Characterizing Model Performance in the Feature Space

Stephen D. Bay and Michael J. Pazzani
Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92697, USA
{sbay,pazzani}@ics.uci.edu

A fundamental problem in machine learning is understanding the conditions for which a learning algorithm works well. Understanding an algorithm's strengths and weaknesses and being able to compare two algorithms with each other are necessary for designers to develop (or select) learning algorithms for a specific problem. Generally, one can attempt to analyze and understand algorithms either theoretically or empirically.

Theoretical analyses of machine learning algorithms have usually resulted in weak performance guarantees that are not much use to a practitioner. Algorithms are typically proven to be *asymptotically consistent* (i.e., will achieve the Bayes optimal error rate given enough training examples) or that the algorithm can be used to *PAC* (Probably Approximately Correct) learn a given concept (Valiant, 1984). Another approach is to analyze average case behavior under specific distributional assumptions, such as learning m-of-n concepts (Langley & Sage, 1999). Although these analyses are useful in understanding the general behavior of an algorithm, they are unable to provide guidance to the designer in the form of specific predictions of an algorithm's performance with a given problem. Thus most researchers and practitioners resort to empirical evaluation to understand the interaction between learning algorithms and a domain.

Unfortunately, most evaluation methods give very little information to the designer. For example, the most common method of empirically evaluating a classifier is to examine its error, or more generally loss, and many comparisons of algorithms use only this metric. Loss can easily be estimated by using a test set or cross-validation. However because loss is a single number, it reveals little about the algorithm except gross performance on the domain.

Other researchers have tried to learn when a classification algorithm is appropriate for a problem domain based on characteristics of the data set. The basic idea is to use properties such as the number of features, number of classes, or the number of instances to learn by inspection or through automated analysis when an algorithm is appropriate. For example, Aha (1992) used a rule learner to automatically derive results such as the following.

> If (# training instances < 737) AND (# prototypes per class > 5.5) AND (# relevant attributes > 8.5) Then IB1 will perform better than C4

We take an orthogonal approach to evaluation where we try to automatically characterize model errors or model differences *in the feature space of the problem.* Specifically, we seek conjunctions of attributes and values that represent areas of good or poor performance (or agreement/disagreement between two models). For example, in the Adult Census domain (Blake & Merz, 1998) the goal is to predict if the salary of a person is greater or less than \$50,000 from demographic variables. When we evaluate models produced for this task, we would like to obtain rules such as:

> Classifier MC4 (Kohavi et al., 1997) is 21% less accurate than average on people who are between 45 and 55 years of age, are high school graduates, and are married. This represents 115 misclassified instances.

Alternatively, if we are comparing two classifiers on this task:

> MC4 and naive Bayes are 9% less likely to agree than average on people who have Masters degrees and are married. This represents 50 instances with different predictions.

Knowledge of model errors or differences could be used in many ways to help guide the process of constructing a machine learning system. For example, it could be used to identify areas in the feature space where the model performs poorly. Given this knowledge, the designer has many options. She could flag examples in those areas for exception and have a human handle them, construct a special purpose model just for those examples, or attempt to improve the model by collecting more data (features or examples).

The task of finding these rules can be framed as a meta-learning problem (Wolpert, 1992). The basic approach we use is to augment a test set of data with information concerning whether or not the classifier made a correct prediction, or alternatively, if the two models differed from each other. We then use an algorithm, such as C5, to determine the conditions when errors or differences occur.

The main difference between our work and past meta-learning approaches is that our focus is on generating *comprehensible descriptions* of model errors or differences. In contrast, work by Merz (1995) and others has concentrated on selecting the best classifier for an instance based on its location in the feature space with the goal of maximizing predictive accuracy.

We implemented this simple framework (Bay & Pazzani, 2000) and tried two different algorithms as meta-learners: We used C5, a standard decision tree algorithm for classification, and we compared it with STUCCO, a contrast-set miner (Bay & Pazzani, 1999). These two algorithms differ in three important ways: First, C5 is discriminative algorithm whereas STUCCO is a characteristic or informative approach (Rubinstein & Hastie, 1997)[1]. Second, C5 is incomplete as it uses heuristic search while STUCCO is complete. Finally, C5 produces an unordered set of rules whereas STUCCO summarizes the rule set hierarchically.

In our experiments, C5 and STUCCO produced very different rule sets when used as the meta-learner. We found that C5 tended to fragment the data and produced many rules affecting only a small number of examples. It was unstable and small variations in the data set resulted in wildly different rule sets being induced. Finally, C5 tended to produce longer rules. STUCCO did not suffer from these problems.

We direct the reader to Bay and Pazzani (2000) for a more detailed description and further discussion of our work.

Acknowledgments

This research was funded in part by the National Science Foundation grant IRI-9713990.

References

Aha, D. W. (1992). Generalizing from case studies: A case study. *Proceedings of the Ninth International Conference on Machine Learning* (pp. 1–10).

Bay, S. D., & Pazzani, M. J. (1999). Detecting change in categorical data: Mining contrast sets. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 302–306).

Bay, S. D., & Pazzani, M. J. (2000). Characterizing model errors and differences. *Proceedings of the Seventeenth International Conference on Machine Learning.*

Blake, C., & Merz, C. J. (1998). UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Kohavi, R., Sommerfield, D., & Dougherty, J. (1997). Data mining using MLC++, a machine learning library in C++. *Journal of Artificial Intelligence Tools, 6*, 537–566.

Langley, P., & Sage, S. (1999). Tractable average-case analysis of naive bayesian classifiers. *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 220–228).

Merz, C. J. (1995). Dynamical selection of learning algorithms. In D. Fisher and H. J. Lenz (Eds.), *Learning from data: Artificial intelligence and statistics.* Springer Verlag.

Rubinstein, Y. D., & Hastie, T. (1997). Discriminative vs informative learning. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 49–53).

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM, 27*, 1134–1142.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*, 241–259.

[1] Discriminative algorithms try to learn a decision boundary without examining the class densities (i.e., focus on $P(y|\mathbf{X})$ where $\mathbf{X}$ is a region of the feature space and y is the class variable). Characteristic algorithms learn a description for each class (i.e., focus on $P(\mathbf{X}|y)$).