CSCI-GA.2566-001 Project

Lucas Swiniarski
NetID : ls4411
lucas.swiniarski@nyu.edu

Rajeev Kumar
NetID : rk2805
rajeevkr@nyu.edu

Rolf Kranold Carlsen
NetID : rkc288
rkc288@nyu.edu

Abstract

Real world data are noisy, containing both classification and feature noise. Getting a good prediction accuracy for a highly noisy data-set is always challenging. Several boosting algorithm has been proposed in the literature to try to improve prediction accuracy by either removing noisy data or making the optimization function better. To improve accuracy results on highly noisy data-sets, we propose two ensemble methods : *Strong Regularized AdaBoost*, modifying the objective function of AdaBoost, and *Random Forest AdaBoost*, reducing noise impact using *majority vote*. Our results show that the first algorithm performs well with noise on labels, while the second one has promising results on feature noise.

1 Introduction

Adaboost is well know to be sensitive to all kinds of noise [1, 2]. The issue arise from its efforts to classify difficult samples. When data-sets are noise free, it's a good approach. However, the real world data-sets such as, medical data, and crowd sourced data are noise prone [3] and boosting algorithms could result in poor generalization performance in the presence of noise by giving too much weight to outliers [4, 5, 6].

In the literature, two sets of boosting problems has been considered, (i) boosting in presence of label noise[7, 8, 9], (ii) boosting in presence of feature noise[10, 11, 12]. However, often noise has been added without considering the meaning of features and labels. Authors in [10] has added random classification (label switching) and feature (Gaussian) noise without taking into consideration if it fits a real world situation. Prior to adding noise, we need to investigate which features and labels are noise prone. For instance, a feature 'How many children do you have' is not prone to noise.

In this paper, we first examine the data-sets to determine the type of noise that it may contain. Next, we propose two algorithms *Strong Regularized-AdaBoost* and *Random Forest AdaBoost* as our boosting approach to enhance the performance on noisy data-sets. The two proposed algorithms are based on *Regularized-AdaBoost* [1] and *Random Forest* [13], respectively. Using AdaBoost on a data-set with outliers may generalize poorly, as it attaches large weights to these points. To improve the generalization error, the regularized-AdaBoost algorithm can be used, which introduces slack variables, a common technique used for instance in the Support Vector Machine algorithm [14, 15], in the objective function of AdaBoost. It reduces the weights of hard to classify observations. In our proposed *Strong Regularized-AdaBoost* algorithm, we modify the regularized-AdaBoost algorithm by emphasizing the weights of slack variables, hence punishing more outliers. This new algorithm should therefore be successful in an environment with label noise. Our second approach is based on random forest [13]. The common AdaBoost algorithm have a similar behaviour to Decision trees in a noisy environment, building a complex classifier to fit perfectly the training set. The techniques used in Random Forest : training on bootstrapped samples, training on random subset of features, majority vote; should help the algorithm generalize better. As AdaBoost is generally a better classifier than Decision Tree, it should yields better results with a bagging algorithm.

This paper is organized as following: Section 2 presents previously discussed boosting and bagging algorithms relevant to our work. Section 3 presents our proposed l_1-regularized AdaBoost framework. We present our proposed random forest Adaboost algorithm in Section 4. We present experiment description in Section 5 and results of our experiment in Section 6. Finally, Section 7 presents the concluding remark of the paper.

2 Related Boosting and Bagging Algorithms

In this section, we briefly present three widely used algorithms, upon which our proposed algorithms builds on.

Algorithm 2 presents the AdaBoost algorithm. The algorithm combines many simple classifiers, often these classifiers are simple threshold functions, or boosting stumps. In the first round it trains a classifier on the data-set, all weights being equal. It then updates the weights of each observations, such that in the next round the classifier will put more weight on the observations previously misclassified. This process continues for T rounds. The final classifier is then a linear combination of all the simple classifiers.

Algorithm 1 Adaboost

```
1: for $i \leftarrow$ to $m$ do
2:     $D_1(i) \leftarrow \frac{1}{m}$
3: for $t \leftarrow$ to $T$ do
4:     $h_t \leftarrow \underset{h \in H}{\text{argmin}} \underset{S \sim D_t}{\Pr} [y_i h_t(x_i) = -1]$
5:     $\alpha_t \leftarrow \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
6:     $Z_t \leftarrow 2\sqrt{\epsilon_t(1-\epsilon_t)}$
7:     for $i \leftarrow$ to $m$ do
8:         $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
9: return $h = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x_i)\right)$
```

Algorithm 2 presents the L2 regularized AdaBoost. The algorithm is similar to AdaBoost, however the regularization term penalize observations that are often wrong. It often offers better generalization performances.

