Automatic Data Cleaning

Darrell Aucoin (1003562316), Nicolas Brandt (1003649386)
Department of Computer Science
University of Toronto
27 King's College Cir, Toronto, ON M5S
{daucoin, nbrandt}@cs.toronto.edu

Abstract

In this paper, we will delve into the problem of working with noisy labels, and detecting which labels may be misclassified. This extends the results provided in Joan Bruna et al. paper [1] on how to correct noisy labels by trying their method on a logistic regression model. We found that their method is robust against heavily mislabelled data ($\simeq 40\%$). We also introduce a procedure to deal with non-recorded classes. We saw that our method could generate a significant increase in accuracy but at the cost of a longer training time.

1 Introduction

With the increase in the size of datasets, it's becoming harder and harder to make sure that each data point is labelled correctly. That's why, recent studies [1][2] are aiming to automatically detect and correct mislabelled data points. Joan Bruna et al. [1] proposed that the labels should be considered as "noisy labels" and to find the true labels given those labels. This idea has already been tested for convolutional networks. The aim of this paper is to check how well this method can be generalized to other models, particularly to logistic regression. Moreover, we are proposing a new idea on how to handle cases when we have more true classes than the recorded classes.

For our experiments, we used the MNIST dataset. In order to corrupt the dataset, we created a probability matrix θ where each element $\theta_{i,j}$ corresponds to the probability of the true label j being modified into the corrupted label i.

2 Related Works

Our project aims to extend the research done by Joan Bruna et al. (2015) [1] concerning how to handle noisy labels in convolutional networks. They mentioned that mislabelled data mostly originates from two sources: label flips where a label is erroneously been given one label instead of another and outliers where the misclassification comes from the fact that the number of true classes is greater than the number of recorded classes. For both cases, they proposed a solution with convolutional networks as well as experimental results. In our paper, we will try to see how well their method can be generalized to logistic regression.

During our research, we also relied on Natarajan et al. (2013) [2] research on binary classification with random classification noise. Their use of a surrogate cost function, that can be expressed by a weighted sum of the original cost functions, allowed them to reach a good accuracy ($\cong 88\%$) even with heavily corrupted data (40% of misclassification).

Finally, Feney & Verleysen survey [3] has shown that logistic regression models usually seem more robust than other models concerning the deterioration of classification performances as a result of noisy labels. They also proposed different kinds of variants which could better handle datasets with corrupted labels.

3 Experimental Model

We used a latent variable model, similar to the work by Joan Bruna et al. paper [1]. For each digit image there is an unobserved true label r, which is misclassified to label c with probability $p(\mathbf{c} \mid \mathbf{r}) = \theta_{c,r}$. Although misclassification is likely dependent on the shape of the written digit, we will assume that misclassification is only dependent on the true label r. Our model can then be expressed as:

$$\begin{aligned} p(c = i \mid \mathbf{x}, \mathbf{w}) &= \sum_j p(c = i \mid r = j)\, p(r = j \mid \mathbf{x}, \mathbf{w}) \\ &= \sum_j \theta_{i,j} p(r = j \mid \mathbf{x}, \mathbf{w}) \end{aligned}$$

While $p(r = j \mid \mathbf{x}, \mathbf{w})$ can theoretically be any probability based classifier, for our experiments we will be using logistic regression.

3.1 Fixed $\theta_{c,r}$

For a baseline, we consider a fixed, and known, $p(\mathbf{c} \mid \mathbf{r}) = \theta_{c,r}$ and learn $p(r \mid \mathbf{x}, \mathbf{w})$. After training, this latent model had a higher accuracy and average predictive log-likelihood than the more naive model that assumes $r = c$.

$$\begin{aligned} p(c \mid \mathbf{x}, \mathbf{w}) &= \sum_{r=0}^{9} p(c \mid r)\, p(r \mid \mathbf{x}, \mathbf{w}) \\ &= \sum_{r=0}^{9} \theta_{c,r} \frac{\exp\left\{\mathbf{w}_r^\top \mathbf{x}\right\}}{\sum_{r'=0}^{9} \exp\left\{\mathbf{w}_{r'}^\top \mathbf{x}\right\}} \\ \ell(\mathbf{w}) &= -\log \sum_{r=0}^{9} \exp\left\{\log \theta_{c,r} + \mathbf{w}_r^\top \mathbf{x} - \log\left(\sum_{r'=0}^{9} \exp\left\{\mathbf{w}_{r'}^\top \mathbf{x}\right\}\right)\right\} \end{aligned}$$

