Robust Learning Models for Noisy Data

Menghan Chen
Department of Computer Science
University of Toronto
menghan.chen@mail.utoronto.ca

Qinghui Yu
Department of Computer Science
University of Toronto
qinghui.yu@mail.utoronto.ca

Abstract

Deep learning has taken off in recent years, thanks to an expanding computing capacity and growing quantity of data. However, training with noisy data often produces weak results. We attempt to solve this problem by fitting a probabilistic model that models the layer of noise between true labels and observed labels. We show that this type of method could be extended to fit beyond logistic classifiers and compares it to logistic regression - a traditional model suited for classification.

1 Introduction

Deep learning with large-scale training data has shown impressive results on image classification, natural language processing and many other machine learning tasks in recent years. Works like Krizhevsky et al. [2012], He et al. [2016], Oord et al. [2016] have revolutionized how most of the fundamental works are done in the industries. For most of these networks, a large amount of well-labeled data is required for good performance, which can be expensive and time-consuming to obtain. Therefore, it is impractical to assume all data are clean and without nonrandom noise. One consequence of this in deep learning is that systematically mislabeled or corrupted data might skew the learning such that the network becomes inconsistent. It is found in Zhu and Wu [2004], Frénay and Verleysen [2014] that predictive accuracy is lower when mislabeled data are present.

There are several types of noise on labels as described in Frénay and Verleysen [2014]. In general, there are random noises and nonrandom noises. Random noises are simply mismatches between images and labels which bears no relationship with the input, for example, white noise found in signal processing. Studies such as Long and Servedio [2010] have shown that performance of some popular machine learning algorithms like Boosting take a hit when random noise is present. However, random noise is often assumed to simply go away when size of the dataset grows without too much issue. On the other hand, nonrandom noises are caused by some poor or insufficient information that fails to perform reliable labeling, and are often correlated with the data itself, for example, in MNIST dataset, 9 and 0 could be mistaken for each other, or in an economic survey, individuals might have incentives to under-report their income. In this case, studies made on these data can have non-significant results.

Our goal in this paper is to build a probabilistic graphical model that simultaneously learns to classify data as well as guess which examples are mislabeled. For simplicity and practicality, we only consider nonrandom noise. Our experiments are based on the MNIST dataset by LeCun and Cortes [1998] which we believe quite perfectly showcases the type of noise we are modeling. Results for different probability models will be compared against each other, and we will assess the effectiveness of the model.

2 Related Work

Because noisy data present such a challenge to statistical modeling, much work have been done in the area of data cleaning. A broad overview of the methods can be found in Frénay and Kabán [2014]. There are several approaches to noisy labels, two of which are the most prominent: preprocess data to ensure its integrity, as shown in Teng [2001], Galhardas et al. [2000a], and building learning models to directly cope with uncertain labels (Daniel Paulino et al. [2003], Ladouceur et al. [2007]). Some models bridges the gap by trying to learn denoising the data Vincent et al. [2008]. We briefly summarize some of these approaches here.

2.1 Data Cleansing

This is a direct approach that detects and removes mislabeled instances. Many data cleansing methods are based on the assumption that the label errors are independent of the particualr model being fit to the data. For example, voting filtering is one of the purposed method in Brodley et al. [1996], in which an instance is removed if it cannot be classified by the majority of classifiers. Experiments with boosting filters are conducted in many studies, as shown in Verbaeten and Van Assche [2003]. Since boosting performs poorly on noisy data, and such mislabeled instances tend to have high weight in the boosting process. After few rounds of AdaBoost, examples with highest weights are removed. Classical approaches with SQL queries on a very large database is also used frequently in the data mining community, it is the approach used by Meng et al. [2016], Galhardas et al. [2000b].

2.2 Learning Algorithms

Some researchers on the other hand, build probabilistic models that take label noise into account. And prior information is necessary here in such identifiable issues, among which Beta priors and Dirichlet distributions are common choices, according to Gustafson et al. [2001]. Other approaches includes clusting, as shown in Bouveyron and Girard [2009]. It is based on the assumption that instances whose labels are inconsistent with the labels of nearby clusters are likely to be mislabelsed. In addition, some use modified version of non-probabilistic models to prevent instances from taking too large weights in neural networks and SVMs. Xiao et al. [2015] takes different types of noise into account and uses EM algorithm instead of gradient based optimizers with great success.

