APPLICATIONS OF BOOSTING ALGORITHMS TO PREDICT PATIENT DISEASE OUTCOMES

Annie Lane, Alejandro Eguren

Dec 14, 2015

Boston University

Department of Electrical and Computer Engineering

BOSTON UNIVERSITY

APPLICATIONS OF BOOSTING ALGORITHMS TO PREDICT PATIENT DISEASE OUTCOMES

Annie Lane, Alejandro Eguren

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

Dec 14, 2015

Contents

1. Introduction

Data modeling and machine learning can play an important role in the field of public health to help reduce medical costs and improve population health. Currently, data sets exist that look at disease outcomes and diagnosis, noting patient demographics and vital signs. The issue is that these data sets are rare, sparse, and difficult to collect, many of which have very dynamic properties per disease and individual patient. Common machine learning techniques may fail to capture the nature of these data sets. But Adaptive Boosting, or AdaBoost, is a type of Boosting classifier that might be able to model such data sets. This issue is important because capturing these dynamic public health data sets will allow society to model disease trends and provide personalized therapies. In the long run machine learning can go a long way in saving inventory costs, decreasing hospital time, and increase patient health.

2. Literature Review

AdaBoost is an algorithm proposed by Freund and Schapire (1999) and is a popular instance of Boosting, one of the most common ensemble methods. Boosting creates a strong learning by iteratively combining weak learners, which can be considered "rough rules of thumb" (Schapire 2003). Weak learners are moderately inaccurate but perform better than random guessing. A good weak learner is simple to calculate. Simplicity both reduces computational complexity and enables non-experts to understand the weak learners, which serve as building blocks for the final strong classifier. [1][2]

The AdaBoost algorithm, or Adaptive Boosting, gains its name from the two sets of weights that are tuned during the iterations of the algorithm. The first set of weights are the n sample weights w, one weight for each of the training examples. These weights w reflect the "hardness" of an example and provide more emphasis to more difficult examples on future iterations. The second set of weights, α, is assigned to each of the T weak classifiers that comprise the final strong classifier. These weights reflect the value and accuracy of the weak

decision rule. The final strong learner is a linear combination of each weak learner applied to a test example times the associated α weight.

The Diabetes data set (Beata Strack et al) originated in a study that attempts to demonstrate the significant bearing that Hyperglycemia has on a diabetic patients' mortality and morbidity. Having collected thousands of patient data and features the study conducts a multi-variable logistic regression to try and find a statistically significant connection with a feature Hba1c (Glycated Hemoglobin). The problem is that this feature was only collected 18% of the time and the data set, collected over 10 years, was very sparse. The study concludes that there is a significant connection between feature Hba1c and the hospital readmission rates to patients which had this feature present. The predicting readmission is an outstanding issue for the majority of the patients in the data set due to the data set sparseness and the absence of this key feature.[3] [4]

3. Problem Statement

Although machine learning holds great promise in classifying and regressing public health data sets, challenges also exist. These data sets to begin with are very sparse; this is because every time a patient goes to the doctor different vitals are measured and different tests are orders. Across several patients with the possibilities of hundreds of tests this can result in many features per patient missing as a whole. Some of these features may be inexpensive to obtain, such as age and race, while others are more costly, such as blood glucose levels. This is also a contributing factor to the difficulty in collecting some data sets depending on the disease and how expensive the diagnosis can result. Another challenge to note is that most public health data sets are rare, with every patient requiring hours to collect features, it may take years before a working set may be developed. To overcome these major challenges, we must analyze how current medical frame works function to design an adaptive algorithm that follows similar diagnosis techniques to hopefully yield valuable results.

To reach a diagnosis there are two main methods a doctor may use. The first is a Rules of Thumb concept and the second is to think of patient data in the form of sets of logical yes now questions. The Rules of Thumb approach applies when doctors look at patient vitals one by one and test to see if the recorded values are above a certain threshold. If so this feature is added to a cause and effect list and after numerous iterations of this process a diagnosis if

formed with some features contributing more then others. For example, if a patient comes in with an elevated temperature, high glucose, and over weight, all these weighted accordingly in their diagnosis since they are all above a standard thresh hold.

