Optimization of AdaBoost in Noise Processing

Wenting Liu
2021533058
liuwt@shanghaitech.edu.cn

Xinhe Chen
2021533093
chenxh1@shanghaitech.edu.cn

Yixuan Li
2021533140
liyx3@shanghaitech.edu.cn

Abstract

AdaBoost algorithm is a commonly used integrated class algorithm for solving classification problems. However, when facing noisy datasets, the classification result will be inaccurate due to the excessive weight of the noisy data. Therefore, in this study, we try to add the dynamic prediction of noise weights to the AdaBoost algorithm. Specifically, we predict noise using AdaBoost classification labels and reallocate the data weight using the noise weights. Finally, we compare our algorithm with XGboost algorithm on artificial datasets to prove the effectiveness of our method on noisy datasets.

1 Introduction

1.1 Overview of AdaBoost

Adaboost (Adaptive Boosting) is an integrated learning algorithm designed to build a powerful classifier by combining multiple weak learners. The core idea is to adjust the sample weights through iterative training so that the model pays more attention to misclassified previous samples.

1.2 Drawback of AdaBoost

The original AdaBoost algorithm continues to increase the weight of misclassified samples when iterating:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & h_t(s_i) = y_i \\ e^{\alpha_t} & h_t(s_i) \neq y_i \end{cases}$$

There is an obvious disadvantage in this process: it amplifies the effect of noise, making it very sensitive to noise and less robust, and prone to overfitting in the presence of noise.

1.3 Criticism of the existing work

Many current studies have attempted to improve AdaBoost by filtering to directly exclude noisy points, but this has the potential to mistakenly kill valid information.

So we try to dynamically estimate the noise weights and use the results of each round of AdaBoost to improve its sensitivity to noise. We hope to form a mutually reinforcing iterative process, in which AdaBoost helps us to estimate the noise by predicting the labels, and the noise estimation improves the weight assignment of AdaBoost.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2 Methodology

2.1 Local Consistency Coefficient

We're inspired by the definition of dNN when reading relevant papers on AdaBoost. It calculates the ratio of the distance from the sample point x to its neighbors with the same label and the distance to its neighbors with different labels.

$$dNN(x_i) = \frac{d(x_i, NN(x_i) \in y_i)}{d(x_i, NN(x_i) \notin y_i)}$$

Its essence is to solve for the central consistency of the sample point x. Therefore, we try to use the more general local consistency coefficient $Lc(x)$ as a noise evaluation factor of sample x, which is defined below:

$$\mathrm{Lc}\,(x) = \frac{\sum_{x_i \in \mathrm{Neig}(x,k)} C\,(x) == C\,(x_i)}{k}$$

$C(x)$ represents the class of x, Neig(x, k) represents the k-nearest neighbors of x. We use the kNN algorithm to find k-nearest neighbors of x and then calculate the proportion of neighbors with consistent labels. Its actual significance is the similarity to the surrounding neighbors, a greater $Lc(x)$ indicates lower noise potential. $Lc(x) \in [0, 1]$, the more closer $Lc(x)$ is to 0, the higher probability that x is a noise sample.

2.2 Preference Coefficient

However, relying only on label consistency may lead to misclassification of outliers. Besides, the local consistency levels vary among different regions, with large local consistency in the center and small local consistency at the edge. In this case, the local consistency coefficients Lc vary significantly from region to region.

Based on the above issues, we consider further weighting based on data set characteristics. This weight should not be biased by the density characteristics and edge points of the data set.

Therefore, we refer to the Local Reachability Density(LRD):

$$\mathrm{LRD}(p) = \left(\frac{1}{k} \sum_{O \in N_k(p)} \frac{reach - dist_k(p, o)}{density_k(o)} \right)^{-1}$$

and Local Outlier Factor(LOF):

$$\mathrm{LOF}(p) = \frac{1}{k} \sum_{O \in N_k(p)} \frac{LRD(o)}{LRD(p)}$$

This is a commonly used anomaly detection algorithm that calculates the density ratio of a data point relative to its surrounding neighbors to determine the degree of anomaly of the data point. Based on the idea of comparing the algorithm with the neighborhood points in different categories, we design the coefficient p(x,c) to evaluate the preference of the points to different categories.

$$p(x, c) = \frac{d(x, k, c)}{\sum_{x_i \in \mathrm{Neig}(x,k,c)} \frac{d(x_j; k, c)}{k}}$$

In which $d(x, k, c) = \sum_{x_i \in \mathrm{Neig}(x,k,c)} d(x_i, x)/k$ where $d(x_i, x)$ represents the Euclidean distance of the samples x_i and x. The larger $p(x, c)$ is, the higher the preference x has for class c. It has the advantage of avoiding the bias caused by the different data density and the different degree of outlier in different categories.

