Automatic Training Data Filtering for Errors Removing and Improving the Quality of the Final Neural Network

N. Z. Valishina[III,IV], S. A. Ilyuhin[II,IV], A. V. Sheshkus[I,IV], V. L. Arlazarov[I,II,IV]

[I]Federal Research Center "Computer Science and Control" of RAS, Moscow, Russia
[II]Moscow Institute of Physics and Technology (State University), Moscow, Russia
[III]Lomonosov Moscow State University, Moscow, Russia
[IV]Smart Engines Service LLC, Moscow, Russia

Abstract. Real-world data are often dirty. In most cases it negatively affects the accuracy of the model trained on such data. Supervised data correction is an expensive and time-consuming procedure. So one of the possible ways to solve this problem is to automate the cleaning process. In this paper, we consider such a preprocessing technique for improving the quality of the trained network as automatic cleaning of training data. The proposed iterative method is based on the assumption that the polluted data are most likely located farther away from the median of the class. It includes detection and subsequent removal of the noisy data from a training set. Experiments on a generated synthetic dataset demonstrated that this method gives good results and allows to clean up the data even at high levels of pollution and significantly improve the quality of the classifier.

Keywords: data cleaning, outlier(s) detection, mislabels, classifier, siamese neural network.

DOI 10.14357/20718632220304

Introduction

Data collection is one of the key points among the many machine learning problems. Data preparation takes a lot of time and includes collecting, cleaning, analyzing, and creating ground truth. As the scope of machine learning is expanding, there is a special need for new training data. In particular, the increased popularity of neural networks requires the availability of representative data sets for their training.

If the data collecting task is solved, then the task of correctly labeling these data is set. And there are several possible cases: either all the data are labeled manually or only a relatively small part of the data is labeled. In the first case, it's a very laborious and time-consuming process that is implemented using crowdsourcing or an own team of people who label the data. In the second, most of the data are labeled automatically, using assumptions related to the link between the distributions of unlabeled and labeled data [1].

In both cases, errors may occur during data labeling, such as mislabels or missing values, or semantic errors. Herewith, in the second case, the percentage of mislabels should intuitively be higher. The presence of errors in the labels can lead to a deterioration in the quality of the trained model [2], in particular, in classification problems.

In this paper, we propose a method designed to improve the data quality by detecting and subsequent deleting of mislabeled samples from data.

Related work

Data cleaning is a preprocessing technique and can be separated in two tasks: error detection and data repairing. At first, data inconsistencies are

identified, the second part involves updating the available data to correct any detected errors. We will next consider only the problem of detecting errors, and as a technique for correcting them, we will use the simplest one, which consists in their removing.

Depending on where inconsistencies occur in the data, errors can be divided into two types [3]: attribute and class errors. And there are many methods for dealing with both attribute errors [4] and class errors [5]. We cover several works related to the detection of class errors, specifically outliers and mislabeled data.

Mislabeled data

The need to clean the data leads us to the concept of a filter or filtering algorithm. Polluted data are fed to the input of such an algorithm, and a positive or negative tag is assigned to each sample at the output, indicating whether the instance is mislabeled or not. After that, objects with positive tags are deleted or are subject to reclassification.

The idea to use not one filtering algorithm, but several (the so-called *ensemble of classifiers*) was implemented by Brodley and Friedl [6]. In their work, authors compare the quality of classification when using three types of filtering: a single algorithm, majority voting and consensus voting. A majority vote filter tags an instance as mislabeled if more than half of base-level classifiers classify it incorrectly. A consensus vote filter requires all votes to make such a decision. The following algorithms are considered as base-level classifiers: decision tree, kNN and linear machine. Samples marked as mislabeled are deleted, and the resulting corrected dataset is used to train the final classifier. This method allows to achieve base-line accuracy of classifier with data contamination not exceeding 20%.

The use of an ensemble of filters is also found in the work [7] in relation to Bioinformatic datasets. The authors also pay attention to the methods of data repairing, namely, they compare three data correction techniques: noise removal, noise reclassification, as well as a combination of these two techniques. The results of their experiments show that a higher classification accuracy is achieved when detected errors are removed from the dataset.

