Learning with Constraint-Based Weak Supervision

Chidubem Arachie

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Bert Huang, Chair
Naren Ramakrishnan
Chandan Reddy
Hoda Eldardiry
Stephen Bach

March 24, 2022
Blacksburg, Virginia

Keywords: Machine Learning, Weak Supervision, Active Learning, Semi-supervised Learning, Adversarial Optimization

Learning with Constraint-Based Weak Supervision

Chidubem Arachie

ABSTRACT

Recent adaptations of machine learning models in many businesses has underscored the need for quality training data. Typically, training supervised machine learning systems involves using large amounts of human-annotated data. Labeling data is expensive and can be a limiting factor in using machine learning models. To enable continued integration of machine learning systems in businesses and also easy access by users, researchers have proposed several alternatives to supervised learning. Weak supervision is one such alternative. Weak supervision or weakly supervised learning involves using noisy labels (weak signals of the data) from multiple sources to train machine learning systems. A weak supervision model aggregates multiple noisy label sources called weak signals in order to produce probabilistic labels for the data. The main allure of weak supervision is that it provides a cheap yet effective substitute for supervised learning without need for labeled data. The key challenge in training weakly supervised machine learning models is that the weak supervision leaves ambiguity about the possible true labelings of the data. In this dissertation, we aim to address the challenge associated with training weakly supervised learning models by developing new weak supervision methods. Our work focuses on learning with constraint-based weak supervision algorithms. Firstly, we will propose an adversarial labeling approach for weak supervision. In this method, the adversary chooses the labels for the data and a model learns by minimising the error made by the adversarial model. Secondly, we will propose a simple constrained based approach that minimises a quadratic objective function in order to solve for the labels of the data. Next we explain the notion of data consistency

for weak supervision and propose a data consistent method for weakly supervised learning. This approach combines weak supervision labels with features of the training data to make the learned labels consistent with the data. Lastly, we use this data consistent approach to propose a general approach for improving the performance of weak supervision models. In this method, we combine weak supervision with active learning in order to generate a model that outperforms each individual approach using only a handful of labeled data. For each algorithm we propose, we report extensive empirical validation of it by testing it on standard text and image classification datasets. We compare each approach against baseline and state-of-the-art methods and show that in most cases we match or outperform the methods we compare against. We report significant gains of our method on both binary and multi-class classification tasks.

Learning with Constraint-Based Weak Supervision

Chidubem Arachie

GENERAL AUDIENCE ABSTRACT

Machine learning models learn to make predictions from data. In supervised learning, a machine learning model is fed data and corresponding labels for the data so that the model can learn to predict labels for new unseen test data. Curation of large fully supervised datasets is expensive and time consuming since it involves subject matter experts providing labels for each individual data example. The cost of collecting labels has become one of the major roadblocks for training machine learning models. An alternative to supervised training of machine learning models is weak supervision. Weak supervision or weakly supervised learning trains with cheap, and easy to define signals that noisily label the data. We refer to these signals as weak signals. A weak supervision model combines various weak signals to produce training labels for the data. The key challenge in weak supervision is how to combine the different weak signals while navigating misleading correlations in their errors. In this dissertation, we propose several algorithms for weakly supervised learning. We classify our methods as constraint-based weak supervision since weak supervision is provided as constraints to our algorithms. We use experiments on different text and image classification datasets to show that our methods are effective and outperform competing methods that we compare against. Lastly, we propose a general framework for improving the performance of weak supervision models by incorporating a few labeled data. With this method we are able to close the gap to supervised learning without the need for labeling all the data examples.

