Incremental Machine Learning to Reduce Biochemistry Lab Costs in the Search for Drug Discovery

George Forman
Software Technology Laboratory
HP Laboratories Palo Alto
HPL-2002-141
May 14th , 2002*

E-mail: gforman@hpl.hp.com

supervised machine learning, support vector machines, reinforcement learning

This paper promotes the use of supervised machine learning in laboratory settings where chemists have a large number of samples to test for some property, and are interested in identifying as many positive instances for the least laboratory testing effort. Rather than traditional supervised learning where the chemists would first develop a large training set and then train a classifier, the paper promotes incrementally re-training from each lab test as it completes and then predicting the next best sample to test, as in the field of reinforcement learning. The method outperformed the 2001 KDD Cup thrombin competition winner, partly due to its reduced risk to concept drift from training set to test set.

* Internal Accession Date Only
Approved for External Publication

Incremental Machine Learning to Reduce Biochemistry Lab Costs in the Search for Drug Discovery

George Forman
Hewlett-Packard Labs
1501 Page Mill Rd. MS 1143
Palo Alto, CA, USA 94304
gforman@hpl.hp.com
+1 (650) 857-7835

Abstract

This paper promotes the use of supervised machine learning in laboratory settings where chemists have a large number of samples to test for some property, and are interested in identifying as many positive instances for the least laboratory testing effort. Rather than traditional supervised learning where the chemists would first develop a large training set and then train a classifier, the paper promotes incrementally re-training from each lab test as it completes and then predicting the next best sample to test, as in the field of reinforcement learning. The method outperformed the 2001 KDD Cup thrombin competition winner, partly due to its reduced risk to concept drift from training set to test set.

1. Introduction

In modern drug discovery, chemists test many different organic molecules to identify those that successfully bind to a target site on a given receptor. This activity was recently used as the basis for a challenge problem in the 2001 KDD Cup [1]. In this classification competition, DuPont Pharmaceuticals Research Laboratories made available the results of lab experiments that tested 1,909 organic compounds for whether they bind to thrombin (a protease involved in blood clotting). Only 42 of the compounds showed a positive result. Contest entrants applied various learning methods, such as Support Vector Machines and Bayesian classifiers, to learn the discriminating patterns among the 139,351 binary features in this training set. Finally, they applied their trained classifiers to the 634 compounds in the competition test set. The entries were judged on how well their predictions matched actual laboratory tests.

Although this procedure suited the classification competition well, actual use in a pharmaceutical research laboratory would be somewhat different. Chemists would use the classifier's predictions to guide their efforts in the laboratory work, and if successful, would end up not testing all 634 compounds, but only those that were predicted positive, for example. Due to the natural limits of laboratory equipment and staffing, these tests would be carried out sequentially (though perhaps in small batches). Because each lab test costs money and time, it would make sense to test the most certain positive predictions first, and proceed down the ordered list. Once an adequate number of binding compounds is identified, further testing expense could be avoided.

Furthermore, since modern computers are fast enough to train a new classifier quickly, the results of each lab test could be incrementally included in the training of the classifier to further improve its precision before the next best predicted organic compound is selected. In fact, this incremental learn and search procedure could well be applied from the outset as soon as just one positive and one negative training example has been identified in laboratory testing, i.e. without the aid and associated cost of building the training set of 1,909 compounds.

This incremental re-training and prediction is the subject of this paper. We demonstrate that it is highly effective for the thrombin classification task. Indeed, it surpasses the performance of the best entry in the KDD Cup contest, *even without the benefit of the original training set*. This is partly due to its reduced exposure to concept drift between the training and test sets, as we discuss later. Finally, this incremental search method is surprisingly fast and scalable. Although a natural interest for computer scientists would be to improve the computation time for incrementally re-training the classifier, traditional batch learning methods yield ample performance. On a modest 733 MHz computer, it took less than a minute to retrain and obtain the next prediction, despite there being 139,351 features and despite spawning a separate Java classifier sub-process on each iteration, rather than using optimized compiled code. We conclude that this application is ready for practical use in lab settings and does not require incremental optimization research.

