Milestone 3 Project Report

Team Members: Preeti Singh and Alex Degtiar
Team Name: Eye of the Tiger

Collaboration Policy Report

Did you receive any help whatsoever from anyone in solving this assignment? No. Only us two (Alex Degtiar & Preeti Singh), did it in a team.

Did you give any help whatsoever to anyone in solving this assignment? No.

Section 1 - Tuning the SMO classifier

Section 1.1 : Background

SMO is used primarily for training a support vector machine, a supervised learning model used for classification. It is used for solving the optimization problems which arise during the training of these support vector machines.

Literature Source: Platt, John (1998), Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, CiteSeerX: 10.1.1.43.4376

Normal name in literature: Sequential Minimal Optimization (SMO)

Weka class name: weka.classifiers.SMO

Section 1.2: Parameters Varied

a) **Kernel Parameter Gamma**

We decided to use a radial basis function (RBF) kernel for the SMO classifier. The Gamma parameter estimates the influence of a single training example. Lower and higher values of the kernel gamma correspond to “far” and “close” reach of an instance, which can mean the difference between a smooth decision surface and one that has a higher accuracy over the training set. This tradeoff primarily determines by how much the classifier overfits the data. It can also affect the performance of the classifier in training time.

b) **Complexity constant** (Also called as C Parameter) :

The C parameter controls how soft the class margins are by determining how many instances are used as support vectors. Somewhat similar to the gamma function, the C can be a means of controlling overfitting by trading off between a smooth surface and a correct classification. A larger C can affect the complexity of the hyperplane, and so subsequently support larger feature spaces.

How we changed the parameters:

1) Kernel Parameter Gamma : We changed the kernel to RBFKernel (Radial Basis Function Kernel). Choosing Kernel function was an important part of our parameter selection. After reading through references we found that a low degree polynomial kernel or RBFKernel with a reasonable width would give good accuracy. We varied the gamma attribute of the RBF kernel between 10^{-5} to 10^{2}, which affected the accuracy of final model.

2) C Parameter: we tried to vary the value of value of C parameter between 1 to 16. The error rate typically steadily decreased with higher C values. The best value of the C parameter was often 16, but ended up varying as low as 3 for some datasets.

The max error ratio across the datasets without tuning was 1.260274. After tuning the kernel gamma and SMO complexity, the max error rate got reduced to 1.219178. The average error rate dropped from 0.823141 to 1.260274 after the tuning. The Kernel Gamma values values ranged between 0.01 to 1.00 and their corresponding error rates were 0.782609 and 0.941176 respectively.

Section 1.3 : Final Classifier Construction

Since the two parameter may be dependent on each other, we did a grid search over the full range of values for both parameters. The GridSearch meta classifier was configured over the range specified above, and picked the optimal point for each dataset.

Section 2: Tuning the AdaBoostM1 Classifier

Section 2.1 : Background

AdaBoostM1 is a type of boosting algorithm which enhances the performance of some base learning algorithm, and can significantly reduce its error rate. This learning algorithm is called as “weak learner”, since the types of base classifier AdaBoost performs best on are those with low accuracies close to 0.5. Decision trees are often a good choice for weak learners (as opposed to something like NaiveBayes), and in our case we chose J48 as the underlying weak learner for AdaBoostM1.

These boosting algorithms run the weak learner on various distributions of the training data and then combine the results into a single composite classifier.

Literature Source: Yoav Freund et. al; Experiments with a New Boosting Algorithm; Jan 22, 1996

Normal Name in Literature: Boosting Algorithm (eg: AdaBoostM1)

Weka Class Name: weka.classifiers.meta.AdaBoostM1

Section 2.2 : Parameters used for AdaBoostM1

a) Weak Learner: First we determined that J48 would be a good fit as a weak learner. J48 is a popular decision tree classifier. We then did a separate round of parameter tuning to determine which J48 parameters resulted in the best AdaBoostM1 accuracy.

Parameters tuned for J48:

1) Pruning Confidence (-C) : Pruning the tree helps by removing branches which are not sufficiently helpful. This can be very effective at preventing overfitting, reducing the total model size, and speeding up evaluation performance. Pruning confidence is used to calculate an upper bound on error rate at leaf/node.
In our case, we varied the value of pruning confidence from 0.1 to 0.5 in 0.1-sized intervals. We used 10-fold Cross-Validation in the process of tuning this parameter. We observed that there was a substantial difference in error ratios after tuning the AdaBoostM1.
The default value for this parameter was 0.25. We changed the value from 0.1 to 0.5 to get better performance. We can infer from our results that the max error ratio reduced to 1.03228064 from 2.305882 , and mean error ratio reduced to 0.66145 from 1.191065.

Section 2.3 : How the Final Classifier got Constructed

AdaBoostM1 has the decision stump as the default “weak learner”. To perform parameter tuning, we first changed the default weak learner to J48, which we found to be more flexible to the tuning of its parameters. We then changed the parameter “pruning confidence” between 0.1 to 0.5 and picked the classifier with highest robustness.

Section 3: Tuning MultiLayerPerceptron Classifier:

Section 3.1: Background

MLP is a popular artificial neural network model that can be used as a classifier. It is a feed-forward network, where data flows forward in one direction from input to output. Multilayer Perceptron also uses the backpropagation supervised learning technique to train the network. It can be effective at solving problems that are not linearly separable.

Literature source: Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

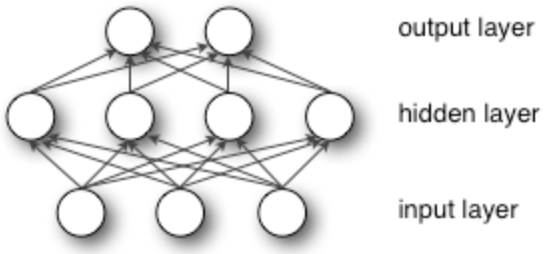
Normal Name in Literature: MultiLayer Perceptron

Weka Class Name: weka.classifiers.meta.MLP

Section 3.2 Parameters tuned for MultiLayer Perceptron:

a) Number of hidden Units (Parameter -N) :

Following is an MLP depicting single hidden layer. By default the value of number of hidden layers is 2.



Source: http://deeplearning.net/tutorial/mlp.html

The number of hidden layers depends on the dataset. If we have a more complicated dataset then we need more hidden layers to model it. The default number of hidden layers is 2. We varied the number of hidden layers from 1 to 5 in order to compare the effectiveness of the different options. Like many parameters, varying this can affect overfitting of data and testing time.

Section 3.3: How the final Classifier got Constructed:

We did CV parameter tuning to vary the parameters and pick the classifier with the best performance. We changed the number of hidden layers to get better robustness. We observed that after tuning the mean error ratio reduced to 0.803902 from 1.080028 and max ratio reduced to 1.131579 from 2.318182.

Section 4: SUMMARY

Among the classifiers we used, the best robustness was initially obtained by MLP. This performed pretty well without any tuning. However, after separately tuning each classifier, the robustness of AdaBoostM1 was substantially improved with the newly tuned values.

The boosted J48 that was appropriately tuned had the best average and max accuracy in general across the datasets compared to the other classifiers we evaluated.