3 Formal Description

3.1 The Model

Denote x to be the input, w the weight in our logistic regression, r the true label, and c the observed noisy label. In our model, since we only care about nonrandom noise, we assume that there is a implicit mapping θ that adds noise to the true label r. The graphical model is depicted here.

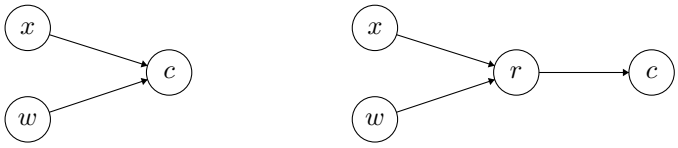

Figure 1: Traditional logistic regression model (left) and our model(right)

Because we only observe the mislabeled data, it is not possible to learn $p(\mathbf{r} \mid \mathbf{x}, \mathbf{w})$ directly, so instead, we must learn

$$p(\mathbf{c} \mid \mathbf{x}, \mathbf{w}) = \sum_{i=0}^{9} p(\mathbf{c} \mid \mathbf{r}) p(\mathbf{r} \mid \mathbf{x}, \mathbf{w}) \tag{1}$$

To train this model, we define 1 as the objective function, and use gradient ascent to optimize it. After we obtain the optimal $\mathbf{w} = \underset{\mathbf{w}}{argmax}\ p(\mathbf{c} \mid \mathbf{x}, \mathbf{w})$, we could do inference by computing

$$p(\mathbf{r} \mid \mathbf{x}, \mathbf{w}, \mathbf{c}) = \frac{p(\mathbf{r}, \mathbf{c} \mid \mathbf{x}, \mathbf{w})}{p(\mathbf{c} \mid \mathbf{x}, \mathbf{w})} \tag{2}$$

$$= \frac{p(\mathbf{c} \mid \mathbf{x}, \mathbf{w}, \mathbf{r}) \cdot p(\mathbf{r} \mid \mathbf{x}, \mathbf{w})}{p(\mathbf{c} \mid \mathbf{x}, \mathbf{w})} \tag{3}$$

$$= \frac{p(\mathbf{c} \mid \mathbf{r}) \cdot p(\mathbf{r} \mid \mathbf{x}, \mathbf{w})}{p(\mathbf{c} \mid \mathbf{x}, \mathbf{w})} \tag{4}$$

$$= \frac{\theta \cdot p(\mathbf{r} \mid \mathbf{x}, \mathbf{w})}{p(\mathbf{c} \mid \mathbf{x}, \mathbf{w})} \tag{5}$$

The idea behind the model is if we specify the correct θ, our classifier will adapt $\mathbf{w}$ such that the prediction it produces is as close to the true label as possible, which is then mapped to noisy label via θ. This way, we maximize likelihood of observing the noisy label. However, since we do not know the distribution of θ beforehand, we would have rely on some heuristics to specify it. It is also possible to learn the kernel through a gradient ascent with the same objective function, which we would implement and compare with a fixed θ.

The classification schema is also extensible. We could use a neural network to predict $p(\mathbf{r} \mid \mathbf{x}, \mathbf{w})$ instead of logistic regression, in fact, we could attach any type of classifier that outputs categorical probability.

3.2 Generating Noise

To train and test our model, we must obtain mislabeled samples, however, it is time consuming and impractical to go through thousands of images and hand label them. To ensure that the labels are reasonably wrong and exhibit nonrandom noise, we trained a poorly performing logistic regression to label the data. Power level of the labeler could be adjusted by simply tweaking its maximum iteration. We summarize the power level in table 1 below.

Max iter	train accuracy	test accuracy
1	0.685	0.697
2	0.767	0.778
3	0.843	0.854
4	0.881	0.889
5	0.899	0.905
6	0.909	0.913

Table 1: Accuracy of Labeler

Upon visual inspection, the mislabeled images are systematic and could easily be mistaken by a human labeler as well. In a blind test, we did not label some of these images correctly and predicted the same value as the logistic labeler. 30 wrongly labeled samples are plotted in figure 2.