The second method utilizes a path of logical questions. Similar to a decision tree a diagnosis is preformed by starting with a seed symptom and tracing a path of existing/non-existing symptoms until a leaf is reached in which a diagnosis is decided. This is a series of simple yes or no questions that a doctor can easily follow and help make an educated decision. For example: does the patient have a fever? Yes, now looking at the blood glucose, are the levels higher then normal? Yes, now looking at the patients with is there an overweight issue? Yes. At this point in the diagnosis, diabetes seems a likely result. There's a subtle difference in these two medical frame works but they can be used in choosing which will be the best preforming algorithms for medical data sets. The first specifically relates to adaptive boosting algorithms, AdaBoost, that combine weak classifiers to adaptively make a strong classifier. And the second, a tree classifier that will cascade nodes down through a series of logical questions.

4. Implementation

4.1 Processing the Thyroid Data Set

The first data set, to be known as the Thyroid data set, is 2800 training examples and 972 testing examples from patients in Sydney, Australia. [5] The patients are classified as having Hypothyroidism (+1) or Negative (-1).

There are a mix of categorical, Booleans, and numerical features in the 27 feature data set. The categorical values, typically given as strings, are mapped to numerical values. All mappings are described in the data set's ReadMe file. The true and false values are mapped to 1 and 0 respectively. The numerical features are left untouched.

Missing values in the original data set are marked as question marks. To handle these missing attributes, all question marks are set to -1.

4.2 Processing the Diabetes Data Set

The second data set, to be known as the Diabetes data set, is composed of patient samples about their remittance status to American hospitals. Each sample is classified as

either “Not Re-admitted”, “<30”, “>30”, meaning the patient was never re-admitted, was readmitted in less than 30 days or was readmitted after 30 days, respectively. The positive class (+1) is those readmitted within 30 days. The negative class (-1) is composed of the remaining examples, approximately 85% of the total data set.

Categorical features were given numerical values. Age ranges were set to the upper limit, “0-10” became 1, “10-20” became 2. Any NaN values were assigned a numerical value or -1. More details are included in the READ-ME file with the diabetes set.

The data set is separated into a training set and a testing set. The training set contains 25% of the samples and the test set contains the remaining 75% of the data.

4.3 AdaBoost

For the purpose of this project, AdaBoost is implemented in a binary manner. The formulation of AdaBoost training has three main stages: (1) Initialization, (2) Iterative training of T weak classifiers, and (3) Output of the model. First, the n sample weights w are uniformly initialized to equal 1/n. In stage two, a for loop trains the 1 to T weak classifiers $h_t(x)$ and calculates the associated classifier weights α_t in cascade. In each iteration, the weak learner $h_t(x)$ is calculated. Then the weak learner $h_t(x)$, is applied to the training feature matrix to make the weak learner’s prediction. The weighted error ε_t is calculated based on Equation 4.1 and is used in Equation 4.2 to calculate the classifier’s associated weight α_t to store in the model. The final step in each iterations is to recalculate each example’s w based on the accuracy of $h_t(x)$; if the cascade of weak learners and weights so far misclassified example j, then w_j is increased so that future iterations will place more emphasis on correctly classifying example x_j. Finally, the Strong Learner $H(x)$ is output.

Equation 4.1 - Weighted Error ε_t

$$\epsilon_t = \sum_{i=1}^{n} w_i \, \delta[\, y_i \neq h_t(x_i)]$$

Equation 4.2 - Weak Learner’s associated weight α_t

$$\alpha_t = \frac{1}{2} log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

In this implementation, a weighted threshold classifier is used as the weak learner. This effectively models the simple “Rules of Thumb”. It is the best threshold on a single feature

column to minimize the weighted error, ε_t, the sum of weights w for misclassified examples**.** This weak classifier requires three inputs: (1) the n by d feature matrix X, (2) the n-long vector of true class labels Y, and (3) the most up-to-date n sample weights w. An exhaustive search calculates the weighted error, using equation 4.1, when a single feature is used to separate the binary data along a threshold. The feature-threshold pair that minimizes the weighted error is selected and stored as parameters for the weak classifier model.

AdaBoost classifies an example by taking the sign of $f(x)$ on the feature vector x. The function $f(x)$ is the sum from 1 to T of weak classifiers, $h_t(x)$ applied to the feature vector, scaled by their respective α_t value, as written in equation 4.3. The function is effectively a linear combination of the T α_t and $h_t(x)$ pairs.