In [8] the authors use an ensemble of filters in a binary classification problem. The ensemble consists of only two base level classifiers - True NN and False NN. The first of them is trained to predict the degree of the true memberships while the second is trained to predict the degree of false memberships. These networks have the same architecture, and their difference is that Falsity NN use the complement of target outputs of the Truth NN to train. If both networks recognize the sample as incorrectly classified, it is removed from the dataset. Experiments conducted on real data show that this technique removes only the highly possible misclassification patterns, in contrast to majority voting techniques, where ANN, DT and kNN are taken as base-level filters. The proposed method is superior to the majority voting and consensus voting filtering from the point of view of the accuracy of the classifier trained on filtered data. However, in general, the classifier accuracy was improved by no more than 2%.

Outliers

Speaking of outliers, it is “a data point that is far outside the norm for a variable or population“[9]. And there are a lot of methods for detecting them in univariate [10; 11] and multivariate [12-15] cases.

To find outliers among the data, you first need to introduce a metric in the space of these data. It is customary to consider the Mahalanobis metric a measure of the distance between two random vectors (for example see [16]). So the distance from the random vector to the center, called Mahalanobis distance (MD), is calculated as follows:

$$MD=\sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)}, \quad (1)$$

where x is a vector of variables $x=(x_1,x_2,\ldots,x_k)$, $\mu=(\mu_1,\mu_2,\ldots,\mu_k)$ is a mean vector, Σ is a $k\times k$ covariance matrix. μ and Σ are generally unknown in advance, and the sample mean and the sample covariance matrix are used as their estimates. However, MD cannot be used in the presence of outliers, since the sample mean and sample deviation are calculated for the entire data.

In the univariate case the median and MAD (median absolute deviation) are more robust statistics ([10]):

$$\text{MAD} = median(|\text{X} - median(\text{X})|), \quad (2)$$

where X is a univariate sample. An object x from sample X is considered to be an outlier if the distance from it to the median exceeds three MAD values:

$$\frac{\text{x-}median(\text{X})}{\text{MAD}} > 3. \quad (3)$$

The mean and the covariance matrix found by the MCD (minimum covariance determinant) [17; 18] method can serve as examples of robust statistics in the multivariate case. This method includes the following steps: for each subsample of length h, the sample mean and the sample covariance matrix, as well as its determinant, are calculated. Then, from all subsamples of length h, a subsample with the covariance matrix having the smallest determinant is selected. μ_{MCD} is assumed to be equal to the sample mean of this "good subsample", Σ_{MCD} is assumed to be equal to the corresponding sample covariance matrix. The number h is chosen in the range from $n/2$ to n, where n is the sample size. Those objects whose Mahalanobis-MCD [12] distance is greater than a certain constant are taken as outliers:

$$\text{MD}=\sqrt{(\text{x-}\mu_{MCD})^{T}\Sigma_{MCD}^{-1}(\text{x-}\mu_{MCD})}, \quad (4)$$

Since this method involves calculations for each subsample of length h, where $n/2 \le h \le n$, it turns out to be computationally demanding. It should be noted that there is a less computationally demanding version of this algorithm, called FAST-MCD [19]. However, it also involves multiple calculations over the entire sample and subsamples of length h, which is quite inefficient for large datasets.

As for NNs, it should also be noted that there is a radically different approach to solving such a problem as training with noisy labeled data, the purpose of which is to build robust models. There are two main methods: changing the network structure [20; 21] and changing the loss function [22; 23].

In this paper, we propose a method designed to improve the data quality. The main assumption used is as follows: if the image has an incorrect label, the feature vector is more likely to be farther away from the median of the class than the vector corresponding to the correctly labeled image. Therefore, the problem of detecting mislabels is reduced to the outliers detection. Using such statistics as the median we detect possible outliers and then remove them.