Dedication

Dedicated to my beloved parents and family, for their patience, love, support and encouragement.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, professor Bert Huang, for his unwavering support and guidance throughout the past five years of my graduate studies. Under his tutelage, I have grown as a researcher and learned a great deal both in my academic and personal life. He has been amazing and has committed a great deal of time in ensuring that I learn to think as a scientist and also do things the right way. I am incredibly grateful to him and I feel very fortunate to have had the opportunity to work with him. I also appreciate Professor Naren Ramakrishnan, Professor Chandan K. Reddy, Professor Hoda Eldardiry, and Professor Stephen Bach for their generosity in serving as my committee members. Their contributions and feedback helped me a great deal to better understand my work and also communicate it better. Their comments and suggestions improved the quality of my work and contents of this dissertation. Furthermore, I like to say thank you to my labmates and friends at Virginia Tech. Their friendship and support was essential in helping me navigate the fun and perils of graduate studies. I am lucky to have a group of friends that helped me create fun adventures and great memories. I also like to thank the co-authors, mentors and colleagues that I have had the opportunity to work with. Each work or paper helped me learn and deepen my knowledge on different topics. Lastly, I would like to thank my parents, siblings and other extended family members for their love, support and encouragement. I am very fortunate and grateful for their different contributions in helping me get to this point.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Successful demonstrations of machine learning systems in different fields have created an explosion of interest in deploying machine learning systems in business applications. The key driver of these successes is the progress in deep learning. Researchers in different industries are applying deep learning to their work with varying degrees of success. Deep learning enables machine learning systems to be trained without the need for feature engineering. Deep learning models make use of massive amounts of data and are able to successfully extract useful features from the data. For example convolutional neural network models use convolution operations to extract meaningful representations from images. The extracted features can be used for different downstream tasks such as image classification, object detection and segmentation. The high degree of performance achieved by these models is as a result of large amounts of data they consume. Moreover, these models are typically trained in a supervised meaning that they require a label for every training data point.

Curating large training data for supervised machine learning models is a major bottleneck to deploying machine learning models. Collecting labels for large training datasets is expensive and time consuming. An expert annotator is required to hand label each data example for quality assurance. In practice this is not always feasible due to time and cost constraints for businesses. Because of these limitations, researchers have turned to other supervised learning alternatives.

Weak supervision offers an alternative for training machine learning models without labels

because it relies on approximate labels that are easily obtained. Weakly supervised learning alleviates some of the difficulties and cost associated with supervised learning by only requiring annotators to provide rules or approximate indicators that automatically label the data. It uses domain knowledge about the specific problem, side information, or heuristics to approximate the true labels. Because annotators provide functions or rules that noisily label data, these functions can be applied to an indefinite amount of training data. The intent is that the cognitive cost of the annotator designing such rules may be slightly more than labeling individual examples, but the payoff is unbounded given an abundant source of unlabeled examples. For example, in a sentiment analysis task where the problem is to classify customer reviews of business products as positive or negative, weak supervision can generate noisy labels by identifying mentions of words in the reviews. The occurrence of positive words such as love, like, great in a review will result in positive labeling for that review while negative sentiments such as sucks, dislike or disappointed will give negative labels. Each word is a source of weak supervision and provides a noisy labeling of the data. We refer to these noisy labels as weak signals.

Weak signals are noisy and can conflict on their label estimates. From our example above, a customer review that says `I love this brand so much but I am very disappointed with this product` would be labeled as a negative sentiment by a human or expert annotator. However, our weak signal labels will conflict as the signal love will provide a positive label while the signal disappointed will provide a negative label. In addition to weak signals having conflicting label estimates, they can make correlated errors that could mislead a model if the weak signals are naively combined by majority voting or averaging. For example, the review `I love this brand so much and I like the packaging but I am very disappointed with the quality of this product` will generate two positive labels from love and like and one negative label from disappointed. Majority voting for these

signals will result in a positive sentiment assignment for the review, which is incorrect. Hence, the key problem in weakly supervised learning is how to reliably combine various weak signal sources to train an accurate model.

The work in this dissertation addresses the challenges in training machine learning models with weak supervision. We propose different algorithms for learning with constraint based weak supervision. We refer to our methods as constraint based since the weak supervision is provided as constraints to the learning model. An advantage of this approach is that we do not assume a family of distributions for the weak signals and the true labels. Assumptions about the labelers or dependence of the weak signals are hard to verify in practice and could cause a method to fail when they do not hold. Our algorithms optimize an objective function subject to the weak supervision constraints in order to produce probabilistic training labels for the data.

1.1 Main Contributions

The main contributions of this research are as follows.

1. Firstly, we develop a general framework for adversarial label learning. Adversarial Label Learning (ALL) trains a model to perform well under the worst possible conditions for the weak supervision. An adversary chooses labels for the data under the constraints provided by the weak supervision and a learner minimizes the errors of the adversarial labels. The training process of ALL can be viewed as a game between an adversary and a learner. We propose an algorithm for solving binary classification tasks with ALL then we extend our approach to cover multi-class classification problems.

