Binary Classification with Positive Labeling Sources

Jieyu Zhang  
University of Washington  
Seattle, United States  
jieyuz2@cs.washington.edu

Yujing Wang  
Microsoft Research Asia  
Beijing, China  
yuyjang@microsoft.com

Yaming Yang  
Microsoft Research Asia  
Beijing, China  
yayaming@microsoft.com

Yang Luo  
Microsoft Research Asia  
Beijing, China  
yangluo@microsoft.com

Alexander Ratner  
University of Washington  
Snorkel AI, Inc.  
Seattle, United States  
ajratner@cs.washington.edu

ABSTRACT
To create a large amount of training labels for machine learning models effectively and efficiently, researchers have turned to Weak Supervision (WS) [25], which uses programmatic labeling sources rather than manual annotation. Existing works of WS for binary classification typically assume the presence of labeling sources that are able to assign both positive and negative labels to data in roughly balanced proportions. However, for many tasks of interest where there is a minority positive class, negative examples could be too diverse for developers to generate indicative labeling sources. Thus, in this work, we study the application of WS on binary classification tasks with positive labeling sources only. We propose Weapo, a simple yet competitive WS method for producing training labels without negative labeling sources. On 10 benchmark datasets, we show Weapo achieves the highest averaged performance in terms of both the quality of synthesized labels and the performance of the final classifier supervised with these labels. We incorporated the implementation of Weapo into WRENCH [40], an existing benchmarking platform.

CCS CONCEPTS
• Computing methodologies → Semi-supervised learning settings.

KEYWORDS
Weak supervision; data programming; binary classification

ACM Reference Format:
Jieyu Zhang, Yujing Wang, Yaming Yang, Yang Luo, and Alexander Ratner. 2022. Binary Classification with Positive Labeling Sources. In Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM ’22), October 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557552

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CIKM ’22, October 17–21, 2022, Atlanta, GA, USA  
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.  
ACM ISBN 978-1-4503-9236-5/22/10...$15.00  
https://doi.org/10.1145/3511808.3557552

1 https://github.com/JieyuZ2/wrench/blob/main/wrench/labelmodel/weapo.py

1 INTRODUCTION
Weak Supervision (WS), one recent paradigm [25] for overcoming the challenge of low availability of training labels, has achieved remarkable success in various real-world applications [1, 13]. Specifically, in WS, expensive, time-consuming manual annotations are replaced with programmaticaly-generated labeling sources, called labeling functions (LFs), applied to unlabeled data. The usable labeling sources include but not limited to heuristics, knowledge bases, and pretrained models- they are typically cheaper than hand-labeling and able to provide potentially noisy labels to unlabeled data at scale. Despite the efficacy and efficiency of WS framework, it heavily relies on high-quality labeling sources to achieve satisfactory performance [39, 40]. In some real-world scenarios, in particular ones with a minority “positive” and majority “negative” class, the data of certain label could be highly diverse, making it difficult for practitioners to create labeling sources. For example, in bot detection where the user aim to detect bots from normal users, it is non-trivial even for experts to create labeling sources that identify the patterns of normal users, since the user behavior could be arbitrary. Note that such a motivation indeed drives the long-standing research problem of Positive-Unlabeled learning [3]. With regard to the importance of labeling sources and the difficulty of generating them for certain applications, in this work, we study the effectiveness of WS approaches under the setting where the labeling source of certain data class is absent. Specifically, we focus on binary classification with positive labeling sources only; in other words, the labeling source at-hand could only assign positive or abstain on any given data point.

We propose Weapo, a simple yet effective method for binary classification with positive labeling sources only. In particular, it is based on the intuition that data receiving more positive votes from the labeling sources are, in expectation, more likely to be positive examples. We formulate this simple idea as constraints on the inferred labels and construct a constrained optimization problem that seeks for a solution satisfying the constraints and minimizing the ℓ-2 regularization loss. On 10 benchmark binary classification datasets, we empirically demonstrate the efficacy of Weapo by showing that it offers highest performance on average. Specifically, we conduct experiments regarding both the quality of generated labels and the performance of a final classifier supervised with the labels. In both batches of experiments, we compare Weapo with modified versions of several existing WS methods as their
We denote scalar and generic items as lowercase letters, vectors as lowercase bold letters, and matrices as bold uppercase letters. For a vector \( \mathbf{v} \), we use \( v_i \) to represent its \( i \)-th value.