Algorithm 3 presents the random forest algorithm. It builds n decision trees. Each decision tree is built using a subset of training samples (bootstrapping), and a subset of the features. The decision tree is usually built without constraint (depth or number of elements in leafs). The decision function is a majority vote from all the decision trees.

3 Strong l_2-Regularized Adaboost

3.1 Motivations

Adaboost is classifying points by using weak learner, usually boosting stumps. At each round, it update the weight of each training example in order to reflect how well it classify each points. If we have have points that are difficult to classify, Adaboost will very quickly put a lot of weight on those points. In a real world environement, it is probable that some of our points are mislabeled. In this setting, Adaboost will tend to put a lot of weights on mislabeled data against all other training points. By doing so, the generalization power of Adaboost is compromised in order to build a

Algorithm 2 l_1-Regularized Adaboost

```
1: for $i \leftarrow$ to $m$ do
2:     $D_1(i) \leftarrow \frac{1}{m}$
3: for $t \leftarrow$ to $T$ do
4:     $h_t \leftarrow \underset{h \in H}{\text{argmin}} \Pr_{S \sim D_t}[y_i h_t(x_i) = -1]$
5:     $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
6:
7:     for $i \leftarrow$ to $m$ do
8:         $\xi_i^t \leftarrow \sum_{t=1}^{T} \alpha_t D_t(i)$
9:         $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i) + C(\xi_i^{t-1} - \xi_i^t))}{NormalizationConstant}$
10:
11: return $h = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x_i)\right)$
```

Algorithm 3 Random Forest

```
1: Input:
2:     $S$: training data set
3:     $m$: Length of feature SubSpace ( usually square root of the number of features).
4:     $bootstrap\_length$: Length of bootstrap, usually 2/3 of the training-set.
5: for $b \leftarrow$ to $B$ do
6:     $S_b \leftarrow$ bootstrapSample$(S, bootstrap_length)$
7:     $F_b \leftarrow$ subFeatureSpace$(S, m)$
8:     $C_b \leftarrow$ Tree$(S_b, F_b)$
9:     $E = E \cup \{C_b\}$
10: return $h = majority\_vote(E)$
```

complex classifier that succesfuly classify all training points. Looking at the regularized version of adaboost 2, we find that, even if the theoretical intuition is very promising, the regularization power of the algorithm is too weak to have the intended behavior.

3.2 Presenting a new weight update rule

In order to regularize Adaboost, we can draw parallel with a Support Vector Machine case, declaring a margin and adding slack variables to take into account outliers (points that are not well classified by the Adaboost at round t).

$$margin(x_i) = y_i \sum_{s=1}^{T} \alpha_s h_s(x_i) \tag{1}$$

We can connect Adaboost's theory to SVM's by adding slack variables, which leads to l_2-Regularized Adaboost 2 :

$$margin(x_i) = y_i \sum_{s=1}^{T} \alpha_s h_s(x_i) - C\xi_i^T \tag{2}$$

$$\xi_i^T = (\sum_{s=1}^{T} \alpha_s D_s(i))^2 \tag{3}$$

The result algorithm is the same as Adaboost, except for the weight update rule. The weight update rule can be obtained by using the exponential expression that Adaboost is trying to minimize using coordinate descent :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_{t+1}} \exp(-y_i \alpha_t h_t(x_i) - C(\xi_i^t - \xi_i^{t-1})) \tag{4}$$

$$\xi_i^t - \xi_i^{t-1} = \xi_i^{t-1}((1 + \frac{\alpha_t D_t(i)}{\sum_{s=1}^T \alpha_s D_s(i)})^2 - 1) \tag{5}$$

$$= \xi_i^{t-1}(2\frac{\alpha_t D_t(i)}{\sum_{s=1}^T \alpha_s D_s(i)} + o(\frac{\alpha_t D_t(i)}{\sum_{s=1}^T \alpha_s D_s(i)})) \tag{6}$$

$$= 2\alpha_t D_t(i)\sqrt{\xi_i^{t-1}} + o(\alpha_t D_t(i)\sqrt{\xi_i^{t-1}}) \tag{7}$$

