A Comparison of Machine Learning Techniques for Classification of the Higgs Boson

Jared Moskowitz & Julia Rowe
(Dated: 8 May 2015)

I. INTRODUCTION

One of the integral tasks of particle physics experiments is identifying elementary particles produced in high energy collisions. This can be especially problematic for the Higgs boson, given that the primary observed decay mode, W-W+, results in the production of quark jets, which are difficult to distinguish from the background. To help with this, supervised machine learning techniques are used for classification. These classification methods depend on the features of the particles predicted by the standard model. Very large and noisy data is being dealt with, so speed and accuracy is of the utmost concern. The purpose of this analysis is to compare machine learning algorithms, looking at both the accuracy and how they scale with the number of features.

II. METHODS

Various algorithms from the Waikato Environment for Knowledge Analysis (WEKA) suite of machine learning software were tested under the same conditions. This suite was used because it is preinstalled on the Cluster super computer (the environment for running these tests) and seems to be popular in the data science community. Learners were trained using Monte Carlo data and tested on real data from ATLAS. Data was separated into four groups: even runs with nine features, odd runs with nine features, even runs with thirty-one features, and odd runs with thirty-one features. These even and odd groups come from splitting our original training and test data in half so that we could run each algorithm twice on disjoint sets. Predictive machine models were built from each of these groups and were used to test the opposite group of runs with the same number of features.

A. The Algorithms

1. Decision Trees (J48)

Decision trees recursively split on values of features based with priority of splits to those with the most information gain. This information gain can more specifically be represented with the equation:

$$\text{Gain(split)} = \text{Ent}(S) - \sum_j \frac{|S_j|}{S}\text{Ent}(S_j) \qquad (1)$$

This equation corresponds to a change in entropy (metric of information) from one state to another. This is the change in entropy from Nodes represent decisions (make next decision on left child if energy greater that 126 GeV otherwise go right). The leaves represent binary classifications (signal or background). WEKA provides the c4.5 variation of this algorithm.

2. Random Forest

Random forests are made up of multiple decision trees using random feature selection and bootstrap aggregation to correct for the common problem of decision tree overfitting. This algorithm creates multiple decision trees by selecting a smaller subset of the features and building trees. Our random forests created ten trees. For runs with nine features each tree selected four of these features and for runs with thirty-one features six features were selected.

3. Naïve Bayes

This algorithm creates simple probabilistic classifiers by applying Bayes' rule under the assumption of statistical independence between the features. Although highly simplified, the Naï"ve Bayes algorithm has been successful in highly complex situations. It is also highly scalable, making it an efficient algorithm.

4. Support Vector Machine

Support vector machines builds a hyperplane that best separate the two categories of data. This hyperplane is defined by a subset of the training data called support vectors. The goal is to maximize the margin, which is the euclidean distance from a support vector to the hyperplane. An advantage of SVMs is it uses a fixed number of support vectors from the sample, avoiding overfitting. Often the data is projected into higher dimensional feature space, if the set is not separable in the given dimension. The hyperplane is defined by the set of points $\mathbf{x}$ which satisfy the equation:

$$\mathbf{w} \cdot \mathbf{x} - b = 0 \qquad (2)$$

where $\mathbf{w}$ is the vector normal to the hyperplane. Anything above the hyperplane should have the label of 1 (signal) and anything below -1 (background).

III. RESULTS

TABLE I. Accuracy of Cross Validation on Training Data

Algorithm	9 Even	9 Odd	31 Even	31 Odd
Decision Tree	88.41 %	87.98%	87.70%	87.75%
Random Forest	88.72%	88.57%	88.82%	88.53%
Naïve Bayes*	87.52%	87.79%	84.94%	84.97%
SVM - linear	87.07%	87.01%	87.33%	87.26%

TABLE II. Area Under the ROC Curve

Algorithm	9 Even	9 Odd	31 Even	31 Odd
Decision Tree	0.907	0.904	0.855	0.861
Random Forest	0.929	0.928	0.929	0.929
Naïve Bayes*	0.931	0.929	0.915	0.913
SVM - linear	0.792	0.794	0.803	0.803

*Naïve Bayes did not run cross-validation, so the even machines were tested on the odd training data and vice versa.

TABLE III. Time (in seconds) per Algorithm

Algorithm	9 Even	9 Odd	31 Even	31 Odd
Decision Tree	10.056	8.605	26.672	29.490
Random Forest	21.346	23.487	37.249	38.542
Naïve Bayes	1.381	1.687	2.989	3.099
SVM - linear	17.110	17.572	86.40	82.086

IV. CONCLUSION