LABEL-EFFICIENT AND ROBUST LEARNING FROM MULTIPLE EXPERTS

Bojan Kolosnjaji

Technical University of Munich

Apostolis Zarras

University of Piraeus
Foundation for Research and Technology, Hellas

ABSTRACT

Learning from multiple sources of labeled data presents unique challenges, including the cost of using many annotators in the training, test, and production time and their limited reliability. In this paper, we analyze the problem of label-efficient learning and propose a method for training a classification system from data labeled by multiple annotators, where only a small subset of them is chosen adaptively to reduce later communication effort and the effect of malicious annotators, especially in the validation and test phase. We define an iterative optimization procedure where the annotation weight vector's sparsity is enforced while reducing the overall classification error. Using real-world data, we show that our approach can pertain to the classification performance on the complete set of annotators while reducing the need for annotator labels by over 50% on several different tasks. Furthermore, we demonstrate that label sparsity reduces label flip attacks' effect on performance.

1. INTRODUCTION

Supervised machine learning depends on large sets of labeled data. Frequently, we benefit from labeled data collected by multiple separate sources with varying reliability and joining this data into one machine learning model. However, we are often limited in the number of data sources we can use because of budget constraints, limited processing power, or insufficient communication bandwidth. Furthermore, we often lack ground truth training labels to help us fit a reliable model. This causes the need for a framework that enables a label-efficient combination of labeled data from multiple sources.

A great fitting scenario for this framework is crowdsourcing. We can gather labeled data from multiple annotators using systems such as Amazon Mechanical Turk. After collecting this training data, we need to join the labeling decisions and train one machine learning system while leveraging all annotations. Relying on a concept of *wisdom of crowds* [1], we could assume that if we gather data from a sufficient number of people, most of them will provide correct labeling decisions. However, gathering data from many annotators can be expensive, as workers on Amazon Mechanical Turk require payment for their work. Workers also have different reliability and expertise or might be malicious and intentionally provide wrong labels. Thus, we need a procedure to determine the annotators' expertise and detect malicious annotators. There is an extensive body of work related to learning from multiple labelers [2–11]. The most similar to our work is the paper of Li and Liu [12], who propose a combinatorial procedure to find an optimal subset of annotators. However, such combinatorial procedures are *very resource-intensive*, especially for a large initial set of annotators.

In this paper, we design a procedure for selecting a small number of highly competent annotators under a predetermined ground truth label budget whose combined expertise enables acceptable performance in machine learning tasks. This way, we save on annotation and ground truth label costs when retrieving label information while minimizing the impact on performance due to a limited communication budget. This also makes our system robust to unreliable or adversarial annotators. As a matter of fact, the experimental evaluation reveals that our approach boosts classification performance while reducing the communication effort by more than 50%.

2. METHODOLOGY

Our methodology is based on the scenario where multiple annotators provide labels for the same training set to a server. During training, the server forwards the training inputs to all the annotators to retrieve labels. By using the data labeled from a large set of annotators, the server trains a machine learning model that jointly learns all the annotators' expertise and selects annotators while minimizing the loss w.r.t. the ground truth. We assume that the server has a limited number of ground truth points available during training and uses them to adapt the annotators' weights. We also assume a possibility of an unlimited communication budget during training. The model would fit into the following framework of classification of data X with N available points in the training set:

$$\hat{Y} = f(X, Y, Z, W, V) \tag{1}$$

The model also depends on the set of ground truth labels (Y), labels provided by a set of M annotators ($Z \in \mathbb{R}_N^M$), a set of parameters for the annotator-specific models ($W \in \mathbb{R}_M^P$) and a set of weights prioritizing the annotators and annotations ($V \in \mathbb{R}_M^N$). Here the point $x_n \in X$ is classified based

on its L input features (x_{nl}), available ground truth ($y \in \mathbb{R}^K$), annotations for this point given by multiple (M) annotators ($z \in \mathbb{R}^M$) and parameters of the model. There are two sets of parameters, representing namely the weights estimating the expertise of different annotators($v \in \mathbb{R}^M$) and the annotator-specific models with P parameters ($W \in \mathbb{R}^P_M$).