In the following section we lay out the procedure, and in the next section we describe experimental results on the 2001 KDD Cup thrombin classification task, demonstrating its viability. Following this we discuss related work and variations we evaluated.

2. Incremental Learn & Search Procedure

Initially, when the training set is empty, candidates may be chosen randomly until at least one positive and at least one negative training example has been confirmed in laboratory testing. We note that the chemist's knowledge comes into play in selecting the overall set of organic compounds to consider. Thus, even random selection already benefits from domain knowledge. Nonetheless, the chemists may optionally provide initial choices superior to random sampling.

The procedure at each step is as follows:

1. train a classifier with the available training data,
2. select the strongest positive prediction from among the untested candidates,
3. test it in the laboratory, and
4. add its result to the training set for future iterations.

If the laboratory setup more efficiently handles a batch of ten at a time, then the ten strongest predictions would be taken. We discuss variants later.

2.1 Classifier for the Thrombin Task

The choice of classifier is left open in the procedure above. For our experiments with the thrombin task [1], we assessed Naïve Bayes (NB), Nearest Neighbor (NN), and Support Vector Machines (SVM), the latter two having been found to perform well in high-dimensional (text-domain) spaces [7]. Feature selection is an important scalability consideration for these data, as there are 139,351 binary features (whose meanings remain undisclosed). At each iteration, we selected features via the well-known Information Gain metric, chosen for its good performance in high-dimensional (text-domain) spaces [8]. (In our pilot experiments, we considered other feature selection metrics, such as Chi-Squared, without finding improvement.)

We wrote the control software in Perl, chosen for its quick prototyping ability. On each iteration, it selects features, prepares new training files, and spawns a new Java sub-process running the WEKA [6] open-source implementation of Naïve Bayes, (k=1) Nearest Neighbor, or (linear kernel) Support Vector Machines, using default parameters. As a nod to scalability, the Perl program only loads the 170 MB thrombin dataset into memory once (taking almost three minutes), and computes the information gain scores for the many features incrementally, updating only those scores that are affected by each additional sparse feature vector.

3. Experimental Results

We performed our experiments on the official test set of the KDD Cup thrombin competition. Of the 634 compounds, 150 are active in binding. Given this 24% positive rate, according to the Geometric distribution, it takes on average 3.2=(1-p)/p random samples to yield an initial training set with at least one positive and at least one negative.[1]

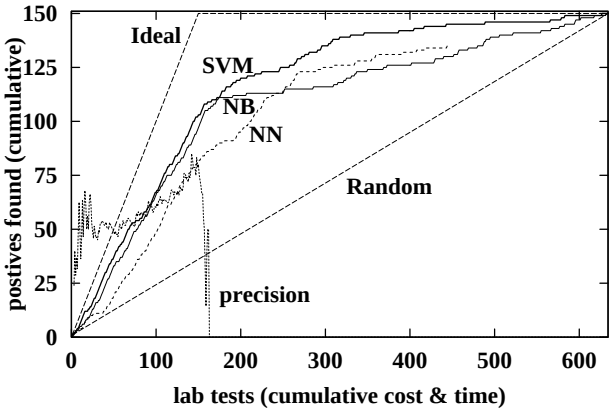

Figure 1: Number of positives found vs. number of iterations. 8000 features were selected at each iteration via Information Gain.[2] The 'precision' curve shows, for the Naïve Bayes classifier, the % correct positive predictions at each step.

The graph in Figure 1 plots the number of positives found over time—the "reward curve." Any procedure carried out to the end will finish with 150 positives found after all 634 lab tests. Since the cost of laboratory testing increases linearly with each iteration, the ideal reward curve would identify all the positives at the beginning, yielding a 45^{o} slope until the positives were exhausted—by contrast, random sampling yields on average a diagonal line with slope 150/634.