3.2 Unknown $p(c \mid r)$

In practice we will likely not know the probability of mislabelling an item to c given the true label r, as such we will have to learn $p(\mathbf{c} \mid \mathbf{r}) = \frac{\exp\{\eta_{c,r}\}}{\sum_{c'=0}^{9} \exp\{\eta_{c',r}\}}$ along with $p(r \mid \mathbf{x}, \mathbf{w})$.

$$\begin{aligned} p(c \mid \mathbf{x}, \mathbf{w}) &= \sum_{r=0}^{9} \frac{\exp\{\eta_{c,r}\}}{\sum_{c'=0}^{9} \exp\{\eta_{c',r}\}} \frac{\exp\left\{\mathbf{w}_r^\top \mathbf{x}\right\}}{\sum_{r'=0}^{9} \exp\left\{\mathbf{w}_{r'}^\top \mathbf{x}\right\}} \\ \ell(\mathbf{w}, \eta) &= -\log \sum_{r=0}^{9} \exp\left\{\eta_{c,r} + \mathbf{w}_r^\top \mathbf{x} - \log\left(\sum_{c'=0}^{9} \exp\{\eta_{c',r}\}\right) - \log\left(\sum_{r'=0}^{9} \exp\left\{\mathbf{w}_{r'}^\top \mathbf{x}\right\}\right)\right\} \end{aligned}$$

However since this model is not convex, we initially started training assuming $r = c$ to give good initial weights for $\mathbf{w}_r$ and then relaxed that assumption to train on both $\eta_{c,r}$ and $\mathbf{w}_r$.

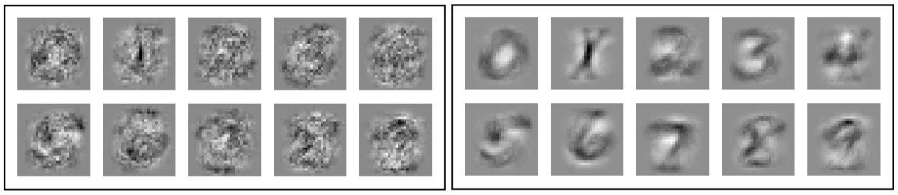
4 Experimental Results

In order to compare our results with the one obtained by Joan et al. [1], we used their method on a logistic regression model. We corrupted 40% of our training labels and ran four different models:

- A classic logistic regression model without corrupted labels
- A classic logistic regression model with the corrupted labels
- A "denoising" logistic regression model with θ known

- A "denoising" logistic regression model with θ unknown

The following results have been reached for 10,000 training samples and 10,000 test samples:



On the left, we can see the weights learned by a logistic regression model. On the right side, the weights learned by the latent model.

Figure 1: Logistic Regression Weights

Model	Corruption %	Test Accuracy	Test Log-likelihood
Logistic	0.%	93.2%	-0.227
Logistic	38.23%	78.1%	-0.732
Latent (fixed θ)	38.23%	92.4%	-0.266
Latent (unknown θ)	38.23%	91.9%	-0.272

Table 1: Comparison of the accuracy and the likelihood obtained for different models

We can see that the use of the latent model improved the test accuracy and the log-likelihood compared to the classical model. The accuracy is almost the same as if we didn't have any corrupted labels. When learning θ we can see a small decrease in accuracy but the results are still quite close to the model without any corrupted labels.

Model	Regularizer	Corruption %	Test Accuracy	Test APL
	$\frac{1}{2\sigma_{\mathbf{w}}^2}$			$p\left(r \mid \mathbf{x}, \mathbf{w}\right)$
Logistic	0.	0.	88.5%	-1.107
Latent (MAP)	0.	0.	88.1%	-1.923
Logistic	0.	6.5%	79.0%	-1.000
Latent (MAP)	0.	6.5%	84.9%	-1.870
Logistic	0.	38.23%	67.9%	-1.223
Latent (MAP)	0.	38.23%	76.4%	-2.520
Logistic	6	0.	90.8%	-0.389
Latent (MAP)	6	0.	91.2%	-0.306
Logistic	6	6.5%	89.7%	-0.473
Latent (MAP)	6	6.5%	91.2%	-0.327
Logistic	6	38.23%	82.6%	-0.948
Latent (MAP)	6	38.23%	81.3%	-0.845

Table 2: Influence of the choice of the prior on the weights on the accuracy of the models

We also studied the influence of the prior on the weights for a non-binarized dataset. As shown in table 2, we saw that finding a good prior improved slightly the results ($\approx 3\%$) for a non-corrupted dataset but was more and more important as the corruption rate of the labels increased. This means that the choice of the prior is really an important factor in order to obtain a model with a good accuracy.

