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.*

**Part e.** Use **perfcurve** to plot the ROC curve and calculate AUC for the SVM model on the testing set. To do so, for each data example x in the testing data, calculate the score score(x) = wTx + b. Compare the AUC for the SVM with the AUCs for the logistic regression and for the Naive Bayes in homework assignment 5. Which model is better?

*svm\_AUC\_test = 0.4821*

*svm\_AUC\_train = 0.8368*

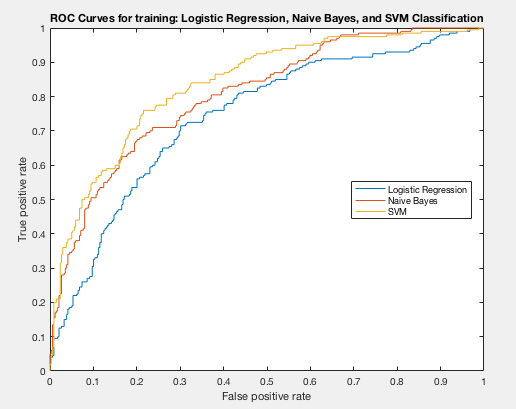
*NB\_AUC\_test = 0.5134*

*NB\_AUC\_train = 0.8060*

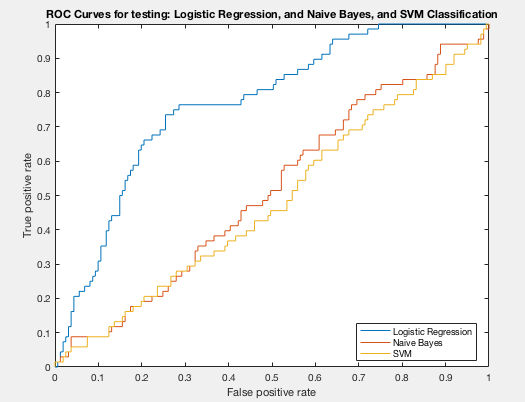
*lg\_AUC\_test = 0.7695*

*lg\_AUC\_train = 0.7430*

*ROC of Training:*



*ROC of testing*



**Problem 2. Decision trees**

The decision tree approach is yet another classification method we covered in the course. The method builds a tree by recursively splitting the training set using one of the attributes by optimizing the gain with respect to some impurity measure.

**Part a.** The script ***run­\_DT.m*** shows how to train, display, and apply the decision tree in Matlab. The script first builds a default tree with minimal restrictions on its size, and after that the tree obtained by restricting the number of nodes in the tree.

Please run and familiarize yourself with the code. What do you think, which tree is better for prediction, the unrestricted or restricted tree? Why? Should we always try to backprune it?

*Default tree:*

*Error\_train = 0.0742*

*Error\_test = 0.2489*

*New tree:*

*error\_train = 0.1633*

*error\_test = 0.2576*

*Although new tree has little bit lower error rate on the testing data set than the default tree, but the default tree definitely has the overfitting problem with the testing error much higher than the training error. The new tree is much better than default tree in the generalization error.*

**Part b.** Experiment with the decision tree function **fitctree.m** and its optional parameters, modifying the algorithm and the tree built.

Report the results of your investigations in the report by listing the settings used for the tree learning algorithm and obtained results. You can find the different settings in the matlab help documents.

*I tested with different settings based on parameter sets (Max Number of Split; Split criterion; Prune Criterion), and then check the training/testing error by using different settings.*

*The best setting is to set:*

new2\_tree=fitctree(x,y,'MaxNumSplits',25, 'MinParentSize',20,'MinLeafSize',15,'splitcriterion','deviance', 'PruneCriterion', 'impurity');

*This setting is able to have:*

*error\_train = 0.1744*

*error\_test = 0.2271*

**Problem 3. Neural network toolbox in Matlab**

In this problem you will learn about and explore the neural network toolbox.