2. Secondly, we propose Constrained Label Learning (CLL) to address some of the shortcomings of ALL. CLL minimizes a simple objective function in order to obtain probabilistic training labels from a constrained space. The true labels of the data lives in the constrained space and CLL tries to generate labels that are close to the true labels. We provided theoretical analysis of CLL algorithm and describe conditions for the weak supervision that can lead to optimal performance.

3. Next, we introduce Data Consistent Weak Supervision (DCWS), a method that takes as input features of the data together with the weak supervision and combines them to produce better training labels for the data. Our method considers different input features and offers flexibility on the choice of parametric model for the learner. We can switch the data consistency model depending on the classification task. Our data consistency approach outperforms our previous methods and other state-of-the-art weak supervision approaches in our experiments.

4. Lastly, we propose Weakly Supervised Pre-Training with Active Refinement (WSPAR), a framework for improving performance of weak supervision models using only a few labeled data. WSPAR combines weak supervision with active learning to choose labels for data points where the weak supervision predictions are uncertain. WSPAR overcomes limitations of both weak supervision and active learning by effectively combining both methodologies to improve on their individual performances.

1.2 Outline

The remaining chapters of this dissertation are organized as follows.

- Chapter 2 presents an extensive review of prior literature on learning with constraints,

adversarial learning and games and weak supervision.

- Chapter 3 introduces a general framework for adversarial label learning and describes algorithms for binary and multi-class classification with ALL.
- Chapter 4 presents constrained label learning and discusses theoretical analysis guarantees about the algorithm's performance.
- Chapter 5 describes data consistent weak supervision and discusses a thorough validation of the algorithm together with ablation studies.
- Chapter 6 explains weakly supervised pre-training with active refinement and shows how it can be used as a general framework for improving weak supervision.
- Chapter 7 summarizes our work, highlights its limitations, and discusses directions for future work.

Chapter 2

Literature Review

Our work builds on progress in three topic areas: weak supervision, learning with constraints, and adversarial learning.

2.1 Learning with Constraints

A predominant theme in label aggregation algorithms is estimating accuracy of weak labels without true labels. Methods for estimating accuracy of classifiers without labeled data has been studied by [2, 3, 4, 5, 6, 7]. These methods assume some knowledge about the true label distribution and then explores statistical relationships such as the agreement rates of the classifiers in other to estimate their accuracies. Our work is related to these methods in that we use error bounds that provide prior information but unlike these methods, we do not try to estimate the error bounds by making statistical assumptions of the weak signals.

Other methods like posterior regularization (PR) [8] and generalized expectation (GE) criteria [9, 10, 11] have been developed to incorporate human knowledge or side information into an objective function. A GE criterion [12] is a term in a parameter estimation objective function that prefers models to match conditional probabilities provided as weak supervision. These conditional probabilities may take the form of the probability of labels given a feature [9], also allowing the weak supervision to include information about the uncertainty of a weak signal. Posterior regularization (PR) [8] is a similar approach that trains models to

adhere to constraints on their output posterior distributions. These constraints can also take the form of weak supervision signals that specify the class of allowable posterior distributions for the learned model. While GE and PR allow incorporation of weak supervision and quantification of weak signal errors, they do not explicitly consider that these weak signals may make errors that conspire to confound the learner. Our algorithms aim to address this shortcoming.

2.2 Adversarial Learning and Games

Researchers have been increasingly interested in adversarial learning [13] as a method for training models that are robust to input perturbations of the data. These methods [14, 15, 16, 17, 18] regularize the learned model using different techniques to defend against adversarial attacks with an added benefit of improved generalization guarantees. Our adversarial methods focus on adversarial manipulation of the output labels to combat redundancy among multiple sources of weak supervision.

Game analyses are gaining importance in machine learning because they generalize optimization frameworks by assigning different objective functions for different players or optimizing agents. The generative adversarial network (GAN) [19] framework sets up a two player game between a generator and a discriminator, with the aim of learning realistic data distributions for the generator. Our methods do not learn a generative model but instead sets up a two-player game between an adversary that assigns labels for the classification task and a model that trains parameters to minimize a cross-entropy loss with respect to the adversarial labels. Researchers have recently examined convergence properties of smooth games [20, 21] and provided theoretical guarantees for them.