As mentioned in the theoretical part, learning both the weights and θ at the same time can result to different results from what we are expecting leading to a drop in accuracy. Thus, it is really important to fine-tune the learning rates of this element with a validation set.

5 Finding Misclassified or Non-Recorded Classes

Next we will consider classification where all the data are classified correctly except for the classes which aren't recorded. The images belonging to these classes are uniformly distributed among the other classes. Joan Bruna et al. [1] solution was to add a few examples of these classes with the right label to the training set and to use the previous method.

In this paper, we decided to use another method. We will consider that we have one non-recorded class and that all the images belonging to this class are uniformly distributed among the other classes. Now, if we run logistic regression and retrieve the highest probability of the classes for all images, we obtain the following figure 2.

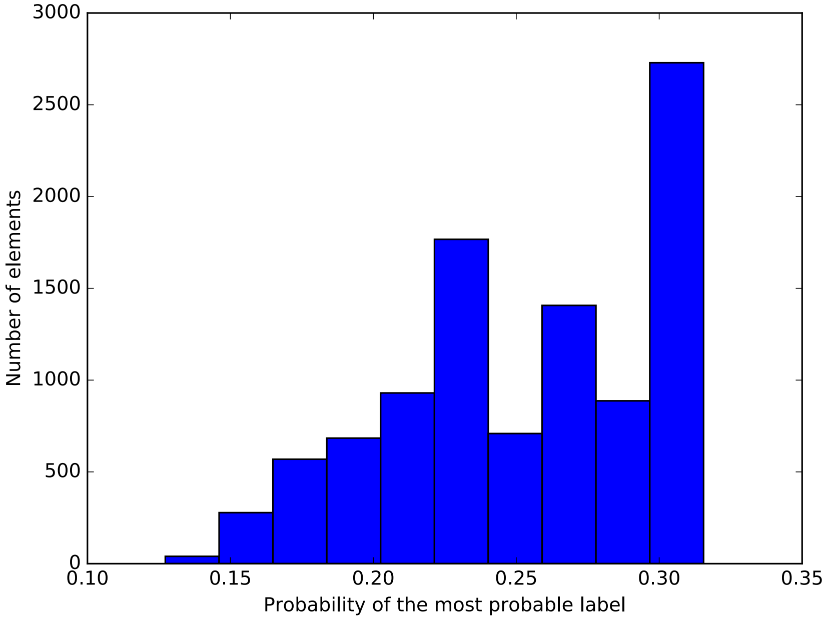


Figure 2: Most of the images are predicted as belonging to a class with a strong certainty (> 0.22)

Our idea is that our model was trained in order to uniformly distribute the images belonging to the non-recorded class. As such, most of those pictures will most likely be predicted with a low certainty. Thus, if we can create a threshold t, we can consider that if the certainty is greater than t, then the image belong to the predicted class, else it belongs to the non-recorded class. Of course, this method is not really accurate and we will make a lot of label flips. Nevertheless, once we have our new labels, we can use the method mentioned in the previous part to remove the noises.

This method has a few advantages and drawbacks. The threshold is easy to estimate and this method doesn't require us to find more training data. Nevertheless, it can only be used correctly when only one true class has not been recorded and the model will be two times longer to train.

5.1 Results

For this part, we considered the same noisy label MNIST dataset. Moreover, we removed all the images belonging to class 9 and we uniformly distributed them into the other classes. We decided to use a threshold of 0.22. We tested the results for each of the previous models as well as our "outliers" model.

We can see that, as we expected, latent models are not able to retrieve alone the non-recorded class. That's why we can see a drop of around 9% of accuracy if we compare their result to the previous part. However, we can see that, without adding any data points, our last model gives better results. Nevertheless, it's important to notice that when we reduced the value of the threshold of only 0.005 the accuracy fell to 88.5%. Thus, this model requires to precisely fine-tune this hyperparameter.