Actual curves for the incremental learn and search process lie between these two extremes. Once the

[1] For situations where positives are very rare, it may be worthwhile to attempt to beat the random odds by explicitly selecting compounds whose feature vectors differ most from the known negative training examples.

[2] Since the Nearest Neighbor classifier was an order of magnitude slower, its curve represents only 4000 features. It is excluded from the remainder of the paper, because its results at 4000 and 8000 features over many runs are consistently and substantially inferior.

minimal training set is established via random sampling (i.e. following the random curve for ~3 samples), the reward curve then shifts to using the classifier. Initially we expect the classifier to perform only a little better than random guessing, and then to improve as the training set fills out. To demonstrate this, Figure 1 also overlays, for the Naïve Bayes classifier, a graph of the precision, i.e. the percentage of positive predictions that are correct at each iteration. (Note that precision, not overall accuracy, is the appropriate goal for the classifier.) As expected, the classifier's performance begins ~24% precision—no better than random—and grows to 75% precision when the training set has ~150 samples. The good precision accounts for the steepness of the reward curve.

Later in the process, when most of the positive compounds have already been identified, the classification problem gets progressively harder to find the remaining few positives among the many remaining negative compounds. After 175 iterations, Naïve Bayes has identified 111 positives, and from that point onward the classifier fails to identify any positives—zero precision—and we see a sharp knee in the reward curve that then proceeds linearly to the top right—as good as random guessing for the final positives. SVM, perhaps because of its more powerful hypothesis space (lower bias), performed substantially better in this region until iteration ~330, at which point its performance is roughly linear (random) in identifying the last 10 positives. Nonetheless, this represents over 44% cost savings over random sampling to attain 140 positives—330 lab tests instead of 592.

It can be especially difficult to find the last remaining positives if they are scattered as noise in negative regions of the feature space, e.g. if they are actually mislabeled negatives in the training set due to noise or experimental error in the laboratory setting.

3.1 ROC Curves

To present these results in more familiar terms, we express them as ROC curves. The initial search begins at 0 true positives and 0 false positives, and climbs upward. Figure 2 presents four example ROC curves with varying parameter settings: the bold lines represent SVM and the thin lines represent Naïve Bayes; meanwhile the dotted lines represent 2000 features selected, and solid lines represent 8000 features selected. At 2000 features, the two classifiers perform comparably and have a rounded shape. At 8000 features, SVM performs substantially better than Naïve Bayes after the knee in its reward curve: the area under the ROC curve for SVM is 87%, while the area for Naïve Bayes is 79%.

For comparison, we overlay the ROC curve of the winning entry of the KDD competition, which in contrast trained from the 1,909 examples of the official training set. We hypothesize that its relatively poor performance on the testing set is due to concept drift between the training set and the test set [2]. While such situations are usually avoided by machine learning researchers, it is not uncommon in real-world industrial problems, where it can be difficult to obtain exactly the right training set for the intended usage of the classifier. Often training data are drawn from an earlier point in time or from a restricted subset of geographical samples, having a somewhat different distribution than encountered in actual use. Such concept drift can severely impair the performance of a deployed classifier.

Fortunately, the incremental predict and re-train paradigm has significantly less exposure to this risk, since it is continually learning on the very instances it is being used to classify. Note this only works because we obtain the true label after each prediction. Complete immunity from concept drift is impossible, however, e.g. if the pool of available instances is changed over time and takes on a different character from the earlier samples.

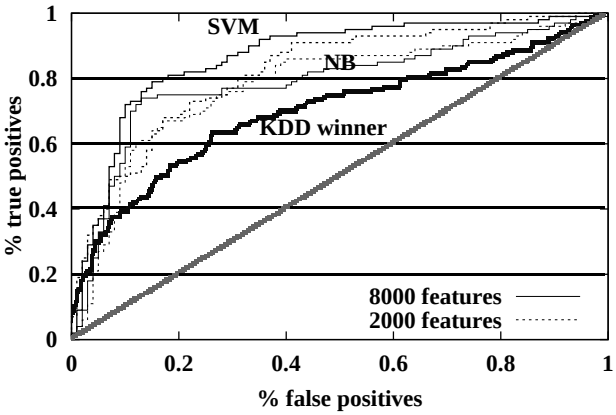

Figure 2: ROC curves generated incrementally by the search procedure for Naïve Bayes and Support Vector Machines using 2000 and 8000 features.