In binary classification, we aim to learn a scoring function \( g \) that could be used to build a binary classifier \( h(x) = \text{sign}(g(x) - \pi) \), where \( \pi \) is a threshold and \( x \) is a data point. In other words, the classifier \( h(x) \) maps data \( x \in X \) into binary label \( y \in \mathcal{Y} = \{-1, +1\} \). In standard supervised learning, we are given the ground truth label \( y = [y_1, y_2, ..., y_N] \) of the dataset \( D = \{x_i\}_{i \in [N]} \) for learning an optimal scorer \( g \), where \( N \) is the size of the dataset. However, for many classification tasks of interest, the collection of ground truth labels could be expensive and time-consuming. To tackle the challenge of low availability of ground truth labels, researchers have resorted to Weak Supervision [25, 37], which leverages programmatically generated, potentially noisy and correlated labeling sources to synthesize training labels. In this work, we follow this Weak Supervision setup and do not assume any ground truth label.

Formally, we have access to \( M \) labeling sources \( S = \{\lambda_j\}_{j \in [M]} \). For concreteness, we follow the general convention of Weak Supervision [25] and refer to these sources as labeling functions (LFs). Different from existing studies of Weak Supervision for binary classification that typically assume LFs could assign positive (+1), negative (-1), or abstain (0) to each data point, we are interested in the setting wherein we do not have negative LFs. We argue that such a setting is of importance because in real-world scenarios (1) high negative-class diversity may make constructing LFs prohibitively difficult [3], or (2) negative data may not be systematically recorded in some domains [2] and therefore it is difficult for developers to summarize labeling heuristics. Thus, in our setting, each LF \( \lambda_j \) either assigns positive label (+1) to a data or abstains (0), resulting in a label matrix \( L \in \{0, 1\}^{M \times N} \). Additionally, we assume the class prior \( p_+ = p(y = 1) \) is known following the convention of Weak Supervision [26]. We use \( \Lambda(x) \) to represent the output of LFs on data \( x \). We also use \( S^d \) to represent \( \{0, 1\}^d \) and \( \Lambda(x) \in \mathcal{B}^M \).

Given these LFs, our goal is to learn a label model \( f_0(\Lambda(x)) \) (short for \( f_0(x) \)), which is also a scoring function similar to \( g \) but inputs \( \Lambda(x) \) instead. It could be used to either directly make predictions on test data or provide supervisions for training a more complex end model \( g \) which inputs the data feature.

## 4 THE PROPOSED APPROACH

### 4.1 Conditional Moment Statistics

First, given a parameterized scoring function \( f_0(x) \) and a possible output of all the LFs \( v \in \mathcal{B}^M \), we define a moment statistic conditional on \( \Lambda(x) = v \) as

\[
E_{x:\Lambda(x)=v}[f_0(x)],
\]

which is the averaged score of \( f_0(x) \) over the set of data where \( \Lambda(x) = v \). Empirically, given the dataset \( D \) at-hand, the conditional moment statistics can be similarly defined as

\[
E_{x_i \in D, \Lambda(x_i)=v}[f_0(x_i)] = E_{x_i \in D_v}[f_0(x_i)] = \frac{1}{|D_v|} \sum_{x_i \in D_v} f_0(x_i),
\]

where \( D_v = \{x_i \in D | \Lambda(x_i) = v\} \).

### 4.2 A Partial Ordering of LFs Output

Then we introduce the covering relation between two binary vector \( v_1, v_2 \in \mathcal{B}^M \).

**Definition 4.1 (Covering Relation).** For \( v_1, v_2 \in \mathcal{B}^M \), \( v_1 \) is covered by \( v_2 \) if \( \forall i \in [M], v_2[i] \geq v_1[i] \) and \( \exists j \in [M], v_2[j] > v_1[j] \). We represent this covering relation using operator \( \triangleright \), e.g., \( v_2 \triangleright v_1 \).

For example, \( v_1 \) is covered by \( v_2 \) if \( v_1 = [1, 0, 0] \) and \( v_2 = [1, 1, 0] \); however for \( v_3 = [0, 0, 1] \), there is no covering relation between \( v_2 \) and \( v_3 \). Hence, the covering relation defines a partial ordering of elements in \( \mathcal{B}^M \).