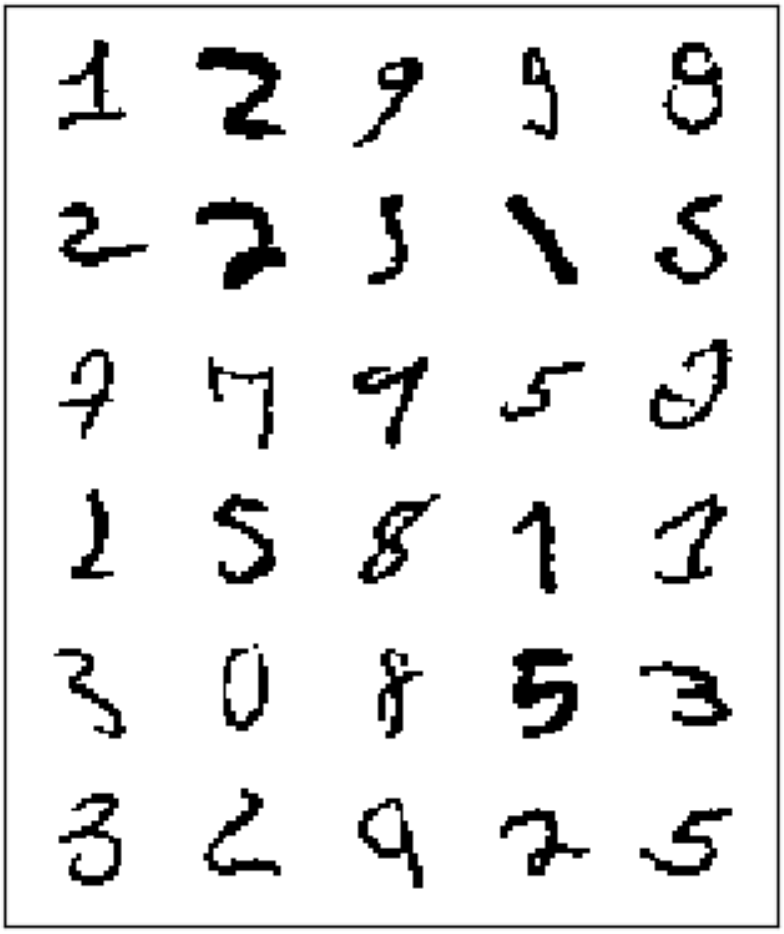

Figure 2: 30 Mislabeled Samples

Of these images, the true labels and predicted labels are:

$$True : [1, 2, 9, 9, 9, 2, 2, 5, 1, 5, 7, 7, 9, 5, 2, 2, 5, 8, 7, 1, 3, 0, 8, 5, 3, 3, 2, 9, 7, 5]$$
$$Mislabeled : [5, 7, 7, 5, 8, 6, 7, 3, 5, 0, 4, 9, 7, 8, 0, 1, 3, 5, 1, 7, 5, 6, 1, 8, 6, 5, 5, 4, 7, 3]$$

4 Results and Comparison

In this section, we present the results from running the model. Here, four models will be compared together at three different power levels: 1, 3, 5. We present the predictive accuracy on both training and validation data and graph the learned representation of true digits (w). We will be comparing logistic regression with a fixed prior $\theta \in \{I, 0.9I + 0.01, p_j I + \frac{1-p_j}{10}\}$, where I is the identity matrix and p_j is training classification rate of the labeler at power level j. This gives us a reasonable approximation on the distribution of classification errors. Also, throughout the training, $\lambda = 0.1, epoch = 500$ for logistic regression.

4.1 Power level 1

As expected, with $j = 1$, the performance of the classifiers are quite poor, however, when considering about 40% of the training labels are wrong, the result is rather satisfying. Below are the accuracies for each models. We can see from the result that the choice of θ affects the accuracy rates to some minimal extent, and $\theta = 0.9I + 0.01$ slightly outperforms the other groups. This indicates that $\theta = 0.9I + 0.01$ is close to the true distribution of $p(c|r)$. To further test our hypothesis, we examined

the distribution of train labels given by noisy dataset and the true labels, and get a true prior matrix M. The entries on the diagonal of M have values roughly equal to 0.9, and all other entries are close to 0, which confirms our hypothesis. Another point to note is that $5, 8, 9$ are not identified, and one important reason behind that is there are no five in the sample and only 4 eights, this means that with our model, it is impossible to predict these classes. Since the true labels are quite balanced, our accuracy is capped at about 70%.