This update rule is comparing ξ_i^t to ξ_i^{t-1}, which end up mostly using the last weight. Unfortunately, what we want is to severly diminish the power of points that have had high weight at the beginning of the training. Here slowly there memory of high weight of the beginning is fading as the update rule mostly look at the last weight. A new update rule, remembering the weight history of a point can be created using :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_{t+1}} \exp(-y_i \alpha_t h_t(x_i) - C\xi_i^t) \tag{8}$$

With this new weight update we can look at the margin definition with slack variables :

$$margin(x_i) = y_i \sum_{s=1}^{T} \alpha_s h_s(x_i) - C \sum_{t=1}^{T} \xi_i^t \tag{9}$$

$$\xi_i^t = (\sum_{s=1}^{t} \alpha_s D_s(i))^2 \tag{10}$$

With this new algorithm it is intended to quickly find outliers and almost not use them to build the classifier, thus having an algorithm that find mislabeled points and don't count on them to learn. Also the power of the regularization term k in $\xi_i^t = (\sum_{s=1}^t \alpha_s D_s(i))^k$ can be fine-tuned, in our experiments, we tried $k = 2$ and $k = 3$ in order to have an idea of the impact of the power function on the generalization error.

3.3 Algorithm

The algorithm 4 describe the pseudo-code of Strong l_2-Regularized Adaboost.

Algorithm 4 Strong l_2-Regularized Adaboost

```
1:  for $i \leftarrow$ to $m$ do
2:      $D_1(i) \leftarrow \frac{1}{m}$
3:  for $t \leftarrow$ to $T$ do
4:      $h_t \leftarrow \underset{h \in H}{\text{argmin}} \underset{S \sim D_t}{\Pr}[y_i h_t(x_i) = -1]$
5:      $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
6:
7:      for $i \leftarrow$ to $m$ do
8:          $\xi_i^t \leftarrow (\sum_{t=1}^T \alpha_t D_t(i))^2$
9:          $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i) - C\xi_i^t)}{NormalizationConstant}$
10:
11: return $h = sign\left(\sum_{t=1}^T \alpha_t h_t(x_i)\right)$
```

3.4 Weight distribution throughout the learning

Our intuition on Adaboost is that Adaboost is not distributing well its weights when data is mislabeled. We have done an experiment using the Spambase dataset, ploting at each round the minimum number of points in order to have 50 % of the weights (fig. 1 and 2).

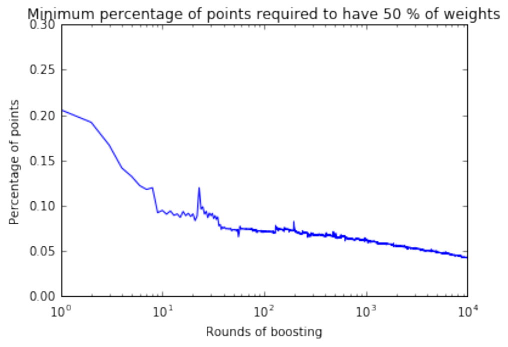

Figure 1: Adaboost

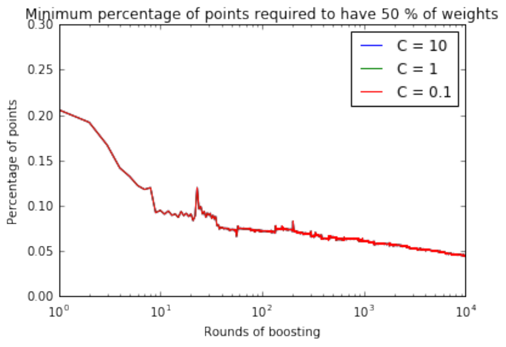

Figure 2: l_2 Regularized Adaboost

The surprising result is both Adaboost and its regularized version behave the same way, and both tend to give quickly most of its weight to few points. The regularization C is also not changing the weight distribution. We expecte our new algorithm to use more points to make a decision (fig. 3 and 4).

