
Problem assignment 10

Due: Thursday, April 12, 2019

Problem 1. Feature/Input ranking

Consider the dataset in file *FeatureSelectionData.txt*. The dataset consists of 259 examples (in rows) where each example is defined by 70 dimensional input vector (represented in columns) and an associated binary label (in last column).

Part a. Write and submit a function $Fisher\_score(x, y)$ that takes as arguments a vector of one-dimensional inputs x and a vector of binary outputs y and calculates the Fisher score as defined in the lecture. Use this function to evaluate the different dimensions of the input space (there are 70 dimensions) to estimate their individual predictive power. Please report the ordered list of dimensions with the top 20 Fisher scores, and their Fisher score values. The dimensions should be labeled from 1 to 70 depending on their position in the dataset.

Part b. Write and submit a function $AUROC\_score(x, y)$ that takes as arguments a one-dimensional vector of inputs x and a vector of outputs y and calculates the area under the ROC curve. You may use Matlab functions to calculate the area under the curve for this purpose. Similarly to part a, evaluate the different dimensions of the input space and their individual predictive power based on AUROC score. Again, report the ordered list of 20 dimensions with the top 20 AUROC scores, and their values. Compare the results from part a and part b and discuss your findings. Are the ordered lists the same? In general, do you expect them to be the same.

Problem 2. Bagging of classifiers

In this problem you will have an opportunity to experiment with the bagging approach that lets you combine multiple classification models into an ensemble with the hope that the ensemble will improve the classification performance.

You have received a bagging code that lets you combine bagging approach with multiple classification models. The bagging function code is implemented in file: $Bag\_classifier.m$. The input and output of the function are as follows:

```
function [test_y, E] = Bag_classifier(tr_x, tr_y, test_x, params)
```

```
% Inputs:
%  tr_x - Train patterns
% tr_y - Train targets
%     test_x   - Test  patterns
%     params:
%        classifier
%     NumberOfIterations (Models),
%        Learner's parameters]
%
% Outputs
% test_y - Predicted targets
%   E               - Errors through the iterations
```

The *Example_SVM.m* file shows how to use the bagging procedure with the SVM model implemented in the *SVM_base.m* file. The number of SVM models built in this example is 5.

Part a. Use the dataset in *hw10_train.txt* *hw10_test.txt* files (last columns are class labels) and the code provided to test and compare the performance of the base SVM model and the bagged SVM model by reporting their the train and test errors. In this experiment, please vary the number of models used in bagging from $T = 2, \ldots 10$. For each T, repeat the learning 20 times and average the train and test errors (for each T). Plot the results of the base SVM (T=1) with the bagged SVM for ($T = 2, \ldots 10$) in a graph. Analyze and discuss the results.

Part b. Following the *SVM_base.m* code definition, write *DT_base_full.m* function that implements the decision tree algorithm without pruning. Submit the code. Run and compare the *DT_base_full.m* alone, and in combination with the bagging procedure for $T = 2 \ldots, 10$, similarly to the experiment in Part a. Plot the graphs showing the train and test errors performances for different T. Analyze and discuss the results.