A Study of Learning with Noisy Labels

Jianmei Ye,
Master of Computer Science, Fulton School of Engineering, ASU
jianmei2017@asu.edu

Abstract— The primary objective of this research was to study and understand the binary classification problem "in the presence of random class-conditional noise." [1] More specifically, in the learning of such random labels that "have independently been flipped with some small probability." [1] The goals were: 1) to study a methodology that can improve the effectiveness and efficiency of learning random binary classification problem; 2) to study and implement method that has high accuracy even when the labels are highly corrupted ;3) to create synthetic dataset for training and testing purpose as well as the validation and analysis. To achieve the above goals, I study a state-of-art algorithm from the paper "Learning with Noisy Labels" [1], which is the first one providing "guarantees for risk minimization under random label noise without any assumption on the true distribution." [1] I will explain the main ideas of the paper and provide the experiment to validate and reproduce the results. Moreover, I developed an implementation of the first proposed method. Through this study, I learned and practiced an efficient methodology with provable guarantees for learning under label noise.

Index Terms— Binary Classification, Class Conditional, Empirical Risk Minimization, Random Noise

1 Introduction

IN classification problem, supervised learning algorithms use the correct labels as it is the ground truth for training and testing. Given this importance setting, this research is to study a state-of-art approach such that can deal with problems when the corrupted labels for supervised learning algorithms.

"In the survey article by Nettleton et al. [2010], there are many practical works has been done to study the learning from nosy labels." Addition to those works from noise-free PAC model to RCN model, the paper "Learning with Noisy Labels" (abbreviated LNL) consider the importance of the risk minimization in the presence of class-conditional random label noise [1]. LNL provides two methods address the problem. I study and implement the first method in LNL. Based on the implementation, I reproduce the result and shows similar output as in the experiment section of the paper.

I introduce the methodology of the first approach from LNL in Section 2. In Section 3, I give the implementation of the method of unbiased estimators with the discussion of problem encountered and related works. I present reproduced experimental results on synthetic data sets in Section 4 and conclusion in Section 5.

2 Description of Methodology

2.1 Methodology Overview and Interpretation

The goal of the methodology is to deal with the situation when the binary classification training label is not reliable and to ensure that the average risk is minimized.

As stated in the first proposed method in LNL, "the learning algorithms will draw iid samples from a noisy data D_ρ of a "clean" distribution $\mathfrak{D}$, where the noise rate ρ ($\boldsymbol{\rho_{+1}} + \boldsymbol{\rho_{-1}} < 1$) depends on the class label."

We will need to pick a classification method with loss function that satisfied the conditions that minimized the average rick. The approach uses the modified loss as an unbiased estimate. And" the method of the unbiased estimator uses the noise rates to construct an unbiased estimate for the loss function."

"The algorithm will tune the noise rate parameter through cross-validation." [1] The lemma 1 in LNL give us a way to construct an unbiased estimator of the loss function through the noisy label.

- ***Lemma 1 in LNL.***

Let $\ell(\mathrm{t},\mathrm{y})$ *be any bounded loss function.*
Then, if we define,

$$\tilde{\boldsymbol{\ell}}(\mathbf{t},\mathbf{y}) := \frac{\left(\mathbf{1}-\boldsymbol{\rho}_{-\boldsymbol{y}}\right)\boldsymbol{\ell}(\boldsymbol{t},\boldsymbol{y}) - \boldsymbol{\rho}_{\boldsymbol{y}}\boldsymbol{\ell}(\boldsymbol{t},-\boldsymbol{y})}{\mathbf{1}-\boldsymbol{\rho}_{+\mathbf{1}}-\boldsymbol{\rho}_{-\mathbf{1}}}$$

we have, for any t,y, $\mathbb{E}_{\tilde{y}}\left[\tilde{\ell}(\mathrm{t},\tilde{y})\right] = \ell(\mathrm{t},\mathrm{y})$
where
$\tilde{\ell}(\mathrm{t},\tilde{y})$ *denotes a suitably modified* ℓ for use with noisy labels
ρ_{+1} *and* ρ_{-1} *are known noise rate,*
$\tilde{y}$ *denotes noisy labels*

The section of "problem setup and background" in the LNL also mentions few important quantities that related to the key Lemma.