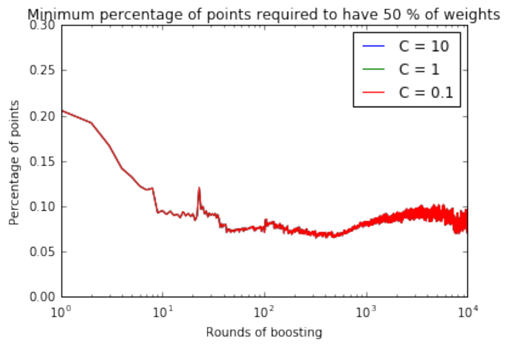

Figure 3: Strong l_2-Regularized Adaboost

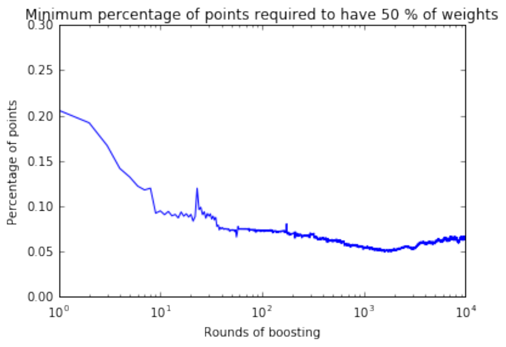

Figure 4: Strong l_3-Regularized Adaboost

We have an expected behaviour for our algorithms in term of weight distribution, the Strong l_2-Regularized Adaboost use always more points in order to build its classifiers at each rounds. More visualization of the weights distributions can be found in the Appendix. From a better distribution of weights we should have better generalization [16].

4 Random Forest Adaboost

4.1 Motivations

In the classical random forest algorithm, random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). Each tree is grown as follows:

- If the number of cases in the training set is N, sample N cases at random – but with replacement, from the original data. This sample will be the training set for growing the tree.

- If there are M input variables, a number $m \ll M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning.

This make the algorithm more robust to noise by eventually ignoring outliers and noise-corrupted examples rather than putting useless efforts on noisy samples as AdaBoost. The random forest algorithm outperform many other classifiers and robust against over-fitting. Using the benefits of both AdaBoost and random forests algorithms, authors in [17] proposed a combined algorithm to predict breast cancer survivability. They showed that combination of AdaBoost and random forest outperforms a single classifier and other combined classifiers for the breast cancer survivability prediction. In their proposed algorithm, they utilize the random forest as weak learner to generate the prediction model. However, the asset of weak learners for AdaBoost is that it is particularly fast to train. We put forward a new combined algorithm of AdaBoost and Random Forest, using AdaBoost as base classifiers for a Random Forest. First, Adaboost and Decision trees both over-fit on noisy data when over-trained, thus it makes sense for both of use to use a bagging algorithm on top to reduce the effect of noise. Secondly, generally AdaBoost performs better than Decision tree as a classifier. The increased performance on base classifier for Random Forest should also be seen at a top-level using a bagging algorithm, and Random Forest Adaboost should thus yields better result.

4.2 Random Forest AdaBoost Algorithm

Algorithm 5 presents steps used in our proposed random forest AdaBoost algorithm.

Algorithm 5 Random Forest

```
1:  Input:
2:  $S \leftarrow$ Dataset
3:  $T \leftarrow$ Number of iteration for Adaboost
4:  $\rho \leftarrow$ Number of features in training set
5:  for $i \leftarrow 1$ to $B$ do
6:      $S_i \leftarrow$ Sample 2/3 of observations from S with replacement
7:      $S_i \leftarrow$ Pick randomly $\sqrt{\rho}$ of the features from $S_i$
8:      $h_i \leftarrow$ Train Adaboost(T) on $S_i$
9:  next $i$
10: return $h \leftarrow sign\left(\sum_i h(i)\right)$
```

5 Experiment Description

We conducted experiments to compare AdaBoost, L_2-regularized AdaBoost, our proposed strong L_2-regularized AdaBoost, and our proposed Random Forest AdaBoost, under two kinds of noises.

- **Classification Noise**: Classification noise can be found in data-sets where the labels are made by humans, and the labels are not easily identifiable. For instance in the data-set we consider, the spam base, it may not always be clear whether an email is spam or not. Hence sometimes people will classify incorrectly. There may be some correlation between the features and the incorrect classification, however, we ignore that link here and simply assume that the incorrect labeling is random. For the spam base data-set we then enhance this noise by randomly switching some of the labels. We add only noise to the training set, and then use the testing set to compare performance.
- **Feature Noise**: Feature noise comes from imperfect observation of the features. In medical data this could for example be measurement error from imperfect measuring devices. This is the kind of data which we consider. The two data-sets we use are Liver Disorder and Pima Indians Diabetes. They both have multiple features which could potentially contain measurement noise, for example blood pressure or insulin level. We conjecture that these noises approximately follows Gaussian distributions. A Gaussian distribution is often used to model noise (see Appendix). The Gaussian distribution for each features have a mean 0 and a variance here taken as 10 % of the variance of the feature. A high variance of a feature

may, however, not be due to measurement error, but just that the variance of the particular feature is high. A better measure of the actual noise would be to look at the particular instrument used for measuring. However, we do not have access to this information, and we hope that this way at least relates the added noise to the actual noise.

