University of Pittsburgh
CS1675 Introduction to Machine Learning — Handout 6
Professor Milos Hauskrecht — March 11, 2021

Problem assignment 6

Due: Thursday, March 18, 2021

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. The dataset has been obtained from the UC Irvine machine learning repository:
$http://www1.ics.uci.edu/\sim mlearn/MLRepository.html$.

Problem 2. 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 different implementations of SVM algorithms with better or worse running time performances. The SVM solver we will use in this assignment relies on two files *svml.m* and *svml_itsol.m* given you you. To run it, please call svml.m that takes care of converting outputs from 0,1 class labels to -1,1 (!!!) and sets other parameters of the SVM.

Part a. Use the svml code to learn the weights $\mathbf{w}$ and b (bias) of the linear model on the training set. Assume the cost for crossing the boundary (a parameter of the svml procedure) is 1.

Part b. Write and submit a function $apply\_svlm(x, w, b)$ that takes an input vector $\mathbf{x}$, and parameters $\mathbf{w}$, and b of the liner SVM and outputs the class decision (use 0 and 1) for the input $\mathbf{x}$. Briefly, the class is 1 if $\mathbf{w^T x + b} \geq \mathbf{0}$, and 0 otherwise.

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.

Part d. Compare the above results to the results for the logistic regression model (Assignment 5). Which model do you think performed the best?

Optional. If you are interested experimenting with existing SVMs tools including tools supporting non-linear kernels please check out the following sofware packages: liblinear, libsvm and svmlight. All these can be interfaced with Matlab.

Problem 2. ROC analysis

The ROC analysis let us explore the ability of the classification model to discriminate in between the two classes including possible sensitivity and specificity trade-offs. In the ROC analysis we assume a changing threshold for calling class 1 based on the projection defined by the model. This can be $P(y = 1|\mathbf{x})$ for the logistic regression and the Naive Bayes, or $g_1(x) = \mathbf{w^T x} + \mathbf{b}$ for the SVM.

Part a: Familiarize yourself with the function *perfcurve* in matlab that lets you calculate coordinates of points defining the ROC curve, as well as, the area under the ROC curve (AUROC).

Part b. Use the function *perfcurve* to plot the ROC curves and calculate AUROC on the testing set for the Logistic regression from homework assignment 4, and the SVM model from this assignment. All models should be trained on the training set.

Part c. Please include the ROC curves and the AUC statistics in the report. Compare the ROC curves and their AUC statistics. What do you think, which model is better?

Problem 3. Deep learning toolbox in Matlab

In this problem you will learn about and explore the deep learning toolbox in the Matlab that lets you build and learn various neural network models. Please note that the deep learning toolbox prior to Matlab distribution 2018b was called Neural Network Toolbox. So if you have 2018a version or older you will need to refer to 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 Deep Learning 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.

  Briefly, the key part of the code is line:

  ```
  net=patternnet([]);
  ```

  that constructs a simple neural network that corresponds to the logistic regression model. patternnet uses a set of default parameters, such as (logistic sigmoidal) to

define the form of the output function in the output layer. In general you can view and set specific parameters of the net variable constructed with patternnet, that are related to the structure of the network and various optimization parameters. Lines:

```
net.trainParam.epochs = 2000;
net.trainParam.show = 10;
net.trainParam.max_fail=5;

%%% set optimization to conjugate gradient to train the model
net.trainFcn='traincgf';
```

set the parameters driving the optimization procedure. For example, *traincgf* says the parameters of the logistic regression model should be optimized using the conjugate gradient descent procedure. The line:

```
train(net,x',y')
```

optimizes (learns) the weights of the logistic regression model, where x' define a matrix of intputs and y' a vector of outputs (labels 0 or 1). Finally, line

```
res_test = net(x_test');
```

applies the model to the test set.

- 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 easily with the Deep learning matlab toolbox. Briefly, by adapting the line defining the logistic regression model on *logistic_NN.m* as:

  ```
  net=patternnet([2]);
  ```

  you create a neural network with one hidden layer with 2 nonlinear units. Similarly, by using

  ```
  net=patternnet([3 5]);
  ```

  you create a neural network with 2 hidden layers, the first layer has 3 units and the second hidden layer has 5 units.

  By modifying the *logistic_NN.m* code please explore NNs with one hidden layer and 2, 3, 5 and 10 hidden units. Run the program for 2000 epochs. Always 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 the best?

- Optional (extra credit). Experiment with neural networks in Deep Learning toolbox by changing the network layout, that is, the number of hidden layers, number of units per layer. Other things you may change are different activation functions, optimization parameters, such as the number of epochs, etc. Report any findings you make.