θ	train accuracy	test accuracy
I	0.6397	0.6018
$0.9I + 0.01$	0.6404	0.6050
$0.7I + 0.03$	0.6402	0.6028
$train$	0.6407	0.6060

Table 2: Accuracy of different models at $j = 1$

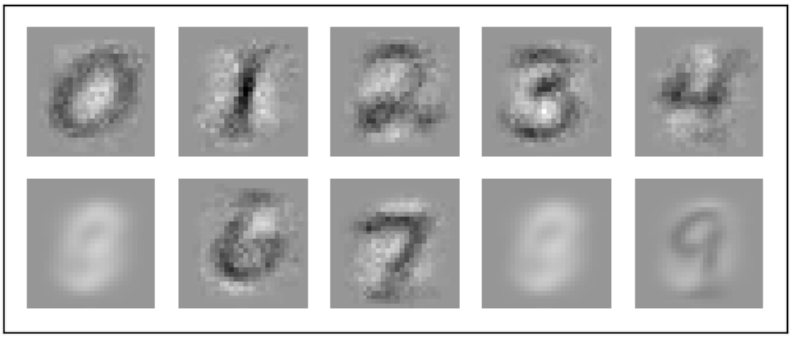

Figure 3: Learned w for $j = 1$

4.2 Power level 3

Similar to the case when $j = 1$, $j = 3$ produces unidentifiable fives, however, both 8 and 9 are seen in figure 4, along with it comes a much higher classification accuracy. This is as expected, since when $j = 3$, the labeler produces 700 and 900 of fives and eights respectively, which solves the no-sample issue when the labeler adds too much noise. However, five is still not recognized by the network. We speculate that because fives look like a lot of other digits such as 2, 3, 6, 7, 8, it is hard to produce an internal representation of it given noisy labels. From table 3, we can see that no particular group possesses a noticeable gain over the identity kernel $\theta = I$, which could be telling that the effect of mislabeling is not as disruptive as expected.

θ	train accuracy	test accuracy
I	0.8078	0.7746
$0.9I + 0.01$	0.8077	0.7758
$0.8I + 0.02$	0.8074	0.7758
$train$	0.8085	0.774

Table 3: Accuracy of different models at $j = 3$

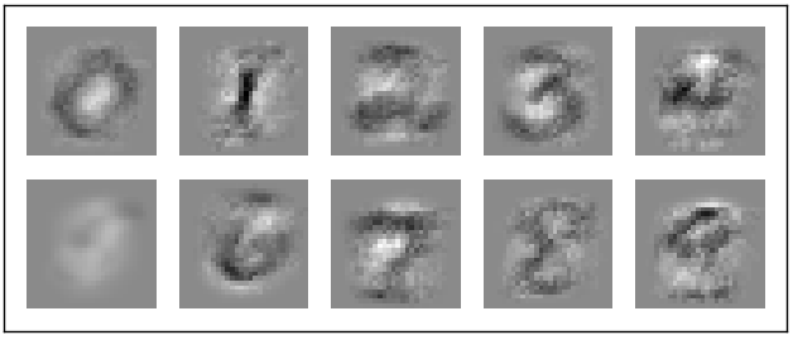

Figure 4: Learned w for j=3

4.3 Power level 5

For $j = 5$, we dropped the case where $\theta = p_j I + \frac{1-p_j}{10}$ since $p_j = 0.9$. Here, we have a moderate 3% gain over the case when $j = 3$, however, five remains unidentifiable. The improvement in accuracy is largely due to having much better training labels, as indicated in figure 5 by the darked feature points. Also similar to $j = 3$, the results of each choice of θ is inconclusive, this is not surprising as all groups identified the same digits. We also plotted the mispredicted digits by the network, see figure 6, it is the same story as figure 5, most of the mispredicted digits are fives, with a large amount of them classified as a six. Also one thing to note is that $argmaxp(\mathbf{r} \mid \mathbf{x}, \mathbf{w}) = argmaxp(\mathbf{c} \mid \mathbf{x}, \mathbf{w})$ for all training data. This means that even with the presence of map θ, our model thinks the noisy label is the same as the real label. This could be because our linear map θ is underfitting the true noise map. More about the effect of θ will be discussed in the next section.