Table 1 presents the statistics of the datasets considered.

- **Spam base**: we do not anticipate any feature noise as all of the features in the data set can be calculated with very high accuracy. Thus, we assume spam base data set contains only classification noise and we a 10% noise on labels only.
- **Liver disorder and Pima indians diabetes**: We consider only feature noise, because one would expect that classifying ex post liver disorder and diabetes is not subject to mislabeling. For each of those data-sets, we went described each features and selected a subset of features that would be subject to noise. A description of such feature is given in appendix.

Table 1: Statistics of Data Sets

Name	**Samples**	**Number of Features**
Spambase	4601	57
Liver Disorder	345	6
Pima Indians Diabetes	768	8

We carry out the experiment in the following way. First, we add noise to the particular data-set. Second, we do 10-fold cross validation on the training set to find to best model parameters. Finally we calculate the error on the testing set. As the noise is random, we repeat this process 100 times for each algorithm, in order to the reduce the randomness. We then compute the average error and standard deviation. For the spambase data-set we repeated the process 10 times, as cross-validating Adaboost and its regularized version took 18 hours already.

For Adaboost we did cross validation over $T = \{100 \cdot i | i \in \{1 : 21\}\}$. For L2-Adaboost and strong L2-Adaboost we also validate over penalty, C for $C = \{10^i | i \in \{-3 : 4\}\}$. For the Random Forest Adaboost algorithm, we dont use any cross validation, due to limited capacity, hence the error for this algorithm should be seen as an upper limit, and we expect it to do better with cross validation. We used Random Forest Adaboost by learning each rounf of adaboost on 2/3 of the training-set randomly selected, and on $\sqrt{n}$ features where n is the total number of features. Then the selected parameters were :

Table 2: Random Forest Adaboost Hyper-parameters

Data-set Name	**Number of Adaboost Classifiers**	**Number of Iterations T**
Spambase	1000	100
Liver Disorder	200	50
Pima Indians Diabetes	400	50

In the bootstrap step of the Random Forest Adaboost, following [18] each Adaboost is trained on $\frac{2}{3}$ data points selected from the data set. Similarly, number of feature selected we follow [19] using $\sqrt{\rho}$, where ρ is the number of features.

6 Results on tests

The results from the experiments described above is in tables 3, 4 and 5. Number are in percentages, with standard deviations is in parentheses

6.1 Strong l_2-Regularized and l_3-Regularized Adaboost Results

Looking at the Liver Disease data-set and the Prima Indian data-set, our algorithms under perform compared to Adaboost, at any given noise. This algorithm is not adding anything new to the Adaboost

Table 3: Results on the spambase dataset

Level of noise	Adaboost	L2-Adaboost	Strong L2-Adaboost	Random Forest Adaboost
0%	5.6 (0.6)	5.5 (0.5)	5.3 (0.3)	11.6 (0.6)
5%	6.9 (0.5)	6.2 (0.2)	5.3 (0.5)	11.9 (0.8)
10%	7.5 (0.6)	7.5 (0.6)	5.7 (0.3)	12.6 (1.2)

A noise level of x% corresponds to randomly switching x% of the labels.

Table 4: Results on the liver disease dataset

Level of noise	Adaboost	L2-Adaboost	Strong L2-Adaboost	Random Forest Adaboost
0%	28.4 (2.4)	26.0 (4.0)	28.9 (4.8)	33.5 (4.3)
5%	30.4 (4.4)	35.2 (2.6)	37.8 (4.1)	34.7 (4.1)
10%	39.4 (1.6)	38.0 (4.0)	39.0 (3.7)	39.3 (4.2)

A noise level of x% corresponds to adding a mean zero Gaussian with variance of x% of the empirical feature variance.