Suggested method

The method consists in sequentially repeating the following steps:

1 the new network is trained on the available training data (initially dirty);

2 in each class, 5% of the feature vectors that are farthest (l2) from the median of the class are determined. They are moved to the storage.

In the multivariate case, the median of the class is calculated as a vector whose components are the medians of the corresponding univariate distributions. The quality of the trained network is measured at each iteration.

The question of how many steps should be applied is solved taking into account the fact that with each iteration of the method, the amount of data decreases. For example, if it is known that the data is not much polluted, then it is better to take a smaller number of iterations. Conversely, if the data is presumably heavily polluted, then the number of steps in the method should be set larger.

Division into classes

We use the Siamese network [24-26] as a classifier in the proposed method, since its architecture allows to map the point from the data space to the embedding space so that the distance between similar data in the embedding space (l_2) is small compared to the distance between different data. That is, in embedding space the distance between samples of the same class will be relatively small in comparison with the distance between samples from different classes.

The question of which class the image belongs to is solved as follows. A small subset (about 100-200) of images belonging to the same class is taken. An already trained network calculates a feature vector for each input image. Next, the class descriptor, which is the feature vector averaged over this subset, is calculated. This procedure applies to each class. When it is completed, a descriptor is assigned to each class. A certain class is assigned to each input image using the nearest neighbor

method. This class is defined by the descriptor closest to the feature vector of the input image.

It should be noted here that the images we used to calculate the descriptors were free from contamination. However the class descriptor can be calculated not only as the average of the samples taken, but also as their median. In this case, the descriptors can be calculated even from samples taken from a polluted data, since the median is a robust statistics.

Experiments and results

Training data

To evaluate the effectiveness of the proposed method, synthetic data were generated using the method proposed in [27]. It was a set of images of digits from 0 to 9 printed in different fonts and located on different backgrounds (Fig. 1). There were 1250000 images in total, which were divided into a training (80%) and test (20%) set. The backgrounds for the test set were different from the backgrounds for the training set. For experiments, the training data were polluted by changing the class labels. In more detail, the same number of samples were taken from each class, and then each of these samples was randomly assigned a class different from the original one. The test data remained unchanged throughout the experiments. All images were transformed to grayscale and scaled to size 25 × 41 to feed to the input of the neural network.

Fig. 1: Examples of generated images

Architecture of Convolutional Neural Network

As mentioned above, we use deep metric learning for data filtering, in particular the Siamese network, because it allows us to measure the distance between the vectors of objects in the resulting space, and the smaller this distance, the closer the objects are located in the input space. Moreover, the size of such a network does not directly depend on the number of classes. Another advantage of such a network is the possibility of additional training when expanding the alphabet, while the classifying network needs to be retrained “from scratch“ to add new classes.

Each of the branches of the Siamese network has the architecture described in Table 1. We use convolutional layers, because such architecture takes into account the topology of input data. This property makes them attractive for problems where images are used as input data [28; 29] since there is a dependence between spatially adjacent pixels. The number of outputs is chosen to be three due to the fact that the alphabet is not large and consist of ten elements. The proposed architecture has 3.22×10^4 weights, the function (5) was chosen as an activation function for convolutional layers. The contrastive loss [30] was used during the experiments, because this function tries to maximize the distance between two samples from different classes and minimize the distance between samples from the same class.

Table 1. Architecture of network

Layers			
#	Type	Parameters	Activation
1	conv	6 filters 5×5, stride 1×1, no padding	softsign
2	conv	6 filters 2×2, stride 2×2, no padding	softsign
3	conv	16 filters 5×5, stride 1×1, no padding	softsign
4	conv	16 filters 2×2, stride 2×2, no padding	softsign
5	fc	84 outputs	softsign
6	fc	3 outputs	

The architecture of the resulting network turned out to be quite simple. In fact, we did not pursue the goal of building a network that will show the best quality, since this is not so necessary for conducting an experiment.

$$\text{softsign}(x) = \frac{x}{1+|x|} \quad (5)$$

Experiments

Before each experiment, the training data were polluted by changing the labels. Moreover, the same number of images were polluted in each class. Each experiment consisted in applying the method described above, the number of iterations was chosen to be six. The percentage of detected true (*TrueM*) and false (*FalseM*) mislabeled samples were measured at each iteration.

$$TrueM = \frac{\text{num of detected true mislabeled samples}}{\text{initial size of the training data}} \cdot 100\%, \quad (6)$$

$$FalseM = \frac{\text{num of detected false mislabeled samples}}{\text{initial size of the training data}} \cdot 100\%, \quad (7)$$

where initial size of training data was 10^6.