1) Let $f: \mathcal{X} \to \mathbb{R}$ be some real-valued decision function

2) The risk of f w.r.t the 0-1 loss is given by

$$R_D(f) = \mathbb{E}_{(X,\ Y)\sim D\,[1\{sign(f(X)\neq Y\}]}$$

3) Empirical $\tilde{\ell}$ − risk on the observed samples

$$R_{\tilde{\ell}}(\tilde{f}) := \frac{1}{n}\sum_{i=1}^{n}\tilde{\ell}\,(f(X_i),\widetilde{Y_i})$$

"A good predictor of the noise label can be learned by minimizing the sample average" [1]

$$\bar{f} \leftarrow \underset{f \in \mathcal{F}}{argmin}\, \hat{R}_{\tilde{L}}(f) \qquad (*)$$

Thus, we then can know that for any fixed $\boldsymbol{f} \in \boldsymbol{\mathcal{F}}$, the (*) sample average converges to $\boldsymbol{R_{\ell,D}(f)}$ the $\boldsymbol{\ell - \text{risk}}$ under the "clean" distribution $\mathfrak{D}$ by the unbiased $\tilde{\boldsymbol{\ell}}$ obtained from Lemma 1. Although the estimate is computed using noisy label, the predictor depends on the true labels.

2.2 Related Work

Follow the path of learning in LNL; I also study the related work of Bylander of many noise tolerant version of the perceptron algorithms. To extend my understanding of noisy label application, I also study the latest study of "Learning from noisy label with deep neural networks ."[Sainbyar 2017]
And "Learning Deep networks from noisy labels with Dropout regularization" [2]. However, when considering of binary classification problem, the approach in LNL is still the most efficient way for such setting.

3 Description of Implementation

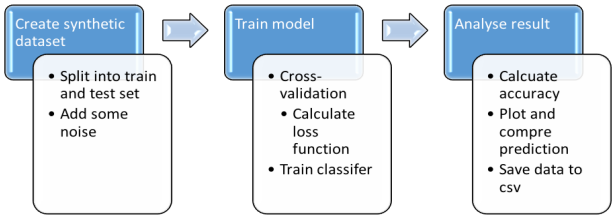

Fig. 3.1. The workflow for the implementation

3.1 Assumptions and Reason

One of the advantages of the algorithm in LNL is that there are no assumptions on the true distribution of $\mathfrak{D}$, which give us the freedom to create random dataset for training and validation. But for the problem, the random flip probability (ρ_{+1}, ρ_{-1}) is assumed to be known as they are essential to the definition for the class-conditional classification problem.

I chose SVM as the classifier. "SVM is a robust supervised machine learning algorithm that used for classification or regression problems. It uses techniques called kernel techniques to transform the data and then find the best boundary between the possible outputs based on those transforms" [3]

3.2 Problem Encountered

A key problem that encounters through the implementation is how to pick the right K fold for the cross-validation. Lower K means cheaper cost and less variance but more bias, while higher k value cost more expensive with more variance but lower bias. I used k = 10 as it is often cited with the conclusion. [4]

3.3 Profiles for the Implementation

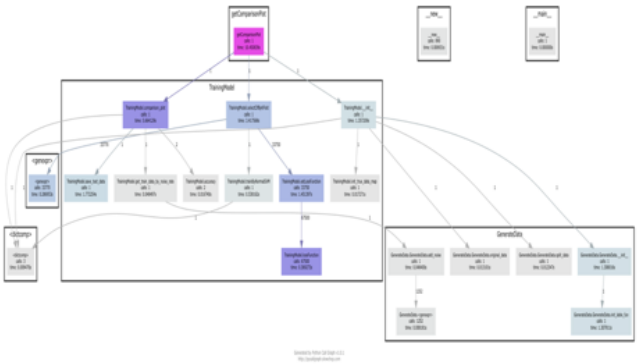

Fig. 3.3.1. Time usage profile for call workflow for training model for each synthetic dataset generated.

Call	Time
Generate data	**1.357209s**
Initialize	1.308016s
split data	0.012347s
add noise	0.048400s
Training model	**3.417568s**
cross-validation	1.451397s
compute loss fun	0.369273s
train classifier	0.538162s
Plot data	**5.664129s**
calculate accuracy	0.016740s
plot prediction	0.048497s
save data to csv	1.772254s

Table. 3.3.1. Time usage profile table for each stage call cost.

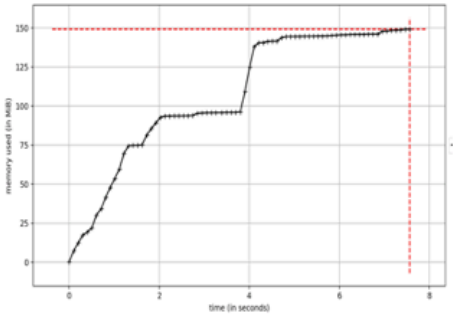

Fig. 3.3.2. Memory usage plot for the core function for training model for each synthetic dataset generated. The maximum memory cost 47mb increment.