Table 5: Results on the pima indians diabetes dataset

Level of noise	Adaboost	L2-Adaboost	Strong L2-Adaboost	Random Forest Adaboost
0%	25.2 (2.8)	24.5 (2.2)	23.9 (2.7)	29.3 (2.6)
5%	24.7 (2.1)	24.5 (2.6)	25.7 (2.1)	30.0 (3.1)
10%	28.4 (2.4)	27.1 (1.9)	26.0 (2.1)	30.9 (2.6)

A noise level of x% corresponds to adding a mean zero Gaussian with variance of x% of the empirical feature variance.

in order to have better results on data-sets with noisy features, hence the result. Also, given the regularization, what effectively happens is that if the decision function to learn is complicated and some points are effectively hard to classify, this algorithm will make those points irrelevant. Thus it is harder with this version of Adaboost to learn difficult decision boundaries, and it can explains the result.

On the other hand, with the spam base data-set we can see a good improvement on the results. As spam base has a noise consisting of mislabeled data, the algorithms are correctly finding those outliers and assigning them very low weights. Thus the algorithm is generalize better and have a lower generalization error.

It is also quite remarquable to look at the different behaviours of Strong l_2-Regularized and l_3-Regularized Adaboost Results. Both have approximately the same results, even though the l_3-Regularization seems to work better on low noise levels. The power of the regularization changes the error rate and can be viewed as a hyper-parameter to cross-validate.

6.2 Random Forest Adaboost Results

The Random Forest Adaboost's hyper-parameters could not be fine-tuned, thus the results are quite far from the best Adaboost. Given a set of hyper-parameters for the algorithm, we can still infer its generalization behaviour and Resistance to noise based on this results by looking at the growth of the error rate given increasing noise.

For the prima indian diabetese data-set, the error rate is increasing very slowly, compared to all other algorithms. Given well-tuned parameters, we can hope to have way better results with and without noise. With the maximum noise level, the Random Forest Adaboost is even beating the regularized versions of Adaboost, and have very similar results to adaboost.

The same pattern can be viewed with the liver disease data-set. This time the Random Forest Adaboost is even beating all algorithms with the maximum noise level.

For the Spam base data-set, we also see a very slow increase in error rate as the noise increase. The bootstrapping procedure allows the algorithm to perform remarkably well with noise on labels.

7 Conclusion

In this work, we identify the methodology to properly add noise to datasets, both classification and feature noise. Thereafter, we proposed two boosting algorithm to get high accuracy in highly noisy environment. Our first boosting algorithm *Strong l_2 Regularization AdaBoost* is the extension of l_1-regularized AdaBoost. Our second algorithm *Random Forest AdaBoost* is based on random forest algorithm. Our proposed algorithms outperforms existing boosting algorithms, such as AdaBoost, l_1-regularized, and l_2-regularized AdaBoost in highly noisey environment. However, on the down side they perform poor on low noisy data.

References

[1] Klaus R. Müller Gunnar Rätsch, Takashi Onoda. Regularizing adaboost.

[2] Adam Tauman Kalai and Rocco A. Servedio. Boosting in the presence of noise. *Journal of Computer and System Sciences*, 71(3):266 – 290, 2005. Learning Theory 2003Learning Theory 2003.

[3] R. Y. Wang, V. C. Storey, and C. P. Firth. A framework for analysis of data quality research. *IEEE Transactions on Knowledge and Data Engineering*, 7(4):623–640, Aug 1995.

[4] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[5] Alexander Vezhnevets and Olga Barinova. *Avoiding Boosting Overfitting by Removing Confusing Samples*, pages 430–441. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[6] Amitava Karmaker and Stephen Kwek. A boosting approach to remove class label noise. *Int. J. Hybrid Intell. Syst.*, 3(3):169–177, August 2006.

[7] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[8] Jakramate Bootkrajang and Ata Kabán. Boosting in the presence of label noise. *arXiv preprint arXiv:1309.6818*, 2013.

[9] Amitava Karmaker and Stephen Kwek. A boosting approach to remove class label noise1. *International Journal of Hybrid Intelligent Systems*, 3(3):169–177, 2006.

[10] Prem Melville, Nishit Shah, Lilyana Mihalkova, and Raymond J. Mooney. Experiments on ensembles with missing and noisy data. In F. Roli, J. Kittler, and T. Windeatt, editors, *Lecture Notes in Computer Science: Proceedings of the Fifth International Workshop on Multi Classifier Systems (MCS-2004)*, volume 3077, pages 293–302, Cagliari, Italy, June 2004. Springer Verlag.