3.2 Average Waste Cost Analysis

The performance curves shown up to this point each represent a single run, which may display significant variation due to the initial random sampling. In determining the best choice of classifier and number of features to use, however, we need to consider an average over multiple runs. There remains the question of which scalar performance metric by which to judge. The average ROC area is one

candidate, but this measure is significantly influenced by the area covered by the latter portion of the curve, and our overall goal is to minimize lab testing costs, avoiding the latter portion of the curve. Thus, a more appropriate measure is the cost (number of iterations) to obtain a majority of the positives, say $2/3^{rds}$, i.e. 100 positives. The minimum cost would then be 100. In order to make the range of costs comparable across different dataset sizes with different numbers of positives, we normalize the cost by first subtracting the minimum cost possible, and then normalize by the number of negatives. This normalized "waste cost" is equivalent to the false positive rate on the ROC curve when $2/3^{rds}$ of the true positives have been obtained.

Using this metric as our guide for selecting the best performance, we average over 10 runs for various choices of classifier and number of features to select. The results are plotted in Figure 3. The number of features has a major effect, and the choice of classifier has a minor but non-negligible effect. The best of these choices was SVM with 8000 features, which accrued ~50 false positives to attain 100 true positives. Even at 4000 features where the difference in the curves appears small, a one-sided Wilcoxon rank-sum test confirms that SVM performance is statistically significantly better than Naïve Bayes at p=0.006.

Using this methodology, we evaluated a number of variations. We discuss some of these next.

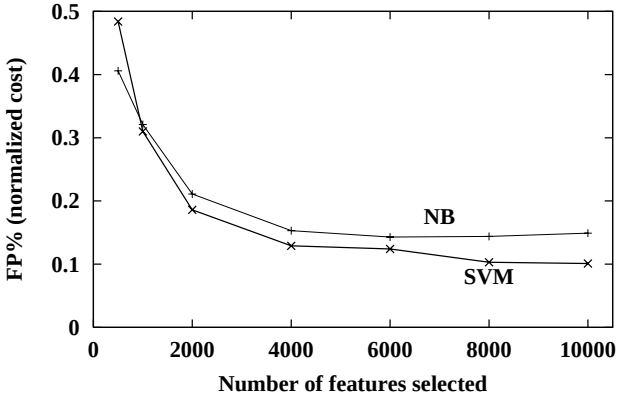

Figure 3: Average waste cost to obtain $2/3^{rds}$ of the positives (i.e. 100 positives), as we vary the classifier and number of features selected.

4. Discussion, Related Work & Variations

The sampling order of instances induced by the incremental search method will tend to reinforce local maxima encountered in the hypothesis space. As a Gedanken experiment, consider a 2-D space that consists of a sea of negatives with an island of positives and another distant, tiny island of positives. If the initial random samples identify only the large island, a generic classifier will concentrate on that island and only once it has been exhausted will it begin sampling outside the island. Toward the end of its search it will be forced to discover the distant, minor island of positives.

Note that *active learning* methods suffer the same blind spot. The goal of active learning is to select the training examples so as to quickly improve the classifier's overall accuracy. An active learner requests samples from the region of greatest uncertainty, as opposed to the present work, which draws from the region of greatest certainty of being positive. For the example above, active learning would extensively sample the edges of the large island, and if effective in its goal, would experience a hit rate of about 50%—a poor strategy for reducing laboratory costs—and would not discover the tiny island of positives until forced.