### 4.3 Constrained Optimization

Now we describe our intuition. We expected that (in expectation) data have more positive votes should be more likely to be positive. Notably, we do not simply count the number of LFs assigning positive label as LF could be noisy, instead we resort to the covering
We simply parametrize the scoring function $f_\theta(x)$ as

$$f_\theta(x) = \Lambda(x) \theta^T,$$

(7)

where $\theta \in \Delta^M$ and $\Lambda(x) \theta^T$ is a convex combination of LFs output. Such a simple parametrization restricts the range of $f_\theta(x)$ to be $[0, 1]$ and therefore $f_\theta(x)$ could be interpreted as $P(y = 1|\Lambda(x))$. Then, we could further incorporate the label prior $p_+$ as a constraint. Specifically, we expect $\frac{1}{N} \sum_{i=1}^N f_\theta(x_i) = p_+$ and the final optimization problem becomes:

$$\min_{\theta \in \Delta^M} \lambda ||\theta||_2^2 + \sum_{i=1}^d \max(A_i f(x; \theta)^T, 0) + \left| \frac{1}{N} \sum_{i=1}^N f_\theta(x_i) - p_+ \right|$$

(8)

Such an optimization problem can be readily and efficiently solved by existing library, e.g., CVXPy [8].

5 EXPERIMENTS

5.1 Datasets

Throughout the experiments, we use the following 10 binary classification datasets from WRENCH [40], a comprehensive benchmark platform for Weak Supervision: **Census, Mushroom, Spambase, PhishingWebsites, Bioresponse, BankMarketing, CDR, SMS, Yelp, and IMDb**. Note that the first 6 datasets are tabular dataset while the remaining ones are textual datasets. For all the datasets, we only use the positive labeling functions provided by WRENCH. For textual datasets, we use pretrained BERT [7] to extract features.

5.2 Compared Methods

We compare **WEapo**, as well as its variant that does not leverage the label prior (**WEapo-prior**), with the following label models in the literature. **MV**: We adopt the classic majority voting (MV) algorithm as one label model. Notably, the abstaining LF, i.e., $\lambda_i = 0$ won’t contribute to the final votes. **DS [6]**: Dawid-Skene (DS) model estimates the accuracy of each LF with expectation maximization (EM) algorithm by assuming a naive Bayes distribution over the LFs’ votes and the latent ground truth. **MeTel [26]**: MeTel models the distribution via a Markov Network and recover the parameters via a matrix completion-style approach. **FS [14]**: FlyingSquid (FS) models the distribution as a binary Ising model, where each LF is represented by two random variables. A Triplet Method is used to recover the parameters and therefore no learning is needed, which makes it much faster than DS and MeTel. However, all existing label models assume the presence of negative LF that is absent in our setting. Thus, we treat the abstain (0) as negative (-1) so that existing label models are applicable.

5.3 Evaluation Protocol

First, we compare the performance of label models on test set. Notably, there is a subset of data not covered by any LF and therefore the label models have no information on them. Thus, we only evaluate the performance of label models on the covered subset of test data. We also found it is beneficial to treat these uncovered data as negative example when training the end model, since covered data are more likely to be positive, leaving the uncovered more likely to be negative.

Then, we compare the performance of end model trained with signals produced by label model. Notably, throughout the experiments, we do not use any clean labels as validation set for model selection, as it contradicts to our setting of absence of clean labels. We found that in such a setup the kernel ridge regression model with RBF kernel outperforms other options, e.g., linear regression, logistic regression, multi-layer perceptron classifier and so on.

For both evaluations, we adopt two common metrics for binary classification, namely, the Area Under the Receiver Operating Characteristic Curve (ROC-AUC score) and the Area Under the Precision-Recall curve (PR-AUC score), since they can be used to directly evaluate the scoring function $f_\theta(x)$.

5.4 Results: Label Models

The main results of label model comparison are presented in Table 1. First of all, we can see that **WEapo** achieves the highest averaged performance under both evaluation metrics and its variant **WEapo-prior** has the second best performance. This demonstrates the efficacy of **WEapo** as well as incorporating label prior as a constraint. However, **WEapo** does not outperform all the baselines on every datasets, which aligns with the recent results in a benchmarking study [40] that it is unlikely to have a universally-best label model.
Table 1: Label model comparison on covered test data. We highlight the best performing method in bold.