Line #	Mem usage	Increment	Line Contents
203	94.2812 MiB	94.2812 MiB	@profile(precision=4)
204			def getComparisonPlot():
205	94.2812 MiB	0.0000 MiB	n, po1,po2 = 5000,0.1,0.5
206	95.7344 MiB	1.4531 MiB	test = TrainingModel(n,random=False)
207	95.7344 MiB	0.0000 MiB	train_data, test_data = test.train_set,test.test_set
208	142.2188 MiB	46.4844 MiB	clf= test.selectClfByKFold(train_data,test_data,po1,po2)
209	149.3594 MiB	7.1406 MiB	test.comparison_plot(clf,po1,po2)

Table. 3.3.2. Memory usage line profile information for training model for each synthetic dataset generated.

4 Experiment and Analysis

First, I implement two approaches to create the binary classification synthetic data. One for linearly separable dataset (Fig 4.1.1~ 4.1.3) similar to the data in Section 5.1 in LNL. The other one is a random non- linearly separable dataset (Fig 4.1.4 ~ 4.1.6).
Second, I implement an add noise method to add noisy to each set based on given noise rate ($\boldsymbol{\rho_{+1}}, \boldsymbol{\rho_{-1}}$).
Third, I split the dataset into the training set and test set (3:1) with random shuffler. After that, I run KFold cross-validation with on training set to find the best $f \in \mathcal{F}$, then use the subset as the final training set to train the classifier.
Finally, I use the trained classifier to predict the test set. The three plots in figures in section 4.1 are all from test set data; they are one of noise-free, the second one with noise rate, and the third one of prediction using the proposed method.

4.1 Experiment for synthetic data sets

Classification of the synthetic dataset using the implementation in Section 3 of LNL. The noise-free data displays in the leftmost panel. The middle plot show training data corrupted with noise rates, the rightmost plot show the accuracy that proposed algorithm achieved.

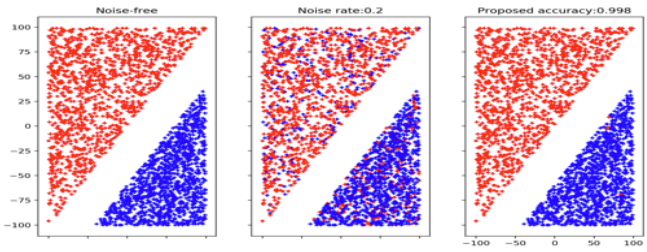

Fig. 4.1.1. Noise rate p+1 = p−1= 0.2, predicted accuracy = 99.8%.

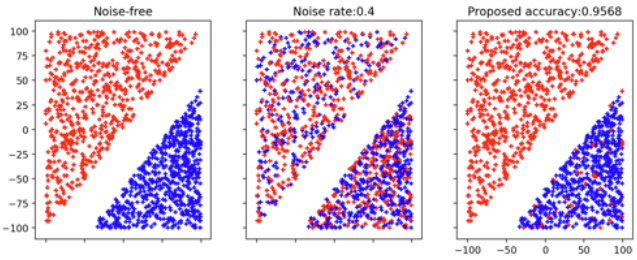

Fig. 4.1.2 Noise rate p+1 = p−1= 0.4, predicted accuracy = 95.68%.

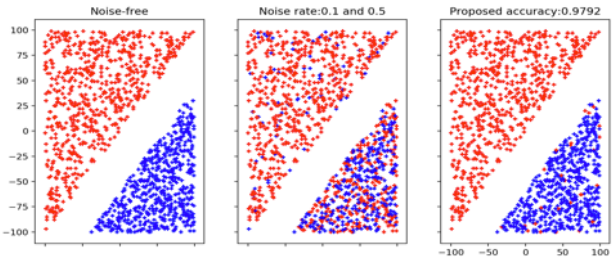

Fig.4.1.3. Noise rate p+1 = p−1= 0.5, predicted accuracy = 97.92%.

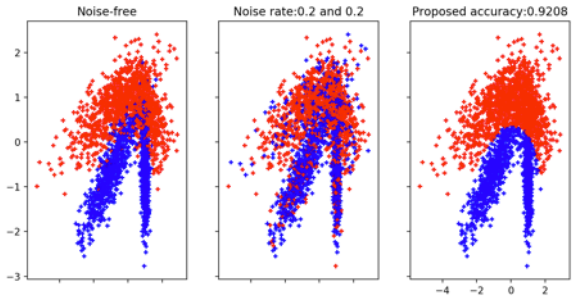

Fig.4. 1.4 Noise rate p+1 = p−1= 0.2, predicted accuracy = 92.08%.