The accuracy of the classifier was calculated as the percentage of its correct answers:

$$\text{Acc} = \frac{N_{\text{correct}}}{N_{\text{all}}} \cdot 100\%, \quad (8)$$

where N_{correct} is the number of correctly classified images, N_{all} is the size of the entire test set (250000 images).

Results

The results of the experiments are presented in Table 2. In six iterations, it was possible to completely get rid of mislabeled data at pollution levels not exceeding 25%. At higher pollution levels, six iterations are not enough. This is not surprising, because for six iterations, a part of deleted data equals $(1-0.95^6) \approx 0.265$. In such cases, more iterations of the algorithm should be applied.

As can be seen from the Table 2, with an increase in the level of contamination, the *FalseM* indicator remains negligible, i.e., the number of deleted "good" samples is very small even with a high level of data contamination. Table 3 shows which samples the algorithm filters out at each step when cleaning data that is 30% contaminated. Correctly labeled samples detected by the algorithm as mislabels are mostly low-quality samples, which can be seen from the right column of the Table 3.

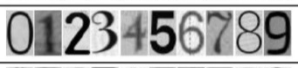
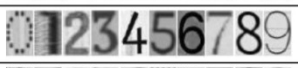
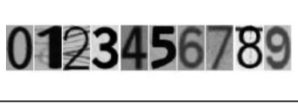
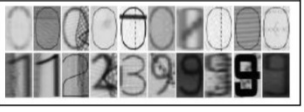
Table 2. Results

		Cleaning iteration number						
Pollution level	Measured indicators	0	1	2	3	4	5	6
0%	*Acc*,%	**99.08**	97.00	96.53	97.14	95.71	95.39	97.53
5%	*TrueM*,%		4.95	5.00				
	FalseM,%		0.05	4.75	9.26	13.55	17.62	21.49
	Acc,%	**98.90**	98.14	97.82	98.08	97.62	97.42	97.08
10%	*TrueM*,%		5.00	9.74	10.00			
	FalseM,%		$1 \cdot 10^{-3}$	0.01	4.26	8.55	12.62	16.49
	Acc,%	91.90	**98.45**	98.36	97.56	97.52	97.60	96.57
20%	*TrueM*,%		5.00	9.75	14.26	18.53	20.00	
	FalseM,%		$2.7 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$	$4.9 \cdot 10^{-3}$	0.02	2.62	6.49
	Acc,%	87.90	91.73	95.97	98.08	98.11	**98.17**	97.20

Table 2. Results (end of the table)

		Cleaning iteration number						
Pollution level	Measured indicators	0	1	2	3	4	5	6
25%	*TrueM*,%		5.00	9.75	14.26	18.53	22.60	25.00
	FalseM,%		$2 \cdot 10^{-4}$	$2.6 \cdot 10^{-3}$	$6.2 \cdot 10^{-3}$	0.01	0.02	1.49
	Acc,%	90.91	84.84	95.64	97.01	97.64	98.20	**98.22**
30%	*TrueM*,%		5.00	9.75	14.26	18.55	22.61	26.48
	FalseM,%		$1 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$8 \cdot 10^{-4}$	$1.7 \cdot 10^{-3}$	$5.6 \cdot 10^{-3}$	$8.8 \cdot 10^{-3}$
	Acc,%	89.16	85.86	95.63	98.02	98.81	**98.90**	98.70
40%	*TrueM*,%		5.00	9.75	14.26	18.55	22.62	26.48
	FalseM,%		0.00	$4 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$8 \cdot 10^{-4}$	$3.7 \cdot 10^{-3}$	0.01
	Acc,%	88.16	86.07	92.31	96.32	97.27	97.93	**98.08**
50%	*TrueM*,%		5.00	9.75	14.26	18.55	22.62	26.49
	FalseM,%		0	$1 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$1.3 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	$2.8 \cdot 10^{-3}$
	Acc,%	82.36	84.91	83.46	86.99	91.48	95.60	**96.70**
80%	*TrueM*,%		4.95	9.66	14.15	18.43	22.50	26.37
	FalseM,%		0.05	0.10	0.11	0.11	0.12	0.12
	Acc,%	47.35	58.69	64.41	67.24	65.23	64.73	**79.90**