| Method   | Census | Mushroom | Spambase | Phishing Websites | Bioreponse | Bank Marketing | CDR | SMS | Yelp | IMDb | Average |
|----------|--------|----------|----------|-------------------|------------|----------------|-----|-----|------|------|---------|
| MV       | 58.28  | 66.02    | 68.25    | 65.41             | 57.73      | 67.97          | 57.81| 49.00| 66.31| 55.59| 60.44   |
| DS       | 37.19  | 66.71    | 64.73    | 30.96             | 57.99      | 48.92          | 47.26| 37.50| 69.13| 61.86| 52.22   |
| Snorkel  | 52.07  | 53.05    | 68.31    | 38.47             | 57.59      | 63.44          | 57.54| 32.50| 68.90| 50.66| 54.24   |
| FS       | 56.76  | 65.91    | 75.27    | 53.97             | 63.71      | 71.12          | 55.98| 32.50| 64.14| 49.05| 58.84   |
| **Weapo** | **64.91** | **62.58** | **71.10** | **57.52**         | **59.21**  | **62.98**      | **62.01**| **72.50**| **67.80**| **50.07**| **63.07** |
| **Weapo** | **56.72** | **68.58** | **75.15** | **67.22**         | **63.71**  | **64.88**      | **62.94**| **75.83**| **65.64**| **61.19**| **66.19** |

Table 2: End model comparison on test data. We highlight the best performing method in bold.

| Method   | Census | Mushroom | Spambase | Phishing Websites | Bioreponse | Bank Marketing | CDR | SMS | Yelp | IMDb | Average |
|----------|--------|----------|----------|-------------------|------------|----------------|-----|-----|------|------|---------|
| Gold     | 89.03  | 100.00   | 96.56    | 99.24             | 80.70      | 91.10          | 81.21| 99.90| 95.67| 89.32| 92.27   |
| Gold-covered | 83.71  | 100.00   | 89.40    | 97.90             | 72.96      | 89.88          | 81.13| 98.40| 94.81| 88.75| 89.69   |
| Gold-slice | 81.96  | 98.32    | 85.26    | 90.10             | 70.11      | 62.63          | 78.12| 98.30| 88.33| 85.81| 85.86   |
| Gold-train | 72.75  | 87.63    | 83.54    | 84.34             | 69.76      | 62.18          | 74.91| 96.16| 90.07| 85.42| 80.66   |
| MV       | 78.74  | 90.85    | 82.66    | **78.08**         | 69.71      | **84.48**      | 76.95| 92.92| 85.68| 84.38| 82.45   |
| DS       | 76.55  | 91.02    | **83.33** | 60.21             | 69.33      | 76.89          | 75.27| 90.76| **87.68**| **85.35**| 79.64   |
| Snorkel  | 80.50  | 85.25    | 82.95    | 67.62             | 69.62      | 80.66          | 77.24| **97.80**| **85.79**| **83.35**| 81.08   |
| FS       | 80.48  | 92.21    | 82.89    | 72.27             | 69.71      | 82.06          | 77.29| 97.63| 85.34| 83.93| 82.38   |
| **Weapo** | **79.30** | **90.59** | **82.44** | **77.37**         | **69.79**  | **83.87**      | **73.96**| **96.03**| **85.99**| **83.88**| **82.18** |
| **Weapo** | **79.12** | **91.98** | **83.15** | **78.01**         | **69.50**  | **84.48**      | **78.01**| **96.82**| **86.40**| **84.93**| **83.24** |

5.5 Results: End Models

For evaluation of end models, we additionally include four methods involving ground truth label for understanding the upper-bound performance we could achieve. Gold: it use all the ground truth labels of training data to train the end model; Gold-cover: it use the ground truth labels of covered training data to train the end model, since for uncovered data we do not have information for their underlying labels as no LF fires; Gold-slice: because label model inputs only the votes of LF, a group of data sharing the same votes would receive the same output from the label model, Gold-slice assigns the most-likely label for such a group of data, which is more close to the upper-bound performance any label model could offer; Gold-train: it use all the ground truth label to train a label model $f_0$ with the same model class as Weapo, i.e., $\theta \in \Delta^M$.

For the results in Table 2, we can first see that Weapo still achieve the highest averaged performance and again, it is not the best method for every dataset as observed in the label model comparison. Surprisingly, we found that Weapo is slightly better than Gold-train in terms of averaged performance, which further indicates the efficacy of our design. Finally, we conclude that even with positive labeling sources only. We propose Weapo, a simple yet effective method for such a novel setting. It leverages the intuition that data receiving more votes from positive labeling sources are in expectation more likely to be positive. Empirically, we compare Weapo with several baselines modified for this setting and show that it offers highest averaged performance.

REFERENCES

[1] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al.