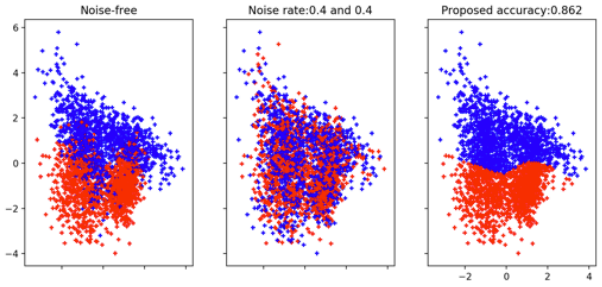

Fig.4.1.5. Noise rate p+1 = p−1= 0.4, predicted accuracy = 86.2% accuracy even at 0.4 noise rate per class.

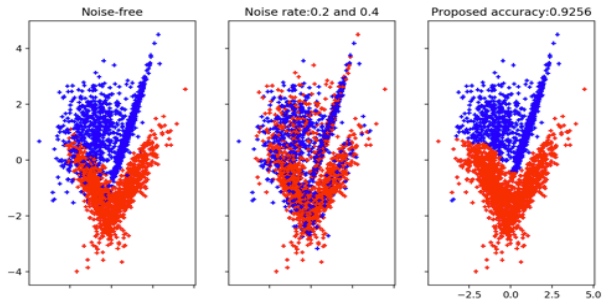

Fig. 4.1.6. Noise rate p+1 = 0.2, p−1= 0.4, predicted accuracy = 92.56%.

4.2 Results for comparison

Data set	d	n+	n-	p+1	p-1	accuracy
1	9	1000	1000	0.2	0.2	0.502
				0.3	0.1	0.504
				0.4	0.4	0.458
2	8	4000	4000	0.2	0.2	0.7005
				0.3	0.1	0.5065
				0.4	0.4	**0.86**
3	5	1500	1500	0.2	0.2	0.4573
				0.3	0.1	**0.932**
				0.4	0.4	**0.864**
4	29	5000	5000	0.2	0.2	**0.9292**
				0.3	0.1	0.8592
				0.4	0.4	0.7576
5	13	750	750	0.2	0.2	**0.92**
				0.3	0.1	0.4933
				0.4	0.4	0.5013

Table 4.2.1. Create 5 Random class balanced synthetic data set and compare with each accuracy, applying implementation in Section 3.

data set	d	n+	n-	p+1	p-1	accuracy
1	9	1400	600	0.2	0.2	0.698
				0.3	0.1	0.68
				0.4	0.4	0.696
2	8	4800	3200	0.2	0.2	0.611
				0.3	0.1	**0.8195**
				0.4	0.4	0.6075
3	5	2250	750	0.2	0.2	0.7507
				0.3	0.1	0.7573
				0.4	0.4	0.62
4	29	5500	4500	0.2	0.2	0.5448
				0.3	0.1	0.558
				0.4	0.4	0.5036
5	13	600	900	0.2	0.2	0.5813
				0.3	0.1	0.376
				0.4	0.4	0.5227

Table 4.2.2. Create 5 Random class weigh balanced synthetic data set and compare with each accuracy, applying implementation in Section 3. Use the same noise rate and dimension parameter as in Table1.

5 Discussion and Conclusion

In Section 4, I validated the implantation on the several different synthetic data with different noise rate combinations. The proposed method works out extremely well for large sample size (n>=5000), it can achieve > 92% accuracy even when 40% of label is corrupted. And it seems work better when the label is balanced. (Table4.2.1~4.2.2)

This research is a comprehensive study and implementation of the proposed method I in LNL. The proposed algorithms addressed the effectiveness of risk minimization, and it performs competitive outcome at the situation when the random classification noise is present.

6 Future Scope

LNL also offer another method for dealing with weighted 0-1 loss under nosy distribution. What's more, the paper also rises open problem for other noise models, such Constant-partition classification noise for discussion. I will study and implement the second method and do analysis on different data set for further understanding for the noisy learning problem.

References

[1] "Nagarajan N, Inderjit S D, Pradeep K R, and Ambuj T. Learning with noisy labels. In Advances in neural information processing systems, pages 1196{1204, 2013."

[2] Ishan Jindal, Matthew Nokleby, Xuewen Chen, "Learning Deep Networks from Noisy Labels with Dropout Regularization," IEEE 16th International Conference on Data Mining, 2016

[3] Greg, "Why use SVM?", http://blog.yhat.com/posts/why-support-vector-machine.html. 2017

[4] Ron Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," International Joint Conference on Artificial Intelligence (IJCAI), 1995