Table 3. Examples of true and false mislabels detected on each iteration of the algorithm

Iter. num	Detected true mislabels	Detected false mislabels
1		
2		
3		
4		
5		
6		

The proposed method also improved the quality of the network at pollution degree in the range from 10% to 50%. In almost all of these experiments (with the exception of 40% of contamination), it was possible to achieve the quality of a network trained on clean data.

Conclusion

In this work, we considered a training data cleaning method that identifies mislabeled instances and remove them. Its advantage lies in the fact that it does not require human control, that is, it is fully automated. The results of experiments conducted on the generated synthetic data demonstrate that the proposed method allows to get rid of incorrectly labeled data even at high levels of pollution. The method also allowed accuracies near to the baseline accuracy of the classifier to be retained for noise levels up to 30%. As the noise level increases, the ability of the method to retain the baseline accuracy of the classifier decreases.

As a part of further research, it is planned to evaluate the effectiveness of this method on real data. Moreover, we are going to improve the method by correcting the detected incorrectly labeled instances. For example, we can determine which class is the closest to the detected instance and mark the instance with the label of this class. As the distance from the object to the class, we can consider the distance in the sense of l_2 from its feature vector to the median of the class.

References

1 J. E. Van Engelen and H. H. Hoos, “A survey on semi-supervised learning,” Machine Learning 109(2), 373–440 (2020).

2 X. Zhu and X. Wu, “Class noise vs. attribute noise: A quantitative study,” Artificial intelligence review 22(3), 177– 210 (2004).

3 X. Wu, Knowledge acquisition from databases, Intellect books (1995).

4 X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, “Data cleaning: Overview and emerging challenges,” in Proceedings of the 2016 international conference on management of data, 2201–2206 (2016).

5 B. Frenay and M. Verleysen, “Classification in the presence of label noise: a survey,” IEEE transactions on neural´ networks and learning systems 25(5), 845–869 (2013).

6 C. E. Brodley and M. A. Friedl, “Identifying mislabeled training data,” Journal of artificial intelligence research 11, 131–167 (1999).

7 A. L. Miranda, L. P. F. Garcia, A. C. Carvalho, and A. C. Lorena, “Use of classification algorithms in noise detection and elimination,” in International Conference on Hybrid Artificial Intelligence Systems, 417–424, Springer (2009).

8 P. Jeatrakul, K. W. Wong, and C. C. Fung, “Data cleaning for classification using misclassification analysis,” Journal of Advanced Computational Intelligence and Intelligent Informatics 14(3), 297–302 (2010).

9 J. W. Osborne and A. Overbay, “The power of outliers (and why researchers should always check for them),” Practical Assessment, Research, and Evaluation 9(1), 6 (2004).

10 C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, “Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median,” Journal of Experimental Social Psychology 49(4), 764–766 (2013).

11 S. Seo, A review and comparison of methods for detecting outliers in univariate data sets, PhD thesis, University of Pittsburgh (2006).

12 P. J. Rousseeuw and B. C. Van Zomeren, “Unmasking multivariate outliers and leverage points,” Journal of the American Statistical association 85(411), 633–639 (1990).