2.3 Pseudocode

The pseudocode of the denoised AdaBoost agorithom is shown as follows:

Perform PCA on the training data to reduce dimensionality;
for *each principal component* **do**
  Normalize the principal component;
end
Initialize weights for each sample: $w[i] = 1/N$, where N is the number of samples;
Initialize the ensemble of weak classifiers, H;
for $t = 1$ **to** T **do**
  Train weak classifier h_t on the training data with weights w;
  Compute the weighted error: $\epsilon_t = \frac{\sum(w[i] \times \text{error})}{\sum(w)}$;
  Compute the classifier weight: $\alpha_i = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;
  for *each sample* i **do**
    $D_{t+1}[i] = \frac{D_t[i] \times \exp(-\alpha_t y(i) h_t(i))}{Z_t}$;
    Compute the local consistency:
    $\mathrm{lc}\,(x) = \frac{\sum_{x_i \in \mathrm{Neig}(x,k)} C(x) == C(x_i)}{k}$;
    Compute the perference:
    for *each category* **c do**
      $p(x, c) = \exp\left(\frac{d(x,k,c)}{\sum_{x_i \in \mathrm{Neig}(x,k,c)} d(x_j;k,c)}\right)$
    end
  end
  Update noise weights:
  if $t > M$ ***and*** t *mod* $m = 0$ **then**
    for *each sample* i **do**
      $\beta(i) = \mathrm{lc(x)} * \mathrm{logistic}(-\frac{\mathrm{p(x_i, C - C_{H_{t(i)}})}}{\mathrm{p(x_i, C_{H_{t(i)}})}})$;
      $D_{t+1}[i] = D_{t+1}[i] \times \exp(\beta(i))$;
    end
    Normalize weights: $D_{t+1} = D_{t+1} / \sum(D_{t+1})$;
  end
  Add the weak classifier h_t with weight α_t to the ensemble H;
end

Algorithm 1: Denoised Adaboost Algorithm

3 Experiment

3.1 Datasets

We use multivariate Gaussian distribution to generate basic binary classification datasets of varying difficulties (overlapping dataset and non-overlapping dataset), and then randomly change a certain percentage of labels to inject the noise. Overlapping data sets make noise identification more difficult. Non-overlapping dataset means that the data points we generate are separable and there is no overlap among different classes, as shown in Figure 1. Overlapping dataset means that there exists overlap among different classes, as shown in Figure 2. Non-overlapping dataset is a more simpler case for Adaboost to classify data with different labels, and overlapping dataset is a more complicated case.

3.2 Experiment Results

We apply our denoised Adaboost algorithm to the artificial overlapping dataset and non-overlapping dataset with noise ratio 5%, 10%, 15%, 20%, 25% respectively, and compare its classification accuracy to XGboost, which is an advanced boosting algorithm for classification. In our method, we select k=10 for kNN, create 50 subclassifiers, and update the noise weight when iterations is greater

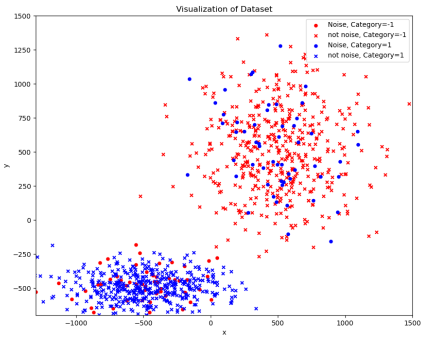

Figure 1: non-overlapping dataset

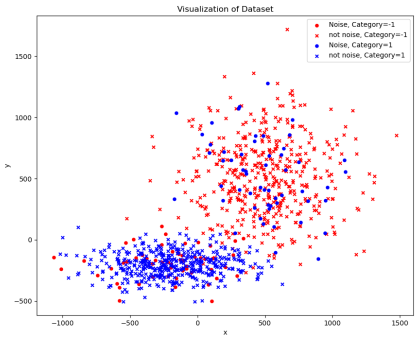

Figure 2: overlapping dataset

than 10 and divisible by 3. The XGboost algorithm is called from library directly.The result is shown in Tabel 1 and Tabel 2 below:

Table 1: Accuracy of denoised Adaboost and XGboost applying overlapping dataset with different noise ratio

	Noise Ratio				
Algorithm	5%	10%	15%	20%	25%
Denoised Adaboost	0.939	0.878	0.886	0.794	0.753
XGboost	0.927	0.856	0.853	0.772	0.716

Table 2: Accuracy of denoised Adaboost and XGboost applying non-overlapping dataset with different noise ratio

	Noise Ratio				
Algorithm	5%	10%	15%	20%	25%
Denoised Adaboost	0.942	0.874	0.832	0.759	0.704
XGboost	0.938	0.86	0.819	0.765	0.71

The results show that the classification accuracy of Adaboost algorithm after de-noising is always higher. When dealing with overlapping data sets, it is more accurate than XGboost, indicating that our algorithm is efficient. However, we found that we performed slightly worse than the XGboost algorithm when applying non-overlapping data sets and when the noise ratio was high. One probable reason is that our algorithm still lacks global fitting within classes compared to XGboost, so may perform worse than XGboost in this case.

4 Conclusion

In this study, we try to add dynamic prediction of noise weights to the AdaBoost algorithm to minimize the impact of noisy data. We introduce local consistency and preference in the processing of data, which provides the model with a reduced impact of noise in the presence of noise. Local consistency allows us to determine the noise likelihood of each data point within its neighborhood, while preference further takes into account the global noise likelihood. In the future, we could develop a set of methods to determine the characteristics of the dataset in order to select appropriate preprocessing and modeling strategies. For example, by determining whether a dataset is overlapping or not, we can decide whether or not to apply our denoised AdaBoost method. In addition, for highly noisy datasets, the noise reduction method can be further optimized, and it may be necessary to adjust the parameters or try other noise reduction techniques to adapt to the noise level of different datasets. Such careful analysis and customization can make the project more targeted and improve the model's adaptability and performance for different datasets.

References

[1] Dohyun Lee & Kyoungok Kim (2022) Improved noise-filtering algorithm for AdaBoost using the inter-and intra-class variability of imbalanced datasets. *Journal of Intelligent & Fuzzy Systems 43* pages 5035–5051.

[2]Markus M. Breunig & Hans-Peter Kriegel & Jörg Sander (2000) LOF: identifying density-based local outliers. *2000 Acm Sigmod International Conference on Management of Data.* pages 93–104.

[3] Rocco A. Servedio (2003) Smooth Boosting and Learning with Malicious Noise. *Journal of Machine Learning Research 4.* pages 633-648

[4] Pokrajac D. & Lazarevic A. & Latecki L.J. (2007) Incremental Local Outlier Detection for Data Streams *IEEE Symposium on Computational Intelligence & Data Mining*

[5] Lou X.J. & Sun Y.X. & Liu H.T. (2013) Clustering boundary over-sampling classification method for imbalanced data sets. *J. Zhejiang Univ., vol. 47 no. 6* pages 944–950.

[6] Zhuo C. (2018) Xgboost classifier for DDoS attack detection and analysis in SDN-based cloud. *Proc. IEEE Int. Conf. Big Data Smart Comput.* pages 1–9

[7] Zhang Z.X. & Chen Y.G. (2017) Improvement of AdaBoost Algorithm Based on Sample Noise Detection. East China Normal University

[8] Robert E.S & Yoram S (1999) Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning 37* pages 297–336.

[9] Duan L & Xiong D & Lee JA (2006) Local Density Based Spatial Clustering Algorithm with Noise. *Systems, Man and Cybernetics, 2006.* IEEE International Conference on.IEEE,2006.DOI:10.1109/ICSMC.2006.384769.

[10] Tsalera E. & Papadakis A. & Samarakou M. (2020) Monitoring, profiling and classification of urban environmental noise using sound characteristics and the KNN algorithm. *Energy Reports.* pages 223-230.

[11] Cao J. & Kwong S. & Wang R. (2012) A noise-detection based AdaBoost algorithm for mislabeled data. *Pattern Recognition.* 45(12): 4451-4465.

[12] Sun B. & Chen S. & Wang J. & (2016) A robust multi-class AdaBoost algorithm for mislabeled noisy data. *Knowledge-Based Systems.* 102: 87-102.

[13] Modarres Z. G. & Shabankhah M. & Kamandi A. (2020) Making AdaBoost Less Prone to Overfitting on Noisy Datasets. *2020 6th International Conference on Web Research (ICWR).* pp. 251-259, doi: 10.1109/ICWR49608.2020.9122292.