Milestone 4 Project Report

Team Members: Alex Degtiar and Preeti Singh
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

Milestone - 4 Report

Section 1.1: Background

To increase robustness we used a combination of Bagging, Dagging, Boosting and Naive Bayes, similar to the method described by Kotsianti et al[1]. This technique is effective because it combines the different strengths of various algorithms, making the final classifier perform better than any single option. Boosting is good for noise-free data whereas bagging and dagging are stronger in case of noisy data. Furthermore, bagging and dagging reduces variance significantly but boosting reduces both bias and variance[1]. That's why we have built an ensemble using a voting methodology of these first three algorithms with 1 classifier in each of them. We added Naive Bayes into our ensemble because we found that this classifier significantly outperformed our 3-classifier ensemble method on a couple datasets, and that introducing its predictions reduced this difference. We also varied our base learners, did tuning, and some other miscellaneous differences.

We used a sum/average voting methodology to make predictions for this combination of four classifiers. For the voting rule we choose, when classifying a data instance, each sub-ensemble gives a confidence value for each possible class. The voter sums up (or averages) the confidence in each candidate class from each of the sub-ensemble classifiers. The instance is then classified as whichever candidate class had the highest aggregate confidence.

Section 1.2 Varied parameters

Below, we give a brief description of each classifier we used in tuning the ensemble.

Section 1.2.1 : Bagging/weka.classifiers.meta.Bagging

In bagging, bootstrapping is first done to create variants of the original dataset. The base classifier is then trained on each each of these variants, and voting is done on each trained model for classification [2]. We found it performed a bit better with a high number of iterations (15) than the default 10. We also used NNge as the base learner. After some experimentation, we found this gave better results than the J48 learner we used for the other enhancer classifiers.

Base Classifier: NNge/weka.classifiers.rules.NNge

NNge is a nearest-neighbor-like algorithm which uses non-nested generalized exemplars (which are hyperrectangles that can be viewed as if-then rules) [3].

Section 1.2.2: Boosting

Boosting algorithms enhances the performance of some base learning algorithm (called as 'weak learner'), and can significantly reduce its error rate [4].

1) MultiBoostAB/weka.classifiers.meta.MultiBoostAB

MultiBoostAB is a combination of Boosting and Wagging. It reduces the error rate by a significant amount. MultiboostAB has advantage over AdaBoostM1 as it uses the power of boosting to reduce both variance and bias and wagging for reducing the variance significantly [5].

2) AdaBoostM1/weka.classifiers.meta.AdaBoostM1

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 MultiBoostAB (extension to AdaBoostM1) performs best on are those with low accuracies close to 0.5. These boosting algorithms run the weak learner on various distributions of the training data and then combine the results into a single composite classifier. In our case we used MultiBoostAB as the boosting algorithm [6].

Weak Learner for MultiBoostAB and AdaBoostM1: We used a tuned J48 as the weak learner for both.

Tuning: besides the CV tuning described elsewhere, we also increased the number of iterations for the boosting classifiers to 25.

Section 1.2.3: Dagging/weka.classifiers.meta.Dagging

Dagging divides whole data into multiple folds of data and then apply copy of base learners to each chunk of data. Since all the base classifiers are put into vote meta classifier hence we make predictions by majority of vote. In our case we used J48 as the base classifier for Dagging [7]. We ended up increasing the number of folds for Dagging to 15 (from 10).

Section 1.2.4: Tuning

To do additional tuning on this ensemble mechanism, we individually tuned each sub-ensemble classifier, where appropriate. In particular, we found that J48 worked well as a base classifier/weak learner for all relevant classifiers (except Bagging, where NNge performed better). Thus we used J48 as the base learner for these sub-ensemble classifiers, and internally tuned the J48 learner based on its performance on cross-validation of the parent classifier (for

each dataset).

Parameter Varied for Tuning J48:

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 0.05-sized intervals. We used 10-fold Cross-Validation in the process of tuning this parameter and selecting the lowest-error classifier.

After tuning for MultiBoostAB and AdaBoostM1, we found a substantial performance increase. There was also a benefit for Dagging, but not as substantial.

Finally, we we also bothered by the training time for building and tuning all of these classifiers in the ensemble. To make the runtime of our classifier more practical for repeated runs as we tried different tuning approaches, we made multi-threaded versions of the voting and parameter-tuning mechanisms, substantially decreasing the total run time of our classification on a multi-core machine.

Section 2: Results and Discussion

From the results we can infer that after combining bagging, boosting and dagging altogether, there is a significant improvement in robustness. The mean and max error ratio of L4 was reduced by (much) more than 10% as compared to L3.0. We also found a reduction of 10.3% compared to L3.1 on the old dataset, but not quite 10% on the new dataset.

The mean robustness of L3.0 (untuned AdaBoostM1) on the new dataset was 2.37 - a pretty lousy performance. L4's robustness was .757, an increase of more than 68%. L3.0's max robustness was 12, and L4's was 3, a 75% improvement in the max robustness.

Initially, we used the Weka experimenter to identify promising results. Our primary means of incrementally improving our classifier was by then running it locally on the old dataset and evaluating its accuracy. We would experiment with different methods and incorporate the ones that improved the results. Towards the end, we also experimented based on the new dataset on autolab, and incorporated a few enhancement (e.g. using Naive Bayes as one of the sub-ensembles in the voter).

References

1. Combining Bagging, Boosting and Dagging for Classification Problems Knowledge-Based Intelligent Information and Engineering Systems (2007), pp. 493-500, doi:10.1007/978-3-540-74827-4_62 by S. Kotsiantis, D. Kanellopoulos
2. William Cohen's Lecture Slides (Ensemble Methods 1)
3. Literature Source: Brent Martin (1995). Instance-Based learning: Nearest Neighbor With Generalization. Hamilton, New Zealand.
4. Yoav Freund et. al; Experiments with a New Boosting Algorithm; Jan 22, 1996
5. Geoffrey I. Webb (2000). MultiBoosting: A Technique for Combining Boosting and Wagging. Machine Learning. Vol.40(No.2).
6. Yoav Freund et. al; Experiments with a New Boosting Algorithm; Jan 22, 1996
7. Ting, K. M., Witten, I. H.: Stacking Bagged and Dagged Models. In: Fourteenth international Conference on Machine Learning, San Francisco, CA, 367-375, 1997.