Winning the KDD99 Classification Cup: Bagged Boosting

Bernhard Pfahringer
Austrian Research Institute for AI
Schottengasse 3
Vienna, Austria
bernhard@ai.univie.ac.at

ABSTRACT

We briefly describe our approach for the KDD99 Classification Cup. The solution is essentially a mixture of bagging and boosting. Additionally, asymmetric error costs are taken into account by minimizing the so-called *conditional risk*. Furthermore, the standard sampling with replacement methodology of bagging was modified to put a specific focus on the smaller but expensive-if-predicted-wrongly classes.

Keywords

bagging, boosting, sampling, cost-sensitive

1. INTRODUCTION

First of all, be warned, this account is both rather short and kind of informal in tone. For winning such a contest you need to - most of all - be lucky, and in addition put effort in a lot of details. Good performance is not enough, you have to strive for the best that is achievable with your tools. In the following we will explain the steps taken for investigating the data, describe customizations specific to the problem, and our final solution.

2. PRELIMINARY EXPLORATION

In an initial test stage we applied various standard learning algorithm including various incarnations the C5 (trees, rules, and boosted trees, to be specific), Ripper, naive bayes, nearest neighbor, a back-propagation neural network and a radial-basis function neural network. This initial scenario was a kind of inverted cross-validation, where the data was split into ten folds. Only one fold was always used for learning and all the other nine folds for testing. The reason to do cross-validation in that uncommon way was the sheer amount of data available, which was about 5 million examples. Still, excessive runtime requirements led to the cancellation of all experiments involving either Ripper, nearest neighbor, or the two kinds of neural networks. Of the four remaining algorithms, all variants of C5 were performing much better than naive bayes. Of the C5 variants, expectedly boosted trees showed a small, but significant lead. In a further experiment we tried to assess whether more than 10 boosting iterations would significantly improve results, which they didn't. Yet another experiment showed that cost-based voting of several of such boosted ensembles would decrease - as expected - the overall variance in prediction.

3. THE FINAL PREDICTOR

So after all the experience we have gotten from our preliminary exploration and after the final evaluation procedure was published, about a week was left for constructing the final predictor. Due to various resource limitations we decided to construct an ensemble of fifty times ten decision trees. The way this ensemble was constructed can best be described as cost-sensitive bagged boosting:

1. Fifty samples were drawn from the original 5 million odd examples set. Contrary to the standard bagging methodology our sampling mechanism was slightly biased. It always included all of the examples of the two smallest classes U2R and R2L, and 4000 PROBE, 80000 NORMAL, and 400000 DOS examples each. Additionally, duplicate entries in the original data set had been removed beforehand (thus changing the distribution).

2. For each sample an ensemble of ten C5 decision trees was induced using both C5's error-cost and boosting options.

3. The final predictions were computed on top of the 50 single predictions of each of the sub-ensembles by minimizing the so-called conditional risk. This risk is defined in [Duda & Hart 1973], where they show that the Bayes optimal prediction is that class which minimizes this conditional risk. This risk is defined as the sum of the error-costs predicting specific classes times the probabilities of the respective classes.

4. MISCELLANEOUS DETAILS

The training sets mentioned above of about half a million examples took C5 a little less than an hour to process on the machine available, which was a two-processor ultra-sparc2 (2x300Mhz) with 512M main memory, and a 9 GB disc running Solaris 5.6. 50 such samples were processed, yielding 50x10 trees, a process which all together took a bit more than a day in the final *production* run (remember, it is two-processor machine).

A mixture of more or less standard software tools was used in these experiments. All sampling, second level computations, and almost all scripting was done using Allegro Common Lisp, with C5 being used as the base-level learning algorithm. A little Gawk/grep/sed was also used interactively, along with gnuplot for *poor man*'s visualization. The initial exploration stage also utilized a Perl script originally written by Johann Petrak for the ongoing long-term EC

research project METAL [MetaL 1999], which investigates meta-learning.

5. CONCLUSIONS

Contrary to standard research practices, quite a few of the above described decisions were not based on solid scientific enquiry, but rather taken pragmatically as to account for the limited resources available. It might be an interesting endeavor to repeat this investigation in a more scientific manner. That way one could assess the contribution of the different elements more carefully. In summary, it should be mentioned that our solution was not significantly better than the two runner-ups. It simply happened to perform a bit better on exactly those test examples which originated from previously unseen sub-classes of the two rarest classes.

6. ACKNOWLEDGEMENTS

I would like to thank Johann Petrak for this scripting magic, as well as all the other members of the METAL meta-learning project. Winning would not have been possible without having this environment. Financial support for the Austrian Research Institute for Artificial Intelligence is provided by the Austrian Federal Ministry of Science and Transport.

7. REFERENCES

Duda R.O., Hart P.E.: Pattern Classification and Scene Analysis, Wiley, Chichester/London/New York, 1973.

MetaL 1999: An ESPRIT Long-Term Research Project: A Meta-Learning Assistant for Providing User Support in Data Mining and Machine Learning, http://www.cs.bris.ac.uk/ cgc/METAL/