2.3 Weak Supervision

Research on aggregating labels for training machine learning systems dates back to early crowdsourcing literature. The first significant work in this research comes from Dawid and Skene [22]. The authors proposed a probabilistic model that uses expectation maximization to estimate the ground truth labels of the data. Various improvements and modifications [23, 24, 25, 26, 27, 28, 29, 30, 31] have been made to this original approach, with perhaps the most significant being the algorithm proposed by Zhou et al. [30]. The authors solve an optimization problem for the ground truth labels using a regularized minimax conditional entropy model. The algorithm has established itself in crowdsourcing literature and is known to be a competitive baseline. While crowdsourcing focuses on generating ground-truth labels from independent human annotators, our work makes no assumption about the error of the individual weak labelers. The weak signals our methods take in can be independent, dependent, or make correlated errors.

Weakly supervised learning algorithms provide another avenue for aggregating labels of training data. These algorithms have gained recent success, partially in part because they allow deep learning models to be trained using only user defined weak signals. A prominent weakly supervised approach is data programming [32], which allows users to combine weak signals via a generative model that estimates the accuracies and dependencies of the weak signals in order to produce probabilistic labels that label the data. Data programming has been implemented as the core of the popular *Snorkel* package for weakly supervised learning. Enhancements have been proposed to the original data programming algorithm [33, 34, 35, 36, 37, 38, 39], with each method proposing a different learning approach. Our work is related to Snorkel methods in that we combine different weak supervision signals to produce probabilistic labels for the training data. However, unlike Snorkel's methods, we

do not make probabilistic modelling assumptions on the joint distribution of the true labels and weak signals. Additionally, we do not take a generative approach for combining weak signals rather we solve an optimization to generate labels for the training data. Also, while Snorkel methods are data free and only use the weak signals to estimate the labels of the data, some of the methods we develop are data dependent and use features of the data to make the generated labels consistent with the data.

Our work is closely related to constraint-based weak supervision methods [40, 41, 42, 43]. These algorithms constrain the possible label space using the weak signals and the error rates of the weak signals and then solve an optimization problem to estimate the labels of the training data. A major advantage of these approach is that they do not make assumptions about the joint distribution of the labeling functions and weak signals. Assumptions about the weak supervision can be hard to obtain in practice and could cause a method to fail on a task. Similar to these methods, we can accept error bound of the weak signals as input when available, and we do not make assumptions on the joint distribution of the true labels and the weak signals. The difference between our work and these methods lies in the objective function or optimization approach of the different methods.

Other algorithms that combine the data with the weak supervision to estimate labels for unlabeled data include [1, 44]. These methods leverage labeled data for their label estimation. Unlike these methods, our work is completely weakly supervised and we do not need labeled data to train our model. Our method is also related to other approaches for learning from noisy labels [45, 46]. These methods focus on correcting for noise in labeled data while our work learns the labels from noisy weak supervision sources.

Ensemble methods such as boosting [47] combine different weak learners (low-cost, low-powered classifiers) to create classifiers that outperform the various weak learners. These weak learners are not weak in the same sense as weak supervision. These strategies are

defined for fully supervised settings. Our settings differs since we do not expect to have access to labeled data.

Weakly supervised methods have enabled knowledge extraction from the Web [48, 49, 50, 51, 52], visual image segmentation [53, 54], tagging of medical conditions from health records [55, 56] and for causal inference [57].

Our weakly supervised approach is built on the idea that it should be inexpensive for human experts or annotators to provide weak indicators of labels on different tasks. Our work focuses on combining these weak labels effectively. In the next sections, we present the weak supervision algorithms we developed.

Chapter 3

A General Framework for Adversarial Label Learning

3.1 Introduction

A key challenge for weak supervision is the fact that there may be bias in the errors made by the weak supervision signals. Using multiple sources of weak supervision can somewhat alleviate this concern, but dependencies among these weak supervision functions can be misconstrued as independent confirmation of erroneous labels. For example, in a classification task to identify diabetic patients, physicians know that obesity can indicate diabetes, and they also know the rate at which this indicator is wrong. However, since the indicator is biased, models trained with this information will learn to detect obesity, not the original goal of diabetes. To correct this problem, one may also consider high blood pressure as a second weak indicator. Unfortunately, these indicators are correlated and may make dependent errors.