Equation 4.3 - AdaBoost Decision Rule

$$H(x) = \text{sign}(f(x)) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

The AdaBoost algorithm was implemented in Matlab. It was adapted from Dirk-Jan Kroon's 2010 code, published on Matlab Central File Exchange. [6] The original code utilizes a single function *adaboost* with two different modes as possible input strings, *train* and *apply*. Some variables and structure were modified to reflect the project's formulation of AdaBoost. For the purpose of this project, code into two separate functions, *train_adaboost* and *test_adaboost.* Kroon's original code also includes a *WeightedThresholdClassifier* and *ApplyClassThreshold* function, both of which are used with minimal modifications.

For *train_adaboost*, there are three inputs: (1) *feature_matrix*, a matrix of n row vectors with the length of the number of features per example, (2) *class_labels*, a column vector with the true labels of the n examples, and (3) *num_T*, the desired number of iterations or weak classifiers T. The training function also has two outputs: (1) *model*, a struct with eight fields and T sets of those fields from each iteration, and (2) *final_predict,* the predicted class of the Strong Learner with T weak learners. The eight fields of the *model* struct are: (1) *alpha*, the α_t value, (2) *feature*, the feature selected by the weak classifier, (3) *threshold*, the threshold value used to split the data, (4) *direction*, which side of the threshold corresponds to the positive examples, (5) *boundary*, the range of each d feature in the training set, (6) *error*, the total classification error rate of the cascaded weak learners 1 to t, (7) *poserror,* the classification error rate of true positive examples, and (8) *negerror*, the classification error

rate of true negative examples. Fields seven and eight, positive and negative error rates, were added into the code for the purpose of this project.

For *test_adaboost*, some variables were also changed to reflect the project's formulation of AdaBoost calculations with the Strong Classifier in Equation 4.3. There are two inputs (1) *feature_matrix*, a matrix of row vectors with the length of the number of features per testing example, and (2) *model*, a struct described as an output of *train_adaboost*.

For all training, *T* is the only parameter that needs to be tuned. For all experiments, *T* is set to fifty. The AdaBoost model allows data to be tested with any number sequential weak classifiers less than or equal to *T*. The test data was then classified using the AdaBoost model.

4.4 Decision Tree Classifier

When implementing Decision Tree Classifier, used the MatLab built in functions *fitctree* and *predict*. Depending on the data set, the maximum height of the tree was set to 6 levels by defining the minimum number of examples per node and the maximum number of total nodes. The tree is trained on the training set and then later tested with the testing set. This is a well-studied classifier used in this study as a base classifier for comparison.

4.5 Support Vector Machines (SVM)

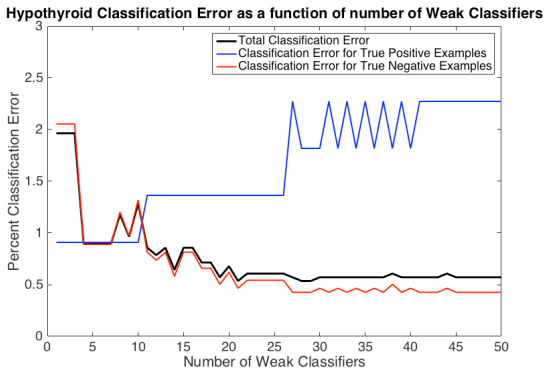
When implementing Support Vector Machines, we used the MatLab built in functions *fitcsvm* and *predict*. We utilized the default settings for the classifier, specifically the linear kernel. The SVM classifier is trained on the training set and then later tested with the testing set. This is a well-studied classifier used in this study as a base classifier for comparison.

5. Experimental Results

5.1 Thyroid Data Set Results

The Thyroid data set was classified with the positive (Hypothyroidism) and negative classes using three main algorithms: AdaBoost, decision tree, and linear kernel SVM. Performance metrics such as Correct Classification Rate (CCR) and Area Under the Curve (AUC) for the ROC results are used to compare the results.

The focus of this report is AdaBoost's application to this problem. Figure T1 shows how the total error, in black, decreases as the number of weak classifiers increases. However, as the number of weak classifiers increases to more than 10, the classification error for the originally true positive examples increases. There are significantly less positive examples, so the error rate is more sensitive to the number of misclassified examples. From AdaBoost's perspective, it is most beneficial to decrease the total error. For comparison purposes, it is worthwhile to show the performance of AdaBoost for T = 5 and for T = 50.