[11] Qiang Wu, Chris JC Burges, Krysta M Svore, and Jianfeng Gao. Ranking, boosting, and model adaptation. Technical report, Technical report, Microsoft Research, 2008.

[12] Marian Craciun Daniel Neagu Elias Kalapanidas, Nikolaos Avouris. Machine learning algorithms: a study on noise sensitivity. 2002.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Mach. Learn.*, 20(3):273–297, sep 1995.

[15] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[16] Rocco A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research 4 (2003) 633-648*, 2003.

[17] J. Thongkam, G. Xu, and Y. Zhang. Adaboost algorithm with random forests for predicting breast cancer survivability. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 3062–3069, June 2008.

[18] Deepanshu Bhalla. Random forest explained in simple terms. `http://www.listendata.com/2014/11/random-forest-with-r.html`.

[19] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

[20] Carlos Fernandez-Granda. Linear regression. 2016.

Appendix

Weight distribution of Strong l_2-Regularized Adaboost

In order to better understand the behaviour of Adaboost and its regularized version, we can plot, at each round, the distribution of weights. In order to do so we have normalized the weights by the average weight value $\frac{1}{m}$ where m is the training sample size. As the range of weights vary a lot, we plotted its log-scale. We plotted both the weight distribution and the cumulative weight distribution. The cumulative weight distribution gives us a good idea of which points are weighting a lot in the decision.

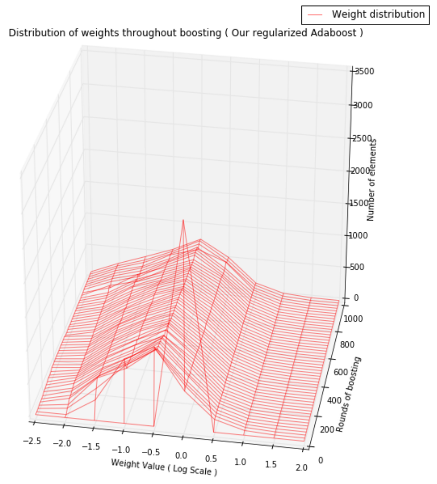

Figure 5: Adaboost Weight Distribution

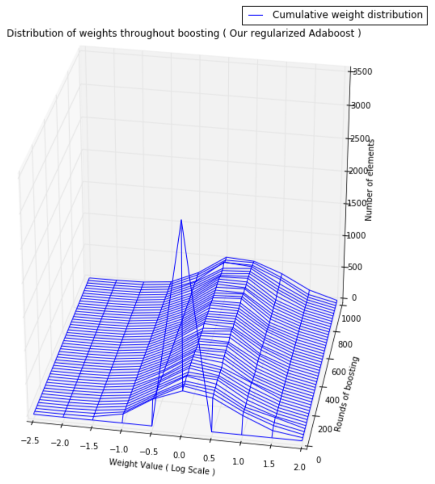

Figure 6: Adaboost Cumulative Weight Distribution

From this charts (fig. 5 and 6), we can see that most of the points have a weight of 0.5 or below, but actually the cumulative weight is centered around 0 and higher. With this graph we can see the phenomena observed earlier of Adaboost almost ignoring most of the weights. Also, the weight distribution very quickly converges to it's end distribution (in less than 5 boosting rounds), which let very few boosting rounds for the majority of points to weight in the balance.

l_2-Regularized Adaboost (fig. 7 and 8) actually have the same distribution weight as earlier.

The Strong l_2-Regularized Adaboost have a completely different weight distribution. The first rounds of boosting are very similar to Adaboost and l_2-Regularized Adaboost in terms of distribution. But further away we can clearly see the effects of regularization, almost no points have very large weights. We have bounded indirectly the maximum weight value to $\frac{10}{m}$, which have the effect of augmenting