**Part a.** In homework 5 you were asked to run a gradient algorithm for learning the logistic regression model. However, the logistic regression model is also supported and implemented in Matlab within its Neural Network toolbox. Please familiarize yourself and run **logistic\_NN.m** function that is given to you and implements the logistic regression model using the toolbox functions.

Try to change the parameters of the model, such as the optimization method and the number of epochs. Report the weights with the best mean misclassification rate for the test set and any graphs you have found interesting.

*With the parameter following:*

net.trainParam.epochs = 2000; %20000

net.trainParam.show = 10;

net.trainParam.max\_fail=50; %10000

%%% use conjugate gradient to train the model

net.trainFcn='traincgf';

*class\_error\_train = 0.2449*

*mse\_error\_train = 0.4905*

*class\_error\_test = 0.2533*

*mse\_error\_test = 0.4874*

*by increase the # of epochs, the following setting is able to decrease the training error and testing error:*

net.trainParam.epochs = 20000;

net.trainParam.show = 10;

net.trainParam.max\_fail=10000;

%%% use conjugate gradient to train the model

net.trainFcn='traincgf';

class\_error\_train = 0.2301

mse\_error\_train = 0.4835

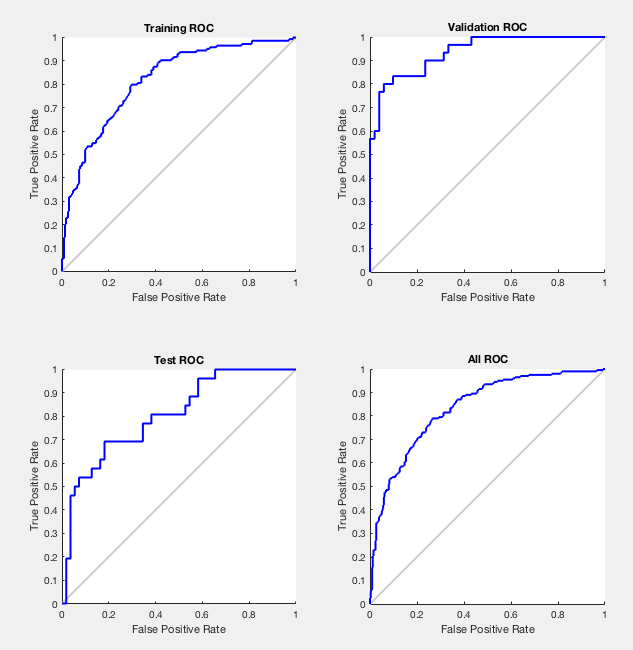
class\_error\_test = 0.2140

mse\_error\_test = 0.4572

*Weights = 0.9191 3.0899 -0.9560 0.3309 -0.3488 2.4485 0.9197 0.5833*

*Bias = -0.0700*

*ROC curve for training/testing/validation dataset*



**Part b.** Multilayer neural network.

The limitation of the logistic regression model is that it uses a linear decision boundary. One way around this is problem is to use non-linear features in combination with a linear model. However, in this case feature function must be fixed and selected in advance. Multilayer neural networks allow us to represent non-linear models by cascading multiple nonlinear units. Multilayer neural networks can be built with the NN matlab toolbox.

Write a program **main3.m** that implements a neural network with two hidden units, that is, there are two nonlinear units we feed the input to, and one unit that combines their results. Run the program for 2000 epochs. Calculate the mean misclassification errors for the training and testing data. Report errors and compare them to results obtained for the logistic regression model for Part a. Which model is better? Why?

*Use the setting of:*

net.trainParam.epochs = 2000;

net.trainParam.show = 10;

net.trainParam.max\_fail=50;

%%% use conjugate gradient to train the model

net.trainFcn='traincgf';

*class\_error\_train = 0.2412*

*mse\_error\_train = 0.4866*

*class\_error\_test = 0.2358*

*mse\_error\_test = 0.4689*

*Under the same setting, this two hidden unit neuro network performed a little bit better than simple logistic regression.*