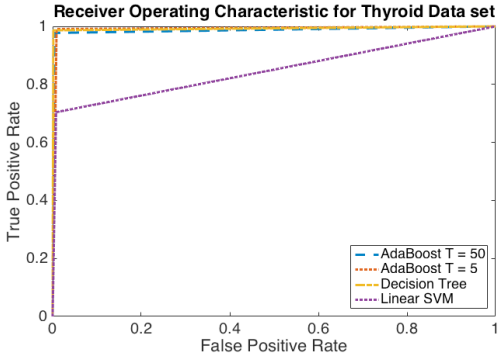


Figure T1 (left): The classification error of the Hypothyroid data as a function of the number of weak classifiers from 1 to 50.
Figure T2 (right): The ROCs for two versions of AdaBoost, a Decision Tree and Linear SVM on the Thyroid data set.

Table T3: Shows two performance metrics on four different classifiers trained and then tested to classify Hypothyroidism.

	AdaBoost (T = 50)	**AdaBoost (T = 5)**	**Decision Tree**	**Linear SVM**
CCR	0.9943	0.9911	0.9975	0.9712
AUC	0.9865	0.9910	0.9924	0.8482

The features used by AdaBoost provide insight into the most valuable features in classifying Hypothyroidism. The first six thresholds are on TSH (Thyroid Stimulating Hormone), TSH, FTI (Free Thyroxine Index), "on thyroxine", TSH, and TT4 (Total Thyroxine). Clearly, Thyroxine and TSH are the most important factors in diagnosing Hypothyrodism, which is consistent with medical literature.

The performance of the 4 algorithms are compared in Figure T2 and Table T3. Decision Trees perform the best by both metrics, but the performance is comparable to both

implementations of AdaBoost. Linear SVM performs the worst by both metrics, particularly by AUC. Linear SVM attempts to separate the data on all features while the other algorithms effectively do feature selection on the most relevant features.

5.2 Diabetes Data Set Results

The Diabetes data set was classified using the AdaBoost implementation, a decision tree with a minimum of 20 instances per node and a maximum of 100 total nodes, and a linear kernel SVM classifier. Metrics such as Correct Classification Rate (CCR), Precision, and the Area Under the Curve (AUC) for the ROC results are shown below:

Table D1: CCR, Precision, and AUC results from testing the three different classifiers. It is important to note that AdaBoost out performed SVM and Decision tree in these three metrics.

	AdaBoost (T = 50)	**Decision Tree**	**Linear SVM**
CCR	0.8903	0.8844	0.6357
Precision	0.5664	0.2941	0.1220
AUC	0.7286	0.5935	0.5092

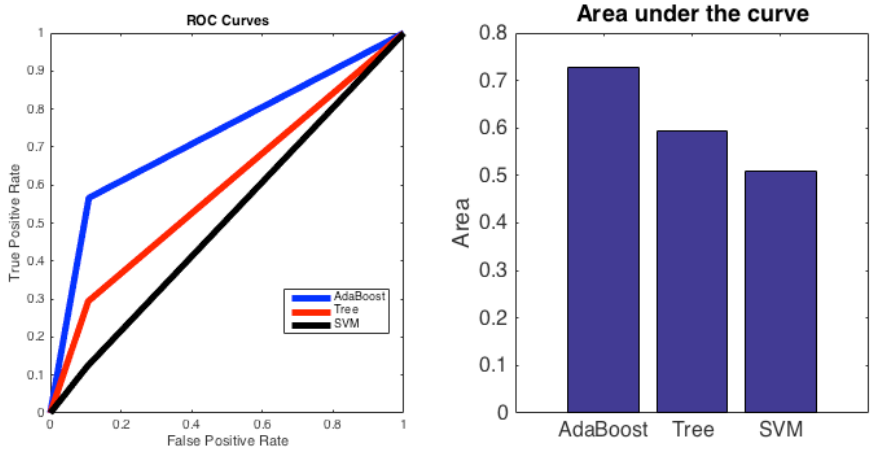


Figure D1: ROC curves of the different classifiers tested. AdaBoost surpasses the both the tree and SVM classifiers.

