Numerical Methods For Estimation
6.867 HW 1

Anonymous

September 28, 2016

1 Gradient Descent

Boosting algorithms have their roots in the theory of *Probably Approximately Correct (PAC)* learning, introduced by Valiant [citation needed] in 1984. In the PAC learning model, the successful learning of an unknown concept occurs when we can obtain, with high *probability*, a hypothesis which *approximates* that concept class well.

More formally, we introduce the following definitions:

Definition 1.1 *A concept class, $\mathcal{H}$, is a set X along with a set of functions $\{f : X \to \{\pm 1\}\}$*

Definition 1.2 *An algorithm PAC learns a concept class $\mathcal{H}$, if, for any hidden distribution D on X, for any $h \in \mathcal{H}$ and for $\epsilon > 0, \delta > 0$, it produces a hypothesis whole error is at most ϵ, with probability $1 - \delta$.*

Within the PAC framework there exists two further concepts which is at the core of the theory behind boosting. These are the ideas of *strong learning* and *weak learning*, which are defined below:

Suppose the algorithm is given access to m labelled examples, $(x_i, f(x_i))$, where $x_i \in X$ is drawn according the the distribution D.

Definition 1.3 *To achieve strong learning, the goals is to output a hypothesis $h : X \to \{\pm 1\}$ which satisfies with probability $1 - \delta$ the inequality*

$$\mathbb{P}_{x \in D}[h(x) \neq f(x)] < \epsilon$$

Definition 1.4 *To achieve weak learning, we only require that the algorithm has accuracy which slightly outperforms random guessing by some edge η. That is*

$$\mathbb{P}_{x \in D}[h(x) \neq f(x)] < 0.5 - \eta$$

In the PAC framework, an algorithm which achieves strong learning is called a *strong learner*, whereas an algorithm which achieves weak learning is called a *weak learner*.

In 1990, Schapire[http://www.cs.princeton.edu/ schapire/papers/strengthofweak.pdf] showed that, in fact, the notions of weak learning and strong learning are equivalent. This formed the theoretical

basis for the boosting algorithm, Adaboost, which, given access to weak learners, produces a strong learner. Proposed by Schapire and Fruend [http://www.site.uottawa.ca/ stan/csi5387/boost-tut-ppr.pdf], **Ada**ptive **Boost**ing was one of the earliest practical boosting algorithms, and it . It is an online algorithm, which is given a weak learner in a series of rounds on the training set and maintains a distribution over the training set, so that the weak learners focus on examples which become difficult to classify.

Boosting algorithms have the merit that they have empirically proven to be resistant to overfitting within the number of iterations run until convergence. However, they have been found to be highly susceptible to noise. Dietterich [CITATION needed] found that noise 'damages the accuracy of Adaboost severely.' This is because noisy examples, or outliers, are quite difficult to classify. Thus, outliers may be assigned excessively high weights, and can heavily influence the final classifier.

However, real world datasets often contain some amount of noise, and as such several variants of Adaboost have been proposed to be more robust to noisy data. A dataset to be classified can be characterized by its attributes and class labels, and classification essentially involves choosing a set of attributes to characterize the class label. Zhu and Wu [Class Noise vs. Attribute Noise: A Quantitative Study of Their Impacts] characterize noise as 'anything that obscures the relationship between the attributes and class.'

Thus, we can view noise in real datasets as falling broadly into two categories; *class noise* and *attribute noise*. Much of the focus in previous work has been placed on class noise, but less attention has been paid to the performance of classifiers when subject to attribute noise. Class noise may be dealt with by throwing away incorrectly labeled examples, as Brownboost attempts to do, but this is less practical for attribute noise, as other attributes of that example may contain useful information for classification[Class Noise vs. Attribute Noise: A Quantitative Study of Their Impacts].

In this paper, we explore the effect of different types and thresholds of both class noise and attribute noise on the performance of four boosting algorithms; Adaboost, Logitboost, Brownboost and Savageboost. We construct an artificial data set and create noisy versions of it, aimed at simulating the types of noise that can arise in real data sets. We then conduct an empirical study of the performance of the four boosting algorithms on data sets with various types of noise.

In Section 2, we will describe and implement Adaboost, Logitboost, Brownboost and Savageboost. In Section 3, we detail our experimental procedure, and in Section 4, describe our findings, followed by a summary of our experiment.

2 Boosting Algorithms

Several adaptations to Adaboost have been proposed with the express purpose of making the algorithm more robust against noisy datasets. In this section, we detail Adaboost and three other boosting algorithms elected for their robustness against noise and describe their background and implementation.

2.1 Adaboost

Adaboos Brief history of Adaboost.

pseudocode

Theoretical predictions/ Why bad at noise

2.2 Logitboost

Logitboost is a boosting algorithm used for classification.

Origins in re-formulation of Adaboost as a additive logistic regression problem

2.3 Brownboost

2.4 SavageBoost

3 Experimental Procedure

3.1 Generating An Artificial Dataset

We conducted a series of empirical evaluations of each algorithm's performance on an artificial data sets, to which varying degrees and types of noise were added.

We constructed an n-dimensional dataset by sampling each dimension's mean and standard deviation from a Gaussian distribution ($N(0,1)$ and $N(1,1)$ respectively), so that dimension i had mean μ_i and standard deviation σ_i. We then generated the n-dimensional data points in our artificial data set by sampling each dimension according to the distribution $N(\mu_i, \sigma_i)$.

For our implementation, we generated 1000 data points in $n = 10$ dimensions.

We defined the 'true classification' of the points in the artificial dataset by a randomly chosen hyperplane, so that our dataset was linearly separable.

3.2 Adding Noise

Following this, we created a set of 'noisy versions' of our dataset, by adding varying amounts and types of noise, following the methods prescribed be Zhu and Wu [CITATION].

We generated class noise as follows:

1. Mislabeled data: We randomly selected $p\%$ of our dataset and flipped the labels for those selected data points.
2. Contradictory labels: We randomly selected $p\%$ of our dataset, duplicated the data points and assigned them the opposite label.

3.3 Running Trials

We implemented Adaboost, Logitboost, Brownboost and Savageboost in Python, using decision stumps as our weak learners.

We then conducted a series of experiments on these implementations for each data set.

We set aside 30% of dataset to be used as a test set, and trained the algorithms on the remaining 700 data points. We noted the training error at each iteration of the algorithm until convergence. Then, we computed the test error for each algorithm.

We repeated this process on each data set for 50 trials, and presented the average results obtained.

TO DO: Explain parameter choices Explain how long we let the algorithm run.

The results of these experiments are presented in Section .