To solve this, we introduce *adversarial label learning* (ALL), a method for training classifiers without labels by making use of weak supervision. ALL works by training classifiers to perform well on adversarially labeled instances that are consistent with the weak supervision. ALL aims to mitigate the problem of dependencies between weak signals by adversarially labeling the data. The adversarial labeling can construct scenarios where dependencies in

the weak supervision are as confounding as possible while preserving the partial correctness of the weak supervision. The learner then trains a model that can perform well against this adversarial labeling. ALL solves these two competing optimizations using primal-dual subgradient descent. The inner optimization finds a worst-case distribution of the labels for the current weight parameter of the model, while the outer optimization finds the best weights for the model for the current label distribution. The inner optimization's maximized error rate can also be viewed as an upper bound on the true error rate, which the outer optimization aims to minimize. By training to perform well on the worst-case labeling, ALL is robust against dependent and biased errors in weak supervision signals.

The inputs to ALL are a set of unlabeled data examples, a set of weak supervision signals that approximately label the data, and a corresponding set of estimated error bounds on these weak supervision signals. Domain experts can design the weak supervision signals—e.g., by defining approximate labeling rules—and they can use their knowledge to set bounds on the errors of these signals. When designing weak supervision signals, experts often have mental estimates of how noisy the signals are, so this error estimate is an inexpensive yet valuable input for the learning algorithm.

First, we consider a binary classification setting where a parameterized model is trained to classify the data by solving a competitive game against an adversary. We make use of multiple weak signals that represent different approximations of the true model. These weak signals can be interpreted as having different views of the data. The estimated error rates of these weak signals are passed as constraints to our optimization. Importantly, we show that ALL works in cases where these weak signals make dependent errors. Our experiments also show that ALL trains classifiers that are better than the weak supervision signals, even when the error estimates are incorrect. The performance of ALL in this setting is significant because domain experts will often imperfectly estimate the noisiness of the weak supervision

signals.

We then extend ALL to solve multiclass problems using additional linear constraints. We develop *Multi-ALL*, a general framework that encodes multiple linear constraints on the weak supervision signals. Unlike in the binary classification task, Multi-ALL learns from the weak supervision by solving a non-zero sum game between an adversary and the model. By using a non-zero sum game, our formulation provides more flexibility than in the binary case and allows for different loss functions to be used. Thus, we enable learning for other forms such as multiclass and multilabel classification. Multi-ALL is stochastic and uses different forms of weak supervision making it possible to train large scale models like deep neural networks.

We validate Multi-ALL on three image classification datasets. We use error and precision constraints to solve multiclass image classification tasks for deep neural network models. In the experiments, we provide weak supervision signals that are both generated by humans and programmatically generated. Our results show that our approach outperforms other weak supervision methods on deep image classification tasks. Our experiments also highlight the difficulties in providing adequate weak supervision signals for solving multiclass image classification tasks.

3.2 Adversarial Label Learning

The principle behind adversarial label learning (ALL) is that we train a model to perform well under the worst possible conditions. The conditions being considered are the possible labels of the training data. We consider the setting in which the learner has access to a training set of examples, and weak supervision is given in the form of some approximate indicators of the target classification along with expert estimates of the error rates of these indicators. Formally, let the data be $X = [x_1, \ldots, x_n]$. (We consider these examples to be

ordered for notational convenience, but the order does not matter.) These examples belong to classes $[y_1, \ldots, y_n] \in \{0,1\}^n$. The training labels $\boldsymbol{y}$ are unavailable to the learner. Instead, the learner has access to m weak supervision signals $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\}$, where each weak signal is a soft labeling of the data, i.e., $\boldsymbol{q}_i \in [0,1]^n$. These soft labelings are estimated probabilities that the example is in the positive class. In conjunction with the weak signals, the learner also receives estimated expected error rate bounds of the weak signals $\boldsymbol{b} = [b_1, \ldots, b_m]$. These values bound the expected error of the weak signals, i.e.,

$$b_i \geq \mathbb{E}_{\hat{\boldsymbol{y}} \sim \boldsymbol{q}_i} \left[\tfrac{1}{n} \textstyle\sum_{j=1}^{n} [\hat{y}_j \neq y_j] \right] \ , \tag{3.1}$$

which can be equivalently expressed as

$$b_i \geq \tfrac{1}{n} \left(\boldsymbol{q}_i^\top (1 - \boldsymbol{y}) + (1 - \boldsymbol{q}_i)^\top \boldsymbol{y} \right) \ . \tag{3.2}$$