In order to discover isolated islands of positives, one needs a method that is willing to sample from predicted negative regions. One method, known as ε-*greedy* [5] in the field of reinforcement learning, replaces the greedy search behavior with random sampling on some iterations with probability ε. We applied this variation for ε = 1% and 5%, but performance only degraded, due to the additional random tests this method inserts into the search. (Note: in evaluating this and subsequent variations, we tested for a performance improvement at 90% (135) of the positives, because of greater sensitivity to sustained performance in this difficult region and because the variations are aimed at improving the latter portion of the search.)

We then evaluated a variation that randomly samples only when the best prediction is (strongly) predicted to be negative, i.e. when the classifier believes it has exhausted the positive regions. Evaluation showed this also to degrade performance for the thrombin task.

Rather than depend on the classifier's own estimation to determine when it is searching in a predominantly negative region, we assessed a variation that decides to use random sampling only when the most recent K samples have been negative, having observed long runs of negatives in practice. Once again, for the thrombin task, this showed only performance degradation for K=3 through 8.

One final variation we tested was to increase the initial random sampling period to 5, 10, or 15 samples, before beginning the use of the classifier. Neither did this improve performance.

Finally, we make a note about the computation workload. Each iteration typically took under 30

seconds. A typical simulation run up to 100 positives found took under 2.5 hours—insignificant compared to the laboratory time to perform the many tests. The iterations slow as the training set gets larger, but in a lab setting, as opposed to our simulation setting, the process would be terminated long before the end. Complete simulations for Naïve Bayes with 8000 features, for example, took almost 10 hours of compute time. Improvements would certainly be welcome for conducting simulation studies.

5. Conclusion

In this paper, just as in the 2001 KDD Cup, we promote the use of supervised machine learning to predict the organic compounds that are most likely to be active in binding in order to help guide the chemist's experiments to reduce costs and improve their yield. Unlike the KDD competition, we propose that the training set be gleaned from the testing set incrementally as it is tested, yielding a "predict and re-train" search process, as in reinforcement learning [5]. The superior results demonstrated for this method reflect its reduced exposure to the risk of concept drift between the training and testing sets. In addition, this eliminates the cost of gathering a separate training set.

Acknowledgments

We would like to express our thanks for the encouragement provided by Umesh Dayal and Jaap Suermondt, and for the statistics consulting that Hsiu-Khuern Tang willingly gave. We extend our appreciation to the WEKA project for their open source machine learning software [6] and to DuPont Pharmaceuticals Research Laboratories and the KDD Cup organizers for making the thrombin dataset available and providing new challenges. We also thank the Informatics and Distribution Laboratory for use of the ID/HP i-cluster [3].

References

[1] Cheng, J., Hatzis, C., Hayashi, H., Krogel, M.-A., Morishita, S., Page, D. & Sese, J. KDD Cup Report. ACM SIGKDD Explorations, 3(2):47-64, 2001.

[2] Forman, G. A Method for Discovering the Insignificance of One's Best Classifier and the Unlearnability of a Classification Task. In *Data Mining Lessons Learned Workshop, 19th International Conference on Machine Learning (ICML)*, Sydney, Australia, July 8-12, 2002. Hewlett-Packard Tech Report HPL-2002-123.

[3] Informatics and Distribution Laboratory, Cluster Computing Center, http://www-id.imag.fr/Grappes

[4] Haztis, C. & Page, D. KDD Cup 2001 Summary Presentation, at Knowledge Discovery in Databases Conference (KDD), August 26, 2001. Available at http://www.cs.wisc.edu/~dpage/kddcup2001

[5] Sutton, R.S., Barto, A.G. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998.

[6] Witten, I.H., Frank, E. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, October 1999.

[7] Yang, Y., Liu, X. A Re-examination of Text Categorization Methods. In *Proc. of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.

[8] Yang, Y., Pedersen, J.O. A Comparative Study on Feature Selection in Text Categorization. In *Proc. of the 14th International Conference on Machine Learning (ICML)*, 1997, pp.412-420.