2.1. Simple Model

In a more straightforward instance of our model, we consider annotator expertise in the form of weights. We introduce a vector of weights that determines the importance assigned to particular annotators. Furthermore, we design an optimization approach that includes the sparsification of this weight vector, i.e. we optimize a model that includes annotations from a reduced number of annotators. Classification is done by evaluating the weighted sum of annotator-specific models:

$$\hat{y}_n = \sum_{m=1}^{M} v_m \sigma(x_n, w_m) \tag{2}$$

where the parameters v_m and w_m are trained based on the inputs (X), ground truth labels (Y), and labels from annotators (Z). Function σ represents the classification model and depends on the chosen machine learning classification method. The result of σ is the binary classification estimate $P(y_n = 1)$ from different annotator-specific models.

The loss function with cross-entropy loss on ground truth, for N training examples and M annotators, used in all our experiments, looks like the following:

$$L(w, v) = -\frac{1}{K} \sum_{k=1}^{N} (y_k \log \sum_{m=1}^{M} v_m \hat{y}_{km} + (1 - y_k) \log(1 - \sum_{m=1}^{M} v_m \hat{y}_{km})) - \psi \frac{1}{N} \sum_{n=1}^{N} (\sum_{m=1}^{M} v_m z_{nm} \log \sum_{m=1}^{M} v_m \hat{y}_{nm} + (1 - \sum_{m=1}^{M} v_m z_{nm}) \log(1 - \sum_{m=1}^{M} v_m \hat{y}_{nm})) + \lambda \mid v \mid \tag{3}$$

subject to:

$$\sum_{m=1}^{M} v_m = 1 \tag{4}$$

where y_k are available ground truth labels for a subset of training examples (we assume that not necessarily all examples have ground truth labels, i.e. $K < N$), $\hat{y}_{nm} \in \mathbb{R}^{N \times M}$ and $\hat{y}_{km} \in \mathbb{R}^{K \times M}$ are labels retrieved by querying current annotator models, $z_{nm} \in \mathbb{R}^{N \times M}$ are training labels from annotators, and v_m is a normalized vector of weights for all the annotators. The final loss function consists of a cross-entropy calculation for the empirical loss w.r.t. the ground truth, the cross-entropy loss w.r.t. the annotations weighted by a hyperparameter ψ, and the L_1-norm of the parameter vector v weighted by another hyperparameter λ. The hyperparameters ψ and λ are predefined before the training and are kept constant during the model optimization. We use a gradient-based optimization and alternate between updating parameters of the annotator-specific models (w) and annotator weights (v).

2.2. Full Expertise Model

The simple expertise model enables us to select the most reliable annotators. However, assuming the annotators do not have the same expertise for various parts of the feature space, we could benefit from a more fine-grained annotator selection. Thus, we also design a full expertise model to select the annotators for each classification query. The classification is done according to the following expression:

$$\hat{y}_n = \sum_{m=1}^{M} v_{nm} \sigma(x_n, w_m) \tag{5}$$

In this case, we have a different loss function:

$$L(w, v) = -\frac{1}{K} \sum_{n=1}^{K} (y_k \log \sum_{i=1}^{M} v_{km} \hat{y}_{km} + (1 - y_k) \log(1 - \sum_{m=1}^{M} v_{km} \hat{y}_{km})) - \psi \frac{1}{N} \sum_{n=1}^{N} (\sum_{m=1}^{M} v_{nm} z_{nm}) \log \sum_{i=1}^{M} v_{nm} \hat{y}_{nm} + (1 - \sum_{m=1}^{M} v_{nm} z_{nm}) \log \sum_{i=1}^{M} v_{nm} \hat{y}_{nm}) + \sum_{n=1}^{N} \lambda \mid v_n \mid \tag{6}$$

We can also optimize this loss function with a similar gradient descent procedure. Here we have a matrix instead of a vector v, where each row in the matrix needs to sum to one, and we need to make the rows sparse. During training, we maintain a multinomial logistic regression model, based on the annotator weights for different data points:

$$v = S(px) \tag{7}$$

where $p \in \mathbb{R}^{M \times L}$ is the model's parameter, $x \in \mathbb{R}^L$ the input example of dimensionality D, S is the softmax function and $v \in \mathbb{R}^M$ the resulting weight for annotators per data example. This model is trained with the annotator weights resulting from the gradient-based model described by the loss function in Equation 6. The resulting model is then used to determine the annotator weights in the validation and test phase.