Model	Corruption %	Test Accuracy	Test Log-likelihood
Logistic	0.%	93.2%	-0.227
Logistic	38.23%	71.5%	-1.502
Latent (fixed θ)	38.23%	83.6%	-0.579
Latent (unknown θ)	38.23%	83.4%	-1.13
Outliers model	38.23%	89.4%	-0.337

Table 3: Comparison of our proposed model with the previous models

5.2 Predicting r From $p\left(r \mid c, \mathbf{x}, \mathbf{w}\right)$

From training $p\left(c \mid \mathbf{x}, \mathbf{w}\right)$ we learned $p\left(c \mid r\right)$ and $p\left(r \mid \mathbf{x}, \mathbf{w}\right)$. We also assumed that c and $\mathbf{x}$ (and $\mathbf{w}$) are independent for some given r. This is also how we corrupted the labels for the MNIST dataset.

$$p\left(c, r \mid \mathbf{x}, \mathbf{w}\right) = p\left(c \mid r, \mathbf{x}, \mathbf{w}\right) p\left(r \mid \mathbf{x}, \mathbf{w}\right) = p\left(c \mid r\right) p\left(r \mid \mathbf{x}, \mathbf{w}\right)$$

Thus we can obtain the probability $p\left(r \mid c, \mathbf{x}, \mathbf{w}\right)$ which should give a higher probability of finding the true label r than $p\left(r \mid \mathbf{x}, \mathbf{w}\right)$.

$$\begin{aligned}
p\left(r \mid c, \mathbf{x}, \mathbf{w}\right) &= \frac{p\left(c \mid r\right) p\left(r \mid \mathbf{x}, \mathbf{w}\right)}{\sum_{r=0}^{9} p\left(c \mid r\right) p\left(r \mid \mathbf{x}, \mathbf{w}\right)} \\
\log p\left(r \mid c, \mathbf{x}, \mathbf{w}\right) &= \log \frac{p\left(c \mid r\right) p\left(r \mid \mathbf{x}, \mathbf{w}\right)}{\sum_{r=0}^{9} p\left(c \mid r\right) p\left(r \mid \mathbf{x}, \mathbf{w}\right)} \\
&= \log \frac{\exp\left\{\eta_{c,r}\right\}}{\sum_{c'=0}^{9} \exp\left\{\eta_{c',r}\right\}} \frac{\exp\left\{\mathbf{w}_r^\top \mathbf{x}\right\}}{\sum_{r'=0}^{9} \exp\left\{\mathbf{w}_{r'}^\top \mathbf{x}\right\}} - \log \sum_{r=0}^{9} p\left(c \mid r\right) p\left(r \mid \mathbf{x}, \mathbf{w}\right) \\
&= \begin{array}{r} \eta_{c,r} + \mathbf{w}_r^\top \mathbf{x} - \log\left(\sum_{c'=0}^{9} \exp\left\{\eta_{c',r}\right\}\right) - \log\left(\sum_{r'=0}^{9} \exp\left\{\mathbf{w}_{r'}^\top \mathbf{x}\right\}\right) \\ - \log \sum_{r=0}^{9} p\left(c \mid r\right) p\left(r \mid \mathbf{x}, \mathbf{w}\right) \end{array}
\end{aligned}$$

For each data point $\mathbf{x}$ and corrupt label c, we can get the probability of r for each record. From table 4, if we can see that if we choose $\hat{r}$ based on the highest probability of $p\left(r \mid c, \mathbf{x}, \mathbf{w}\right)$ for each class, there is considerable overlap with the true labels. Furthermore, if we take the element wise product of $p\left(r \mid c, \mathbf{x}, \mathbf{w}\right)$ and c, we can get a numerical value for how strongly our model believes each datapoint is mislabeled. This list of probabilities can be sorted, giving a priority queue for checking which images are mislabeled. Figure 3 shows the first 20 digits of MNIST that our algorithm says are possibly misclassified.

