

Problem assignment 6

Due: Wednesday, March 21, 2007

In this homework we continue to study and experiment with "Pima" dataset. The dataset (*pima.txt*) and its description (*pima_desc.txt*) can be downloaded from the web. The dataset has been obtained from the UC Irvine machine learning repository [http : //www1.ics.uci.edu/ ~ mlearn/MLRepository.html](http://www1.ics.uci.edu/~mlearn/MLRepository.html). In addition to the complete dataset *pima.txt*, you have *pima_train.txt* and *pima_test.txt* you need to use for training and testing in problems 1-3.

Problem 1. Logistic regression in Matlab

The logistic regression model and the procedure for training weights can be implemented in Matlab using functions in the Neural network toolbox. File *logistic_NN.m* illustrates how one can build, train and simulate the model. Tasks:

- (a) Familiarize yourself with the program in *logistic_NN.m* and NN toolbox. Run the program for 2000 epochs and calculate the mean misclassification errors for the training and testing data. Report your results. Note that the neural network toolbox lets you to observe the errors on fly. However, also note that it uses LMS errors and not missclassification errors.
- (b) Program *logistic_NN_graphs.m* modifies *logistic_NN.m* and allows you to plot the progress of mean misclassification errors during learning. Report the results obtained by the program and its analysis.
- (c) Experiment with the *logistic_NN_graphs.m* 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 graphs you have found interesting.

Problem 2. 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. Multilayered neural networks allow us to represent non-linear models by cascading multiple adaptive logistic units. Multilayered neural networks can be built relatively easily with the NN toolbox. File *multi_NN.m* illustrates how one can build, train and simulate a multilayer neural network with 2 hidden units. Tasks:

- (a) Familiarize yourself with the program in *multi_NN.m* and NN toolbox capabilities to model multilayer networks. Run the program for 2000 epochs and calculate the mean misclassification errors for the training and testing data. Report errors and compare them to results obtained for the logistic regression model. Which model is better? Why?
- (b) Run *multi_NN_graphs.m* to obtain the graphs for the misclassification errors, report the graphs and its analysis.
- (c) Experiment with neural networks with 2, 3, 5 and 10 hidden units, while changing other learning parameters, e. g. the optimization method or number of epoch. Analyze and compare the results. Submit the misclassification error graphs.

Problem 3. 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.

Write and submit a Matlab program that:

- Loads training and test data.
- Calls linear SVM solver to learn the linear decision boundary;
- Computes the mean misclassification error for both the training and test data.
- Computes the confusion matrix for the test set. Write a special function *confusion_matrix* that takes class labels from the data and compares them to those computed by the classifier.

In your report include the misclassification errors and confusion matrix obtained for the train and test sets. Compare the result to the results of the logistic regression and neural network models.

Optional. If you are interested in SVMs with non-linear kernels and would like to try them on the pima dataset you can find and download the Matlab code for SVMs from <http://www.kernel-machines.org/> website. Try quadratic or cubic kernels.

Problem 4. Evaluating generalization performance

The average test misclassification error measures the capability of a learner to classify correctly examples not used in the training (learning) stage and thus reflects its capability to generalize. However, by performing experiments with datasets we can observe that misclassification errors can vary depending on the test and training set split.

Write a program (you do not have to turn it in) that performs the evaluation of three classifier models via random subsampling. The classifier models to be used are: (1) the logistic regression (from Problem 1) (2) the neural network with 2 hidden nodes (from Problem 2), and (3) the SVM (from Problem 3).

The evaluation program should.

- Divide 'pima' data into the training and test set so that 30% of examples should go into the test set. Use function *divideset2.m* you wrote earlier for this purpose.
- Repeat the learning of models thirty (30) times, but each time on different train/test split. Use *divideset2.m* function to randomly divide the data (keep the 30-70% split fixed).
- Compute and reports the average of confusion matrices (in the ratio form) and the average and standard deviation of the mean misclassification errors (for both training and testing stages) over all train/test sets.

Report the results obtained by via averaging. You should observe that the mean misclassification results vary for different training and testing choices. If you have to evaluate the learner and its capability to generalize on the learning problem, which would you prefer, averages or single train/test split ? Explain why? Which classifier do you think would be the best choice for the pima dataset out of the three models tested in the assignment?