θ	train accuracy	test accuracy
I	0.8388	0.8048
$0.9I + 0.01$	0.8378	0.8046
$train$	0.8392	0.8022

Table 4: Accuracy of different models at $j = 5$

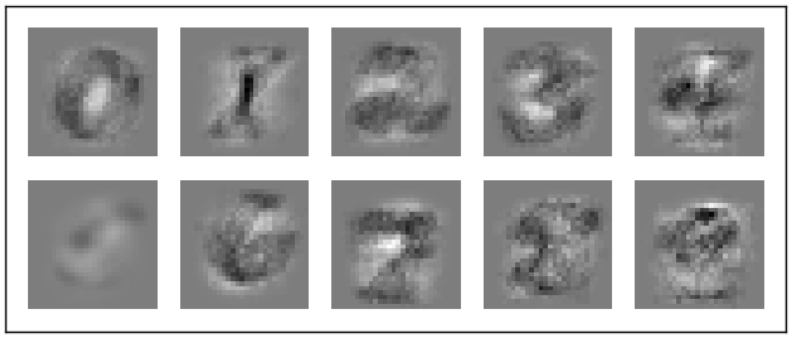

Figure 5: Learned w for j=5

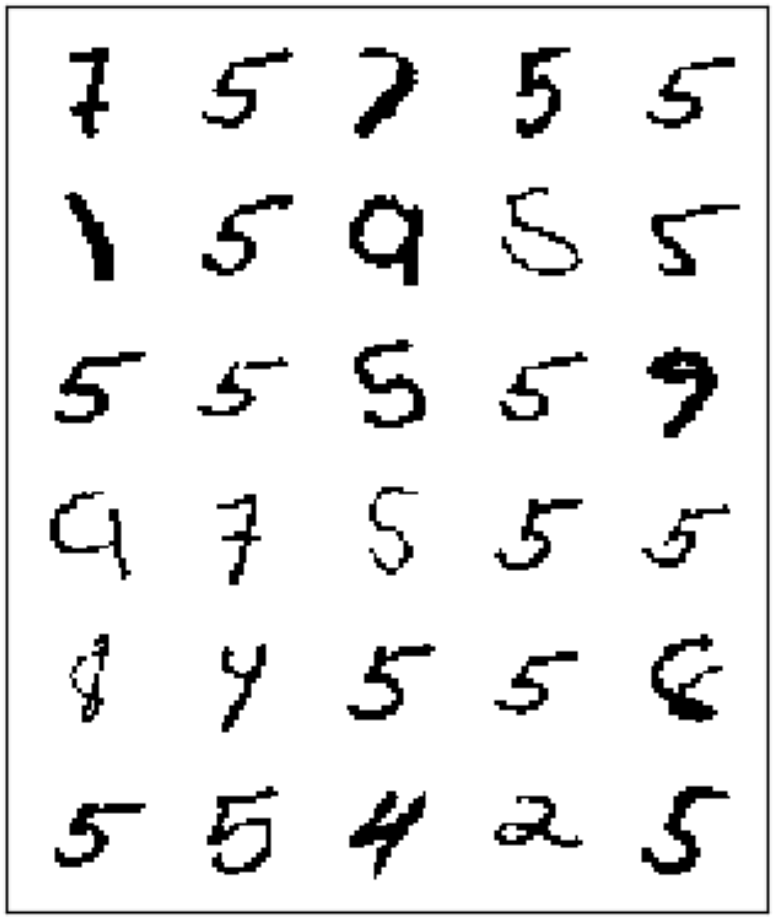

Figure 6: 30 Mispredicted Samples

5 Limitations

One limitation of the above experiments may stem from the labeler model we used to generate noisy data. To produce an adequate amount of noisy data, we chose a logistic regression model with power levels. However, when power level j is low, a lot of noise examples are produced, among which most are fives and eights. In this case, the weak points of the labeler might be propagated to our classifier. Namely, the fact that the labeler cannot recognize fives made it hard for our model to learn representation of fives. A wider problem is that the labels given by such labeler fail to be balanced, and the performance of our data cleaning model is adversely affected by it, as shown in Wei and Dunbrack Jr [2013]. To deal with this issue, we could select the first 1000 examples of each class that the labeler produced and use them as training data. Also, using other methods to get noisy data may help as well.

