Problem assignment 6

Due: Thursday, March 15, 2018

In this assignment we continue our investigation of the “Pima” dataset. As in the previous assignment, you can download the dataset (pima.txt) and its description (pima desc.txt) from the course web page. In addition to the complete dataset pima.txt, you have pima train.txt and pima\_test.txt you will need to use for training and testing purposes.

Problem 1. Support vector machines

Support vector machines represent yet another technique one can apply to the problem of binary classification. The idea is to find the hyperplane that separates the examples in two classes the best. The best hyperplane is defined in terms of the maximum margin. The learning problem reduces as usually to optimization, in this case, a quadratic optimization problem.

There is a number of implementations of SVM algorithms with better or worse running time performances. Here we use a Matlab code implementing SVM solver for the linear decision boundary proposed by *O.L. Mangasarian and D. Musicant*. The paper describing this method can be downloaded electronically at: <http://www.ai.mit.edu/projects/jmlr/papers/volume1/mangasarian01a/html/>

The SVM solver is in files ***svml.m*** and ***svml\_itsol.m*** that can be downloaded from the course web page.

***svml­­\_itsol.m*** is a slightly modified version of the original program by O.L. Mangasarian and D. Musicant. To run it you call ***svml.m*** that takes care of **converting outputs from 0,1 class labels to -1,1** (!!!) and sets other parameters of the Lagrangian SVM.

Part a. Use the ***svml*** code to learn the weights w and b (bias) of the linear model on the training set. Assume the cost for crossing the boundary (a paramater of the svml procedure) is 1.

Part b. Write and submit a function ***apply­\_svlm(x, w, b)*** that takes an input vector x, and paramaters w, and b of the liner SVM and outputs the class decision (use 0 and 1) for the input x.

Briefly, the class is 1 if w \*x + b > 0, and 0 otherwise.

apply\_svlm.m

Part c. Use function ***apply\_svlm(x, w, b)*** to calculate the confusion matrices and other stats for both the training and testing data. More specifically, please report the confusion matrix, misclassification error, sensitivity and specificity for both the training and testing set.

For training data:

Confusion matrix:

|  |  |  |
| --- | --- | --- |
|  | *Target-1* | *Target-0* |
| *Predict-1* | *115* | *85* |
| *Predict-0* | *40* | *299* |

Mis-classification (train) = 0.2319

sensitivity = 0.7419

specificity = 0.7786

For testing data:

Confusion matrix:

|  |  |  |
| --- | --- | --- |
|  | *Target-1* | *Target-0* |
| *Predict-1* | *20* | *48* |
| *Predict-0* | *50* | *111* |

Mis-classification (test) = 0.4279

test\_sensitivity = 0.2857

test\_specificity = 0.6981

Part d. Compare the above results to the results from Homework assignment 5 obtained for the logistic regression and the Naive Bayes models. Which model do you think performed the best?

So, from SVM, I have the misclassification error for training/testing set as:

Mis-classification (train) = 0.2319

Mis-classification (test) = 0.4279

From last homework, I have the logistic regression on testing data with:

*The final misclassification error for training/testing set is*

*Train\_error = 0.3061*

*Test\_error = 0.2707*

The error from Naïve Bayes:

*misclass\_train = 0.2393*

*misclass\_test = 0.4454*

I will still prefer the logistic regression as both SVM and Naïve Bayes have model cannot be generalized well into the testing dataset – it has testing error much higher than the training error.