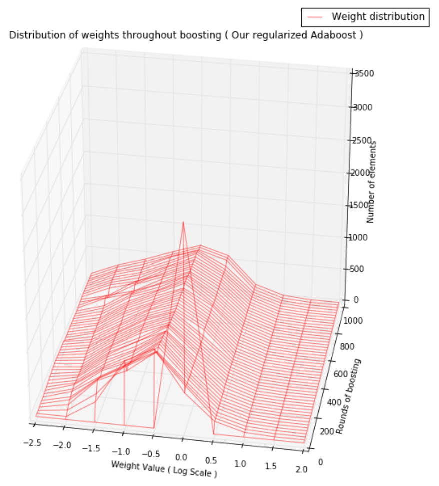

Figure 7: l_2-Regularized Adaboost Weight Distribution

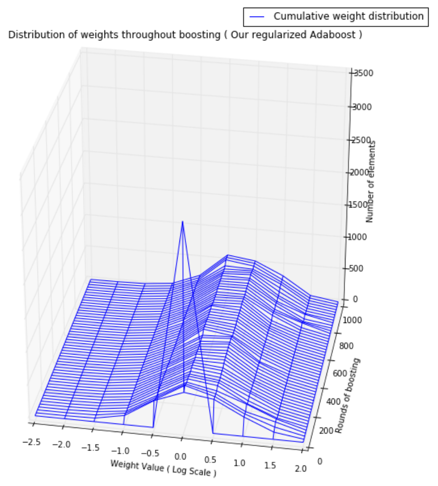

Figure 8: l_2-Regularized Adaboost Cumulative Weight Distribution

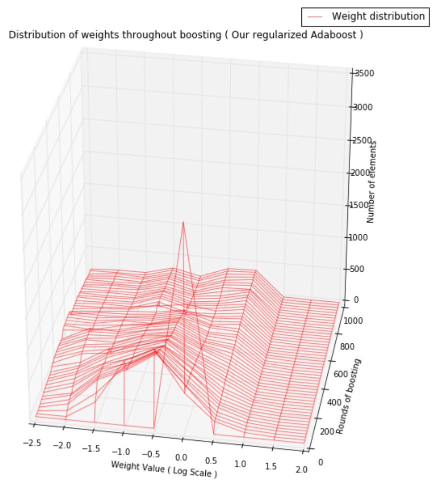

Figure 9: Strong l_2-Regularized Adaboost Weight Distribution

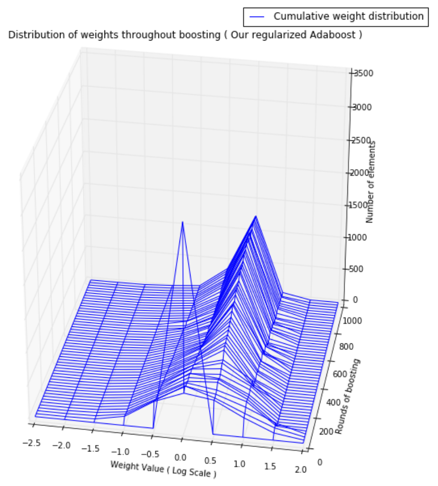

Figure 10: Strong l_2-Regularized Adaboost Cumulative Weight Distribution

the number of points used in the decision, or reducing the number of points totally left out of the decision process because of a weighting value very low.

Noise as a Gaussian distribution

Often when modeling data with noise, we model the noise as a Gaussian distribution. For instance, in Linear regression [20] we model the features being affected by a noise following a Gaussian distribution, which leads the solution of the maximum likelihood estimator to be the solution of the least-squares problem. Given a feature in our data-set, which comes from a measurement. We can say that the random variable of the feature is given by the true value of the measurement with a sum of independent phenomenon which is the noise. This sum of independent phenomenon with bounded variance can be approximated given the Central Limit Theorem as following a Gaussian Distribution. Thus the noise modeled as a Gaussian.

Selection of noisy features

Here, we describe the features of data-sets considered in this paper. For simplicity, we considered few of the features for the description. Table 7 and 7 represents medical data-sets for pima Indian diabetes and liver-disorder cases. In the PIMA Indian diabetes the feature space such as the number of time a lady got pregnant can evaluated without any mistake. Thus, such type of feature space can be assumed of noise free. However, other measurements are subject to Gaussian noise. Similarly, liver disorder data-set contains machine measurements and all the features are prone to noise.

Table 6: Pima Indians Diabetes Dataset

Features	**Number of times pregnant**	**Glucose concentration**	**blood pressure**	**Triceps skin**
Type	Not-noisy	noisy	noisy	noisy
STD	0	31.98	19.35	15.96

Table 7: Liver Disorder Dataset

Features	**MCV**	**alkphos**	**spgt**	**sgot**
Type	noisy	noisy	noisy	noisy
STD	90.15	69.87	30.40	24.65