As a whole we can observe that the AdaBoost Implementation outperformed our other two classifiers. Both AdaBoost and the decision tree did very well in correctly pedicting classes overall in which SVM fell short. This helps support out notion that classifiers have to

be modeled under the framework of the field they intend to be applied to. Precision was another important metric where AdaBoost did really well. This is an important metric since it calculates what percentage of Hospital Readmissions are we predicting correctly. Finally we look at the ROC curves. These ROC curves are a function of the classifiers Sensitivity vs. Fallout. After conducting all these experiments we can see that the results fall in line with our original goal, which was to implement AdaBoost, and test our hypothesis that it's adaptive algorithm will prove successful in public health data sets.

6. Conclusions

For the Thyroid data set, it is interesting to compare the CCR and AUC for AdaBoost with T = 50 and T = 5. Table T3 shows that while the CCR for T = 50 is better than for T = 5 (0.9943 vs. 0.9911), the AUC for T = 50 is less than for T = 5 (0.9865 vs. 0.9910). This demonstrates that the performance metric selection is important; it reflects the desired classification. If the application is only interested in the total classification error, then CCR is a more favorable metric. However, AUC may be more favorable if the correct classification of positive examples is more important.

Looking at the experimental results for the Diabetes data set, we can conclude that AdaBoost and Tree classifiers did quite well in classifying hospital re-admissions for the diabetes data set. We can see in Figures D1, D2, and D3 that the CCR, Precision and ROC curves demonstrate the powerful capabilities of the adaptive boosting implementation. This goes to show that whilst SVM takes into account all features, AdaBoost focuses on the most important features and dynamically weights each of its weak classifiers. This follows our notion that medicine work under a Rule of Thumb concept and that classifiers should be modeled as such. Taking a look at our second notion, a path of logical questions, the decision tree also preformed quite well, with a high CCR values and moderate precision and ROC results. This result emphasizes the importance of carefully before selecting which classifiers to apply.

As a whole AdaBoost provides a powerful framework to work under by combining weak classifiers on individual features. AdaBoost is also inherently very useful because it is quick and easy to implement; the only true parameter to set is T which controls how many of these weak learners will iterate through the data set. This algorithm can also be generalized to many

different data sets by modifying the weak learner being used which makes it even more adaptable. Further improvements on AdaBoost will be to test different combinations of weak learners in an attempt to reduce our false positive and false negative rates in hopes to maximize the precision and ROC results.

Public health information and data sets offer many opportunities for machine learning to identify possible diseases, plan inventory, and help prevent diseases rather then curing them. Whilst there are many benefits to obtain from applying these concepts it is important to always consider the implications of a false positives and false negatives in the medical field. A false negative could allow a terminal disease to go unnoticed in a patient. A false positive could have big social implications to an otherwise healthy patient, for example an HIV diagnosis. Classification, Regression, and Clustering, amongst other tools, should always be carefully studied and rigorously tested to suit well with these medical needs.

7. Description of Individual Effort

Annie Lane primarily focused on investigating and implementing the AdaBoost algorithm for the thyroid and diabetes data sets as well as processing the thyroid data set, applying algorithms and generating results.

Alejandro Eguren primarily focused on preparing and gathering results from the diabetes data sets as well as defining the problem statement and algorithmic comparisons.

Both members worked together to prepare the presentation, write this final report, and discuss the conclusions.

8. References

[1] Freund, Yoav, and Robert E. Schapire. "A Short Introduction to Boosting." *Journal of Japanese Society for Artificial Intelligence* 14.5 (1999): 771-80.

[2] Schapire, Robert E. "The Boosting Approach to Machine Learning: An Overview." *Nonlinear Estimation and Classification Lecture Notes in Statistics* (2003): 149-71.

[3] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, "Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records," BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.

[4] Clore, John, Krzysztof J. Cios, Jon DeShazo, and Beata Strack. "Diabetes 130-US Hospitals for Years 1999-2008 Data Set." *UCI Machine Learning Repository*. 03 May 2014. <http://archive.ics.uci.edu/ml/datasets/Diabetes+130-US+hospitals+for+years+1999-2008>.

[5] Quinlan, Ross. "Thyroid Disease Data Set." UCI Machine Learning Repository, 01 Jan. 1987. <https://archive.ics.uci.edu/ml/datasets/Thyroid+Disease>.

[6] Kroon, Dirk-Jan. *Classic AdaBoost Classifier*. N.p.: Matlab Centra, 20 Jan. 2010. Matlab Source Code.

9. Appendix

Matlab source code developed for this project can be downloaded from the EC500 B1 Blackboard Site.