While the learned classifier does not have access to the true labels $\boldsymbol{y}$, it will use the assumption that this bound holds to define the space of possible labelings. Let the current estimates of learned label probabilities be $\boldsymbol{p} \in [0,1]^n$. We relax the space of discrete labelings to the space of independent probabilistic labels, such that the value $\hat{y}_j \in [0,1]$ represents the probability that the true label y_j of example x_j is positive. The adversarial labeling then is the vector of class probabilities $\hat{\boldsymbol{y}}$ that maximizes the expected error rate of the learned probabilities subject to the constraints given by the weak supervision signals and bounds, which can be found by solving the following linear program:

$$\begin{aligned} \underset{\hat{\boldsymbol{y}} \in [0,1]^n}{\arg\max} \quad & \tfrac{1}{n} \left(\boldsymbol{p}^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{p})^\top \hat{\boldsymbol{y}} \right) \\ \text{s.t.} \quad & b_i \geq \tfrac{1}{n} \left(\boldsymbol{q}_i^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{q}_i)^\top \hat{\boldsymbol{y}} \right) , \ \forall i \in \{1, \ldots, m\} \ , \end{aligned} \tag{3.3}$$

which we present in this unsimplified form to convey the intuition behind its objective and constraints; some algebra simplifies this optimization into a more standard form.

The adversarial labeling described so far is a key component of the learning algorithm. ALL trains a parameterized prediction function f_θ that reads the data as input and outputs estimated class probabilities, i.e., $[f_\theta(x_j)]_{j=1}^n = \boldsymbol{p}$. We will write $\boldsymbol{p}(\theta)$ to mean $[f_\theta(x_j)]_{j=1}^n$ when it is important to note that these are generated from the parameterized function f. For now, we assume a general form for this parameterized function. For our optimization method described later in Section 3.2.2, we assume that the function f is sub-differentiable with respect to its parameters θ. The goal of learning is then to minimize the expected error relative to the adversarial labeling. This principle leads to the following saddle-point optimization:

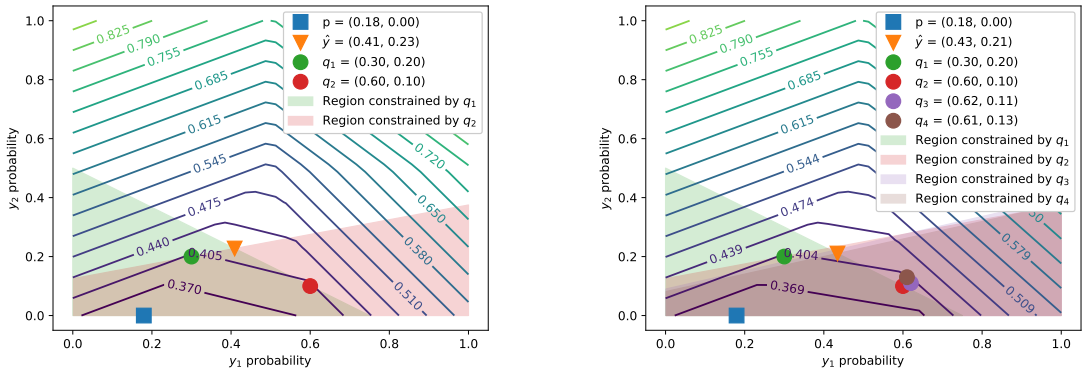
$$\begin{aligned} \min_{\theta} \quad \max_{\hat{\boldsymbol{y}} \in [0,1]^n} \quad & \tfrac{1}{n} \left(\boldsymbol{p}(\theta)^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{p}(\theta))^\top \hat{\boldsymbol{y}} \right) \\ \text{s.t.} \quad & b_i \geq \tfrac{1}{n} \left(\boldsymbol{q}_i^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{q}_i)^\top \hat{\boldsymbol{y}} \right), \ \forall i \in \{1, \ldots, m\} \ . \end{aligned} \tag{3.4}$$

We can view the outer optimization as optimizing a primal objective that is the maximum of the constrained inner optimization. Define this primal function as $g(\theta)$, such that Eq. (3.4) can be equivalently written as $\min_\theta g(\theta)$. If the weak supervision error bounds are true, *this primal objective value is an upper bound on the true error rate.* This fact can be proven by considering that the true labels $\boldsymbol{y}$ satisfy the constraints, and the inner optimization seeks a labeling $\hat{\boldsymbol{y}}$ that maximizes the classifier's expected error rate. In the next section, we visualize this primal function and the behavior of adversarial labeling before describing how we efficiently solve this optimization in Section 3.2.2.