13 C. Leys, O. Klein, Y. Dominicy, and C. Ley, “Detecting multivariate outliers: Use a robust variant of the mahalanobis distance,” Journal of Experimental Social Psychology 74, 150–156 (2018).

14 P. Filzmoser, A. Ruiz-Gazen, and C. Thomas-Agnan, “Identification of local multivariate outliers,” Statistical Papers 55(1), 29–47 (2014).

15 M. Riani, A. C. Atkinson, and A. Cerioli, “Finding an unknown number of multivariate outliers,” Journal of the Royal Statistical Society: series B (statistical methodology) 71(2), 447–466 (2009).

16 V. V. Mazhuga and M. V. Khachumov, “Algorithms of images processing for biological systems status classification,” Information Technologies and Computational Systems (2), 54–63 (2012).

17 P. J. Rousseeuw, “Least median of squares regression,” Journal of the American statistical association 79(388), 871–880 (1984).

18 P. J. Rousseeuw, “Multivariate estimation with high breakdown point,” Mathematical statistics and applications 8(37), 283–297 (1985).

19 P. J. Rousseeuw and K. V. Driessen, “A fast algorithm for the minimum covariance determinant estimator,” Technometrics 41(3), 212–223 (1999).

20 J. Goldberger and E. Ben-Reuven, “Training deep neural-networks using a noise adaptation layer,” (2016).

21 A. J. Bekker and J. Goldberger, “Training deep neural-networks based on unreliable labels,” in 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2682–2686, IEEE (2016).

22 G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, and L. Qu, “Making deep neural networks robust to label noise: A loss correction approach,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 1944–1952 (2017).

23 A. Ghosh, H. Kumar, and P. Sastry, “Robust loss functions under label noise for deep neural networks,” in Proceedings of the AAAI Conference on Artificial Intelligence, 31(1) (2017).

24 W. Zheng, Z. Chen, J. Lu, and J. Zhou, “Hardness-aware deep metric learning,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 72–81 (2019).
25 F. Fedorenko and S. Usilin, “Real-time object-to-features vectorisation via siamese neural networks,” in Ninth International Conference on Machine Vision (ICMV 2016), 10341, 103411R, International Society for Optics and Photonics (2017).
26 S. A. Ilyuhin, A. V. Sheshkus, and V. L. Arlazarov, “Recognition of images of korean characters using embedded networks,” in Twelfth International Conference on Machine Vision (ICMV 2019), 11433, 1143311, International Society for Optics and Photonics (2020).
27 Y. S. Chernyshova, A. V. Gayer, and A. V. Sheshkus, “Generation method of synthetic training data for mobile ocr system,” in Tenth international conference on machine vision (ICMV 2017), 10696, 106962G, International Society for Optics and Photonics (2018).
28 O. P. Soldatova and A. A. Garshin, “Convolutional neural network applied to handwritten digits recognition,” Computer optics 34(2) (2010).
29 E. Holm, A. A. Transeth, O. Ø. Knudsen, and A. Stahl, “Classification of corrosion and coating damages on bridge constructions from images using convolutional neural networks,” in Twelfth International Conference on Machine Vision (ICMV 2019), 11433, 1143320, International Society for Optics and Photonics (2020).
30 R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06), 2, 1735–1742, IEEE (2006).

Valishina N. Z. Smart Engines Service LLC, Prosp. 60-letiya Oktyabrya, 9, Moscow, 117312, Russia, e-mail: n.valishina@smartengines.com

Ilyuhin S. A. Smart Engines Service LLC, Prosp. 60-letiya Oktyabrya, 9, Moscow, 117312, Russia, e-mail: ilyuhinsa@smartengines.com

Sheshkus A. V. Smart Engines Service LLC, Prosp. 60-letiya Oktyabrya, 9, Moscow, 117312, Russia, e-mail: asheshkus@smartengines.com

Arlazarov V. L. Doctor of Sciences, corresponding member of the Russian Academy of Sciences. He is a head of sector 9 at the Institute for Systems Analysis FRC CSC RAS, e-mail: vladimir.arlazarov@smartengines.com