2.3. Robustness

There is a chance that some of the annotators may be malicious. Adversarial annotators can influence the training labels to increase the test loss, reducing the system's performance. However, an adversarial annotator also aims to remain undetected while exerting influence. As the annotator can only affect annotations that they can feed to the server, we consider adversarial label flips a possible type of attack. According to Biggio et al. [13], security analysis of a machine learning algorithm should be done by investigating the change in the algorithm performance under an increasing amount of perturbation. We conduct such tests on our models by executing a label flip attack based on the label poisoning procedure by Paudice et al. [14]. Algorithm 1 describes one iteration of the attack procedure for a chosen annotator. Precisely, it shows how we find an optimal label flip for one attack iteration.

We flip the labels of one annotator's annotations one by one using Algorithm 1. After each label flips, we follow the

Input: Training Set: (X_s, Y_s, Z_s)
Test set: (X_t, Y_t, Z_t)
Model parameters: (W, V)
Training algorithm: (train)
Test loss: $(\mathrm{loss_{tt}})$
Flipping function: $\mathrm{flip(X, i)}$ - flip i-th label in label list X
Result: Z_s
$S = \mathrm{set}(Z_s)$
while $S \neq \emptyset$ **do**
  $losses = []$
  for $z_i \in S$ **do**
    $\overline{Z}_s = \mathrm{flip}(Z_s, i)$
    $(W, V) = \mathrm{train}(X_s, Y_s, \overline{Z}_s)$
    $l = \mathrm{loss_{tt}}(W, V, X_t, Y_t, Z_t)$
    $losses = losses \cup l$
  end
  $k = \mathrm{argmax}(losses)$
  $Z_i = \mathrm{flip}(Z_i, k)$
  $S = S \setminus z_k$
end

Algorithm 1: Label flip for one annotator in the robustness test to label poisoning attack

change in the test loss. We use a greedy algorithm to choose a label to flip in a particular attack step. In each step, we try to flip every remaining label separately, train the model and compute the test loss. Finally, we select the label flip that results in the highest increase in test loss due to the flipping action. This is repeated until the desired effect on the model is achieved, or all labels are flipped. We expect the test loss to change with each chosen label flip in a successful attack.

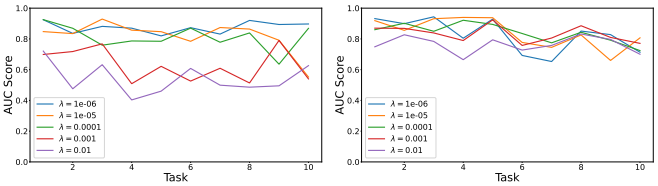
We use two types of attacks: white-box and grey-box. In white-box, the attacker has complete knowledge of the system and can easily find a label flip that maximizes the attack. This is the most potent attack, but not always realistic. In grey-box, the attacker is an annotator and only knows the parameters of their part of the model, making it more challenging to affect the model's performance. The choice of labels to flip is only optimized based on the loss known to the attacker during optimization. Overall, we gauge the system's resilience by the number of label flips needed to impact the model in the form of increased test loss or decreased performance on the test set.

3. EVALUATION

This section describes the evaluation procedure. In principle, when measuring the system's performance, we use 3-fold cross-validation to avoid biased results.

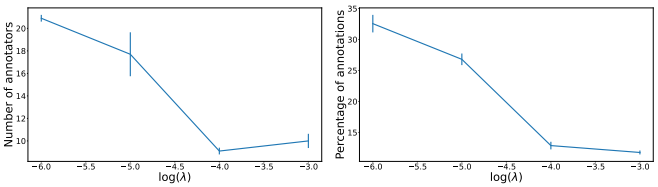
3.1. Amazon Mechanical Turk

We retrieve access to a real-life annotator set, consisting of results from an experiment with 21 workers at the *Amazon Mechanical Turk* [10]. These workers executed ten tasks; all of them were images' binary classification, particularly detecting dogs of assigned breeds on images. Since each worker has different expertise, the challenge is to determine a small subset of highly competent workers and combine their abilities for a superior classification system.