3.2.1 Visualizing Adversarial Label Learning

In this section, we investigate a simple case that illustrates the behavior of the primal objective function g on a two-example dataset ($n = 2$). For a small dataset, we can visualize in two dimensions a variety of concepts.

In Fig. 3.1a, we illustrate the constraints set by the two weak supervision signals. The first signal $\boldsymbol{q}_1$ estimates that $\hat{y}_1$ is positive with probability 0.3 and that $\hat{y}_2$ is positive with probability 0.2. The second signal $\boldsymbol{q}_2$ estimates that $\hat{y}_1$ is positive with probability 0.6 and that $\hat{y}_2$ is positive with probability 0.1. The bounds for each weak signal error are set to $b_1 = b_2 = 0.4$. Note that both weak signals agree that $\hat{y}_2$ is most likely negative, but they disagree on whether $\hat{y}_1$ is more likely to be positive or negative.



(a) Two weak signals

(b) Redundant weak signals

Figure 3.1: Illustrations of the primal objective function from Eq. (3.4), the constraints set by the weak supervision, and the optimal learned probabilities and adversarial labels for a two-example problem.

Constraints on $\hat{\boldsymbol{y}}$ The shaded regions represent the feasible regions determined by the linear constraint corresponding to each weak signal. The intersection of these feasible regions is the search space for label vectors. Note how the pink region determined by $\boldsymbol{q}_2$ allows $\hat{y}_1$

to be either extreme of 0 or 1. With more examples ($n \gg 2$), the possibility of ambiguous labels increases significantly.

Primal Objective Function The contour lines illustrate the objective value of the primal function g, which finds the expected error for the adversarially set labels $\hat{\boldsymbol{y}}$. Since the adversarial inner optimization is a linear program, the solution jumps between vertices of the constrained polytope, making the primal expected error a piecewise linear convex function of $\boldsymbol{p}$.

Adversarial Labeling In Fig. 3.1a, the blue square is the minimum of the primal function, i.e., the solution to the ALL objective. This solution shows that the ideal learned model should predict $\hat{y}_1$ to be positive with probability 0.18 and $\hat{y}_2$ to be positive with probability 0. In the optimal state, the adversarial labeling of the examples is illustrated as the orange triangle at $(0.41, 0.23)$, i.e., the label probability vector that induces the most error for the current predicted probabilities $\boldsymbol{p}$ that still satisfies the constraints set by $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$.

Robustness to Redundant and Dependent Errors A key feature of ALL is that it is robust to redundant and dependent errors in the weak supervision. In Fig. 3.1b, we plot a variation of the setup from Fig. 3.1a, except we include two noisy copies of weak signal $\boldsymbol{q}_2$. Since our optimal solution disagreed with weak signal $\boldsymbol{q}_2$ on the most likely label for $\hat{y}_1$, one might expect that adding more weak signals that agree with $\boldsymbol{q}_2$ would "outvote" the solution and pull it to a higher probability of $\hat{y}_1$ being positive. But if weak signal $\boldsymbol{q}_2$ is highly correlated with weak signals $\boldsymbol{q}_3$ and $\boldsymbol{q}_4$, they may suffer from the same errors. Instead of these extra signals inducing a majority vote behavior on the solution, their effect on ALL is that they slightly change the feasible region of the adversarial labels, which leaves the optimum unchanged.

These two-dimensional visualizations illustrate the behavior of ALL on a simple input. In higher dimensions, i.e., when there are more examples in the training set, there is more freedom in the constraints set by each weak signal, so there will be more facets to the piecewise linear objective.