True Labels Untouched $(c = r)\,\%$	Concordance with the true labels $(\hat{r} = r)\,\%$	Number of Mislabeled $\lvert r \neq c \rvert$	Correctly Identifying Mislabeled $(\hat{r} \neq c) \wedge (r \neq c)$	Percentage of Mislabeled in $\hat{r} \neq c$
100%	99.0%	0	0	N/A
93.5%	97.8%	634	84.9%	84.86%
61.77%	94.5%	3841	91.3%	93.67%

Model run on all 60,000 MNIST training digits and metrics scored with 10,000 MNIST test set digits (with randomly corrupting the test set labels).

Table 4: Recovering true labels from $p\left(r \mid c, \mathbf{x}, \mathbf{w}\right)$

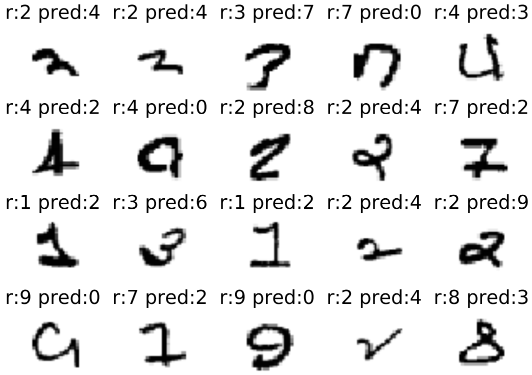


Figure 3: Possible Misclassified Digits

6 Limitations of Approach

It should be noted that $p\left(c \mid \mathbf{x}, \mathbf{w}\right)$ is difficult to learn as it's objective function is not convex. This can be easily seen by considering a fixed $\mathbf{x}, \mathbf{w}$, thus $p\left(c \mid x, w\right)$ can be thought of as a dot product of a matrix $A = p\left(c \mid r\right)$ and a vector $y = p\left(r \mid \mathbf{x}, \mathbf{w}\right)$ and some invertible matrix P.

$$\begin{aligned} Ay &= c \\ APP^{-1}y &= c \\ \left[AP\right]\left[P^{-1}y\right] &= c \end{aligned}$$

As long as the entries conform to the rules of probability (summing along the columns of AP, $P^{-1}y$ evaluates to 1 and entries are between 0 and 1), then P can absorbed into the parameters of our model without affecting c. Permutation matrices are an example of a group of matrices with this property. Since there are multiple optimal values of $\eta, \mathbf{w}$, then the loss function is not convex.

Thus, using a gradient descent method alone can't guarantee us to find the correct value of θ. In Joan Bruna paper[1], it was shown that under strong assumptions, it was possible to force the algorithm to converge toward the true value of θ by minimizing the trace. In practice, we found that if our prior value of theta wasn't far from its true value (e.g. Identity matrix) and with a good learning rate, we could obtain an accurate estimation.

7 Future Improvements

In this paper we assumed that a noisy label is independent of the picture of its number given its true class. However, we can see in the figure representing possible misclassified numbers that this assumption is not necessarily true in a real dataset. We will often find that misclassified numbers had an odd orientation or were written badly. Thus, instead of learning the value of $p\left(c \mid r\right)$, we could try to learn $p\left(c \mid r, \mathbf{x}\right)$. Of course, doing this would dramatically increase the number of parameters for our models to learn. One alternative is using PCA on the images in order to only keep the most important information. This new model should allow us to more accurately detect misclassified images while decreasing the number of parameters to learn.

8 Conclusion

In conclusion, we can say that the latent model proposed in Joan Bruna paper [1] for convolutional networks works well for logistic regression. Even when the data are heavily corrupted, the latent model can reach an accuracy close to a model obtained without noisy labels. The main drawbacks are a longer training time and more careful attention while learning $p\left(c \mid r\right)$ in order to train a good model.

We also proposed a new model which could handle cases where we have one more true class than recorded classes. This new model gave fairly good results compared to the latent model or a logistic regression model. However, it is only working properly under heavy assumptions like the fact that we can only have one non-recorded true class or that the images belonging to the non-recorded class should be roughly uniformly distributed among the other classes. Thus, it can only be used in a restricted number of cases.

Bibliography

[1] S. Sukhbaatar et al., “Training convolutional networks with noisy labels,” 2014.

[2] N. Natarajan et al., “Learning with noisy labels,” in *Advances in Neural Information Processing Systems 26*, 2013, pp. 1196–1204.

[3] B. Frenay & M. Verleysen, “Classification in the presence of label noise: A survey,” in *IEEE Transactions on Neural Networks and Learning Systems*, May 2014.