

COMMENT ON “SEPARATING SIGNAL FROM BACKGROUND USING ENSEMBLES OF RULES”

HARRISON B. PROSPER

Florida State University, Tallahassee, Florida, USA

Friedman and Popescu have introduced a novel machine learning algorithm, based on rules, that performs very well relative to other methods. We make a few general remarks about ensemble methods and comment on their particular method.

1. Introduction

It is often instructive to view things from a general perspective. This is true, in particular, of machine learning algorithms, if only because a general perspective clarifies the relationship between different algorithms, and, makes it easier to discern whether or not a “new” method is truly new. By embedding algorithms, such as AdaBoost ¹, Bagging and Random Forests ², into the framework of *ensemble learning*, Friedman and Popescu ³ have clarified the nature of these algorithms. Moreover, having understood that boosting and bagging are “merely” interesting variations on a theme, many other interesting variations spring to mind, for example, the one described by Jerry Friedman at this meeting.

2. Ensemble Learning

The idea of ensemble learning is to construct a mapping $y = F(x)$, based on some training data $\{(x_1, y_1), \dots, (x_n, y_n)\}$, where

$$F(x) = a_0 + \sum_{m=1}^M a_m f_m(x),$$

and $\{f_m(x)\}$ is an *ensemble* of functions called *base learners*. The base learners are chosen from a function class, each of whose elements is labeled by the values of a set of parameters $p = (p_1, p_2, \dots)$. Given a class of parameterized functions $f(x; p)$, an algorithmic procedure is specified to pick functions from the class, typically, by minimizing some *loss function* L . The ensemble method is very general: any function class may be used, along with any mechanism to choose from it.

As noted by Jerry Friedman, one anticipates that different procedures might be required for different data sets. The rule-based method of Friedman and Popescu, however, seems broadly applicable.

3. RuleFit

In the *RuleFit* method ³, the function class used is the class of all conjunctions

$$f_m(x) \equiv r_m(x) = A \& B \& C \dots,$$

where A , B , etc., are simple statements with truth value 0 or 1. The function $r_m(x)$ is called a *rule*. For example, $A = E_T > 25 \text{ GeV}/c$ and $B = 85 < M_{ee} < 95 \text{ GeV}/c^2$ might be statements typical of a signal/background classification problem in particle physics, in which one requires a jet of particles to have transverse energy exceeding 25 GeV *AND* the mass of an electron-positron pair be consistent with that of the Z boson. In practice, the class of rules is defined by the leaves of a forest of decision trees. One arrives at a leaf, appropriately, by following branches, starting at the root. Having found a set of rules, the coefficients a_m are found by minimizing the *lasso* loss function, a principal virtue of which is that, at its minimum, a large fraction of the coefficients are typically zero. Therefore, the number of rules that remain is generally far fewer than the number in the original set.

The RuleFit method has been shown to perform very well. Moreover, since its function class is based on decision trees, the method is fast. Its other advantage is that the meaning of each function $f_m(x)$, being a simple rule, is readily apparent. That being said, the function $F(x)$ is a linear sum of rules; therefore, even though each rule is easy to understand, it is less clear that $F(x)$ itself can be as readily interpreted.

An important benefit of the RuleFit method is that it provides a way to assess the importance of a variable. This is extremely useful because one can rank variables according to their importance and keep only those that are judged significantly more important than the rest. Thus can one reduce the di-

mensionality of the problem, and therefore the computational burden. The importance measure suggested by Friedman and Popescu is an intuitively plausible one that might be expected to work well most of the time. It is not clear, however, that it will *always* rank variables the same way as would a method in which all possible combinations of variables were tried and the subsets ranked accordingly.

4. Let a Thousand Flowers Bloom

The generality of the ensemble method invites the exploration of potentially interesting variations on that theme. One possibility, might be to use a function class defined by

$$f(x; p) = \tanh\left(p_0 + \sum_{i=1}^N p_i x_i\right),$$

together with any one of the standard loss functions. Given the ensemble of functions $f_m(x) = \tanh(p_{m0} + \sum_{i=1}^N p_{mi} x_i)$, and coefficients a_m – obtained, perhaps, by lasso regression – one would then

have

$$F(x) = a_0 + \sum_{m=1}^M a_m \tanh\left(p_{m0} + \sum_{i=1}^N p_{mi} x_i\right).$$

This function is typical of the kind that appears in a neural network with N inputs, a single hidden layer of M nodes and a single output.

Acknowledgments

I would like to thank Jerry Friedman for an enlightening conversation and for providing a timely copy of his slides.

References

1. Y. Freund and R. E. Schapire, *Journal of Computer and System Sciences* **55**, 119 (1997).
2. Wald Lecture, Leo Breiman, <http://www.stat.berkeley.edu/users/breiman/wald2002-1.pdf>.
3. Jerome H. Friedman, these proceedings.