Problem assignment 6

Due: Thursday, February 27, 2020

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:

Problem 1. 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:

  \[ \text{net} = \text{patternnet}([]); \]

  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:

  \[ \text{net.trainParam.epochs} = 2000; \]
  \[ \text{net.trainParam.show} = 10; \]
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 inputs 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.
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 \textit{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?

• Part b. Experiment with the decision tree function \textit{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.