Another limitation is that effect of θ is minuscule due to the nature of the model. As pointed out in the results section, in our experiments, $p(\mathbf{c} \mid \mathbf{x}, \mathbf{w})$ and $p(\mathbf{r} \mid \mathbf{x}, \mathbf{w})$ is hardly different. Since $\theta = p(c|r)$ is a linear map from true data to noisy data, the values on the diagonal are relatively large. Despite the noise, θ may only have little to do with the maximal predictive likelihood of a given example. To test our guess, we used different θ's and tested the performance on $j = 1$. When we set all entries of θ to be 0.1, that is, there is no preference towards the true class label, the accuracy rate dropped dramatically to 36.40% on training data and 34.85% on testing data. But this is an extreme scenario. When we use back propagation to train θ, after a few rounds, the entries on the diagonal will tend to have some values very close to 1, and all other entries near 0.

6 Conclusions

In this project, we explored the problem of fitting a learning model under noisy labels. We stated the main problem of training with noise and how state-of-the-art studies attempt to deal with them. We formulated our approach to this problem with a probability model and conducted thorough experiments that produced surprising results: the model was underfitting the data and did not outperform the benchmark. We investigated the underlying problems and made suggestions that could potentially improve the outcome. It must be reminded that the problem of noisy data is inevitable and will remain a roadblock to better machine learning results. Continual work must be made in order to better tackle this problem in the future.

References

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210, 2004.

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.

Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78(3):287–304, 2010. ISSN 1573-0565. doi: 10.1007/s10994-009-5165-z. URL `http://dx.doi.org/10.1007/s10994-009-5165-z`.

Yann LeCun and Corinna Cortes. The mnist database of handwritten digits, 1998.

Benoît Frénay and Ata Kabán. A comprehensive introduction to label noise. In *ESANN*, 2014.

Choh-Man Teng. A comparison of noise handling techniques. In *FLAIRS Conference*, pages 269–273, 2001.

Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon. Declaratively cleaning your data using ajax. In *In Journees Bases de Donnees*. Citeseer, 2000a.

Carlos Daniel Paulino, Paulo Soares, and John Neuhaus. Binomial regression with misclassification. *Biometrics*, 59(3):670–675, 2003.

Martin Ladouceur, Elham Rahme, Christian A Pineau, and Lawrence Joseph. Robustness of prevalence estimates derived from misclassified data from administrative databases. *Biometrics*, 63(1): 272–279, 2007.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. pages 1096–1103, 2008.

Carla E Brodley, Mark A Friedl, et al. Identifying and eliminating mislabeled training instances. In *AAAI/IAAI, Vol. 1*, pages 799–805, 1996.

Sofie Verbaeten and Anneleen Van Assche. Ensemble methods for noise elimination in classification problems. In *International Workshop on Multiple Classifier Systems*, pages 317–325. Springer, 2003.

Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *Journal of Machine Learning Research*, 17(34):1–7, 2016.

Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon. Ajax: an extensible data cleaning tool. In *ACM Sigmod Record*, volume 29, page 590. ACM, 2000b.

Paul Gustafson, Nhu D Le, and Refik Saskin. Case–control analysis with partial knowledge of exposure misclassification probabilities. *Biometrics*, 57(2):598–609, 2001.

Charles Bouveyron and Stéphane Girard. Robust supervised classification with mixture models: Learning from data with uncertain labels. *Pattern Recognition*, 42(11):2649–2658, 2009.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, 2015.

Qiong Wei and Roland L Dunbrack Jr. The role of balanced training and testing data sets for binary classifiers in bioinformatics. *PloS one*, 8(7):e67863, 2013.