and a report with analysis of your results when you vary algorithm options or paramters.

Part A. *k* Nearest Neighbor (kNN) Supervised Learner (40 points)

Write a program that performs *supervised classification* using the kNN algorithm which assigns the majority label of the k closest stored labeled instances to a new input instance. For this project, you will experiment with two variations of kNN training which either stores all or a subset of the training instances:

Store All: Store every training instance. This option makes training trivial, but has higher storage and classfication costs.

Store errors; Store the first k points of each class that is input. Thereafter, store an input instance only if it is misclassified by the $k - NN$ algorithm, using only the previously stored instances. This option has relatively higher training complexity, but has lower storage and classficiation costs. Note: the points stored in the learnt model depends on the order in which the input instances are presented. You might, but is not required, to avaerage performance over several random ordering of the input sequence.

You are provided one artifiial dataset of 1000 points in a file **labeled-examples** which contains one input instance per line in the form *(class, x, y, name)*, where *class* represents the classification of a datapoint of name *name* and described by the real-valued attributes x and y.

Provided a labeled data set, your program should perform N-fold cross validation ($N = 5$), where you first divide the data set into N equal-sized blocks, and then repeatedly test performance on each of the blocks in turn, while training on the remaining blocks (see Algorithm 1). Report the average training and testing classification accuracy over all the N blocks. The classification accuracy of any instance, either in the training or in the test set, is determined by comparing the provided label with the majority label of the k nearest stored instances. For this project the `trainModel` call will invoke one of the kNN variants to store a subset of the training instances. Subsequently, the `eval` call will use those stored instances to classify the provided set of examples.

Algorithm 1 N-fold Cross-Validation on dataset, D

```
procedure CROSS-VALIDATION(N,D)
    Partition D into N roughly equal splits, $D_i$, $i = 1, \ldots, N$
    $Acc_{train} \leftarrow 0$, $Acc_{test} \leftarrow 0$
    for $i \leftarrow 1$ to $N$ do
        $D_{train} \leftarrow D \setminus D_i$, $D_{test} \leftarrow D_i$
        $M_i \leftarrow trainModel(D_{train})$                        ▷ call kNN learner to build model
        $Acc_{train} \leftarrow Acc_{train} + eval(M_i, D_{train})$   ▷ Accumulate training set accuracy
        $Acc_{test} \leftarrow Acc_{test} + eval(M_i, D_{test})$      ▷ Accumulate test set accuracy
    Return $\langle Acc_{train}/N, Acc_{test}/N \rangle$              ▷ Return average train and test set accuracies
```

In addition to the artificial dataset provided, primarily to help develop your code, you should also present results from your learning program on some real world data set from the https://archive.ics.uci.edu/ml/datasets.php. Choose data sets that with only real or integer valued attributes.

Part B. Local Search Algorithms (60 points)

In this part you will implement stochastic optimization techniques. You will work individually and turn in commented code, runs on some test data to show the functions work, and a report with analysis of your results when you vary algorithm options or paramters. The stochastic optimization techniques you will implement and experiment with are the following:

Simulated Annealing (SA): You will implement the single point approach of *simulated annealing*, as described in Figure 4.5, page 115 of the Russell & Norvig textbook (4th Edition).

Genetic Algorithms (GA): You will implement the multi-point approach of *genetic algorithms*, as described in Figure 4.8, page 119 of the Russell & Norvig textbook (4th Edition).

You will evaluate the performance of the algorithms on some difficult test problems, provided as part of the assignment, as well as other problems of varying degree of difficulty in terms of the number of local optima, function smoothness, etc. of your choice. You are required to vary the algorithm parameters, such as annealing schedule in SA and crossover/mutation rates in GA. You are also encouraged to try variants of these algorithms, such as no crossover in GAs. Use bit-string representations of suitable size and map bit strings to real values in the range of functions (some code provided for this).