(a) Simple expertise model (b) Full expertise model

Fig. 1. Performance per task for different λ

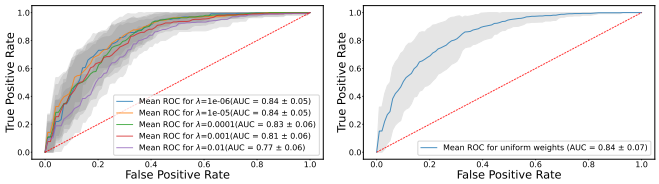


(a) Number of annotators for various $\log(\lambda)$ when using the simple model (b) Number of annotators for various $\log(\lambda)$ for the full expertise model

Fig. 2. Change in the number of annotators for different λ

Performance Breakdown. To examine the performance of our model, we attempt to use it in various conditions and numbers of selected annotators, while assigning a small amount of ground truth available (5% of training points). In Figure 1, we review the breakdown of AUC (area under ROC curve) performance measure per labeling task for different values of λ. Our classifier's performance does not significantly vary for different tasks in the case of the simple model. The results are also similar for varying λ, except for $\lambda = 0.01$, where the performance falls fast. This illustrates how the amount of sparsity enforcement regulated by λ can determine the resulting model performance, indicating the tradeoff between performance and sparsity. On the other hand, the full expertise model does not contain such a drastic fall in AUC. This indicates that an advantage of full expertise evaluation is a better stability of model performance. However, there is still some variation in performance per task in the full expertise model. Figure 2 shows how the sparsity in annotations and annotators increases with the change of λ in both simple and full expertise models. This change can be abrupt (e.g. from $\log \lambda = -5$ to $\log \lambda = -4$), therefore the right choice of hyperparameter is crucial for a successful model.

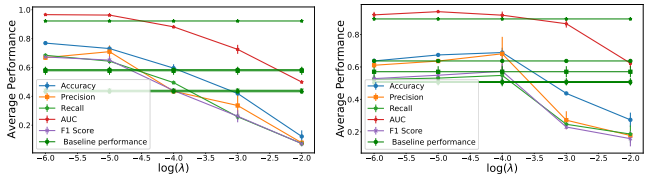
Comparison with Baseline Uniform Weights. We show a comparison of our scheme's performance for various λ to a baseline system where each annotator has an equal weight. In the baseline system, we keep the weight of annotators uniform; only the model per annotator (weights w) is optimized, and the hyperparameter λ does not have an effect. Figure 2 shows the change in the final number of annotators for different values of sparsity constant λ. For the value of 0.0001, we already have a fast fall in the number of annotators by over 50%. Nevertheless, the performance of our model does not



(a) ROC curves for different values of λ with the AUC results

(b) ROC curve for model with uniform annotator weights

Fig. 3. Comparison with different λ and the baseline for the full expertise model



(a) Performance of the simple model w.r.t. the baseline

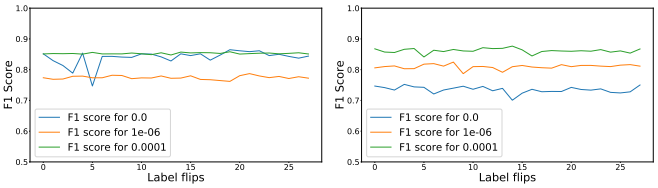
(b) Performance of the full expertise model w.r.t. the baseline

Fig. 4. Performance for the software defects dataset

significantly fall compared to the baseline, as can be noticed in Figure 3. This means that we still get a high-performance model with less than half of the annotators. Comparing the simple and full expertise model results confirms the previous observation about the stability of the model performance. We also see that the ROC results are more stable for different values of λ in the case of the full expertise model.

3.2. Software Defect Classification Dataset

This dataset contains the classification of over 600 software defects by impact, labeled by ten non-expert annotators. It is collected from the Mozilla project and comes from the paper of Hernandez et al. [15], which contains an investigation of the idea that the wisdom of crowds can be used to categorize software defects in the lack of experts that could provide a reliable assessment. Based on descriptive features of defects, the annotators classify software defects by impact into one of ten predefined classes. This dataset does not contain ground truth labels for software defects. Instead, we compute ground truth labels using a K-means clustering algorithm. We execute a test with a simple variant of our model and a full expertise model. Figure 4 shows the performance of our models on the defect reports dataset. These results show that we can achieve high performance for our software defects dataset when correctly setting up the λ parameter. The results highly depend on the value of this parameter. However, performance measures are a simple parameter function; thus, they can be easily adjusted. In the optimal assignment of the parameter λ, the performance measures can go over the baseline, determined by using uniform annotator weights. This means that



(a) White box attack (simple model)

(b) Grey box attack (simple model)

Fig. 5. Label flip results for various sparsity constraints

the annotators have different expertise, and their performance in annotating data should be considered in training time.

