TERM PROJECT II,
MACHINE LEARNING

1. Write a program that implements backpropagation of error for a predefined number of output and hidden neurons, and for a predefined number of attributes. Apply this program to 2-3 benchmark domains from the UCI repository. Choose domains with numeric attributes only.

   For learning rate, use $\eta = 0.01$. Choose a relatively small number of hidden neurons, say, 10. Plot a graph showing how the error rates on the training and testing sets gradually decrease, perhaps reaching some minimum beyond which no improvement seems possible.

   In the next stage, experiment also with different values of the learning rate. Will the network's ability to converge be affected? Evaluate convergence in terms of the number of epochs, and in terms of the best error rate that can be achieved.

2. Implement the same program as in the previous exercise. Experiment with different numbers of hidden neurons, say ranging from 5 to 50. Make observations as to how the growing number of hidden neurons affects the ability to converge to low error rate. Also, observe the potentially different behavior in terms of the error rates on the training versus testing sets.

3. Write a program that implements a radial basis-function network. Train the output layer with perceptron learning. Evaluate this program on 3 numeric domains from the UCI repository, observing error rates on the training and testing sets. Use domains that do not have more than a few hundred examples.

4. Implement the same program as in the previous exercise. This time, however, use benchmark domains with thousands of examples. For the centers of radial-basis functions, choose a random subset of the N examples. Experiment with different values of N.

5. Apply the decision-tree generating software from WEKA to 3 domains from the UCI repository. Experiment with different values of the pruning constant, and prepare graphs illustrating how the changing values of this constant affect testing-set errors. Note that different data may call for different amount of pruning. Optionally, show a graph showing how the pruning constant affects the size of the induced decision tree.

6. Choose a big domain from the UCI repository. Leave aside 30% of the examples for testing, using the remaining 70% for training. Induce several decision trees, each from a different-sized subset of the training set. Plot a graph where the horizontal axis gives the number of examples, and the vertical axis gives the computational time spent on the induction. Plot another graph where the vertical axis will give the error rate on the testing set. Discuss the obtained results.

7. Implement the basic version of *Adaboost*. The number of voting classifiers is determined by a user-set constant. Another user-set constant specifies the number of examples in each training set, T_i. The weights of the individual classifiers are obtained with the help of *perceptron learning*. Apply the program task to some of the benchmark domains from the UCI repository. Make observations about this program's performance on different data. For each domain, plot a graph showing how the overall accuracy of the resulting classifier depends on the number of subclassifiers. Also, observe how the error rate on the training set and the error rate on the testing set tend to converge with the growing number of classifiers.

8. Implement the *stacking* algorithm for different base-level learning algorithms and for different types of the master classifier. For the base-level algorithms, you can use either software from WEKA or you previous implementation from Project 1. Apply the implemented system to a few UCI-domains, and observe its behavior.

9. Design an experiment that illustrates the problem with imbalanced class representation (See Section 10.2 in the textbook). To address this problem, use the majority-class underampling mechanism based on one-sided selection. For performance evaluation, use *precision* and *recall* discussed in Section 11.2.

10. Implement the k-means algorithm, and write a program that shows its behavior (on 2-dimensional data) by a graphic animation.

11. Experiment with Reinforcement Learning, following the ideas of Computer Assignment 2 from Chapter 17 in the texbook.

12. Implement the Genetic Algorithm and apply it to you own synthetic domain. Show how the convergence speed (the number of generations needed to find a solution) is affected by different mutation rates.