3.3. Evaluation of Robustness

Here we describe the results of our test of the robustness of our methodology to adversarial label flip attacks. Our test contains measurements of how the model performance depends on the amount of perturbation imposed on the system (number of label flips). We gradually flip labels for various annotations, thereby changing the test loss. We do this for specific steps and measure the change in the F1 score caused by these perturbations. We also compare the attack's success for different sparsity levels, enabling us to get evidence for the hypothesis that using sparsity enables the robustness of our model. This is indeed shown in Figure 5. The figures show a similar baseline, but higher variation in performance (lower stability) of results when testing models without added sparsity on annotator weights ($\lambda = 0$).

4. CONCLUSION

In this paper, we consider the problem of label-efficient learning from multiple annotators. We envision a learning scenario where each annotator receives the same input but creates different annotations due to a different expertise level. We design an architecture for a small number of annotators to create a performant learning system and test it on real-world data. We show that an optimized annotator selection model performs similarly to a model that uses all the available annotators, handy for systems where the data annotators' reliability and budget for annotations are limited. We also introduce a label flip test to assess our methodology's robustness to adversarial attacks. We show that an adversarial annotator's influence is limited due to our sparse weighting mechanism.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 883540 (PUZZLE)

5. REFERENCES

[1] James Surowiecki, Mark P Silverman, et al., "The Wisdom of Crowds," *American Journal of Physics*, vol. 75, no. 2, pp. 190–192, 2007.

[2] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy, "Learning From Crowds.," *Journal of Machine Learning Research*, vol. 11, no. 4, 2010.

[3] Mark E Nelson and Malcolm A MacIver, "Sensory Acquisition in Active Sensing Systems," *Journal of Comparative Physiology A*, vol. 192, no. 6, pp. 573–586, 2006.

[4] Yan Yan, Rómer Rosales, Glenn Fung, Ramanathan Subramanian, and Jennifer Dy, "Learning From Multiple Annotators With Varying Expertise," *Machine learning*, vol. 95, no. 3, pp. 291–327, 2014.

[5] Peter Welinder, Steve Branson, Serge J Belongie, and Pietro Perona, "The Multidimensional Wisdom of Crowds.," in *NIPS*, 2010.

[6] Yuandong Tian and Jun Zhu, "Learning From Crowds in the Presence of Schools of Thought," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012.

[7] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy, "Learning Whom to Trust With MACE," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013.

[8] Ping Zhang, Weidan Cao, and Zoran Obradovic, "Learning by Aggregating Experts and Filtering Novices: A Solution to Crowdsourcing Problems in Bioinformatics," *BMC bioinformatics*, vol. 14, no. 12, pp. S5, 2013.

[9] Filipe Rodrigues, Francisco C Pereira, and Bernardete Ribeiro, "Gaussian Process Classification and Active Learning With Multiple Annotators.," in *ICML*, 2014.

[10] Wei Bi, Liwei Wang, James T Kwok, and Zhuowen Tu, "Learning to Predict From Crowdsourced Data.," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2014.

[11] Joseph Wang, Kirill Trapeznikov, and Venkatesh Saligrama, "Efficient Learning by Directed Acyclic Graph for Resource Constrained Prediction," in *Advances in Neural Information Processing Systems*, 2015.

[12] H. Li and Q. Liu, ""Cheaper and Better: Selecting Good Workers for Crowdsourcing"," *ArXiv e-prints*, 2015.

[13] Battista Biggio, Giorgio Fumera, and Fabio Roli, "Security Evaluation of Pattern Classifiers Under Attack," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 4, pp. 984–996, 2013.

[14] Andrea Paudice, Luis Muñoz-González, and Emil C Lupu, "Label Sanitization Against Label Flipping Poisoning Attacks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2018.

[15] Jerónimo Hernández-González, Daniel Rodriguez, Iñaki Inza, Rachel Harrison, and Jose A Lozano, "Learning to Classify Software Defects From Crowds: A Novel Approach," *Applied Soft Computing*, vol. 62, pp. 579–591, 2018.