3.2.2 Optimization Approach

We use projected primal-dual updates for an augmented Lagrangian relaxation to efficiently optimize the learning objective. The advantage of this approach is that it allows inexpensive updates for all variables being optimized over, and it allows learning to occur without waiting for the solution of the inner optimization. The augmented Lagrangian form of the objective is

$$\begin{aligned} L(\theta, \hat{\boldsymbol{y}}, \boldsymbol{\gamma}) = \frac{1}{n} &\left(\boldsymbol{p}(\theta)^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{p}(\theta))^\top \hat{\boldsymbol{y}}\right) \\ &- \sum_{i=1}^{m} \gamma_i \left(\boldsymbol{q}_i^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{q}_i)^\top \hat{\boldsymbol{y}} - n b_i\right) \\ &- \frac{\rho}{2} \sum_{i=1}^{m} \left\| \left[\boldsymbol{q}_i^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{q}_i)^\top \hat{\boldsymbol{y}} - n b_i\right]_+ \right\|_2^2 , \end{aligned} \tag{3.5}$$

where $[\ \cdot\ ]_+$ is the hinge function that returns its input if positive and zero otherwise. This form uses Karush-Kuhn-Tucker (KKT) multipliers to relax the linear constraints on $\hat{\boldsymbol{y}}$ and a squared augmented penalty term on the constraint violation.

We then take projected gradient steps to update the variables θ, $\hat{\boldsymbol{y}}$, and $\boldsymbol{\gamma}$. The update step for the parameters is

$$\theta \leftarrow \theta - \frac{\alpha_t}{n} \left(\frac{\partial \boldsymbol{p}}{\partial \theta}\right)^\top (1 - 2\hat{\boldsymbol{y}}) \ , \tag{3.6}$$

where $\left(\frac{\partial \boldsymbol{p}}{\partial \theta}\right)$ is the Jacobian matrix for the classifier f over the full dataset and α_t is a gradient step size that can decrease over time. This Jacobian can be computed for a variety of models by back-propagating through the classification computation. The update for the adversarial

labels is

$$\hat{\boldsymbol{y}} \leftarrow \left[\hat{\boldsymbol{y}} + \alpha_t \left(\frac{1}{n}(1 - 2\boldsymbol{p}(\theta)) + \sum_{i=1}^{m} (\gamma_i (1 - 2\boldsymbol{q}_i) - \boldsymbol{v}_i \,) \right)\right]_0^1 , \tag{3.7}$$

where

$$\boldsymbol{v}_i = \rho(1 - 2\boldsymbol{q}_i) \left[\boldsymbol{q}_i^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{q}_i)^\top \hat{\boldsymbol{y}} - n b_i\right]_+ \quad ,$$

and $[\,\cdot\,]_0^1$ clips the label vector to the space $[0, 1]^n$, projecting it into its domain. The update for each KKT multiplier is

$$\gamma_i \leftarrow \left[\gamma_i - \rho \left(\boldsymbol{q}_i^\top (1 - \hat{\boldsymbol{y}}) + (1 - \boldsymbol{q}_i)^\top \hat{\boldsymbol{y}} - n b_i\right)\right]_+ , \tag{3.8}$$

which is clipped to be non-negative and uses a fixed step size ρ as dictated by the augmented Lagrangian method [58]. These primal-dual updates for the optimization converge in our experiments. Though L is not convex with respect to θ, it does satisfy some of the necessary conditions for convergence derived by Du and Hu [59]: The objective L is strongly convex in $\boldsymbol{p}$ and γ and concave in $\hat{\boldsymbol{y}}$, while the penalty term for the augmented Lagrangian is strongly convex. These properties may explain its convergence in practice. The full algorithm is summarized in Algorithm 1.

Algorithm 1 Adversarial Label Learning

Require: Dataset $X = [x_1, \ldots, x_n]$, learning rate schedule $\boldsymbol{\alpha}$, weak signals and bounds $[(\boldsymbol{q}_1, b_1), \ldots, (\boldsymbol{q}_m, b_m)]$, augmented Lagrangian parameter ρ.

1: Initialize θ (e.g., random, zeros, etc.)
2: Initialize $\hat{\boldsymbol{y}} \in [0, 1]^n$ (e.g., average of $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m$)
3: Initialize $\boldsymbol{\gamma} \in \mathbb{R}^m_{\geq 0}$ (e.g., zeros)
4: **while** not converged **do**
5: Update θ with Equation (3.6)
6: Update $\boldsymbol{p}$ with model and θ
7: Update $\hat{\boldsymbol{y}}$ with Equation (3.7)
8: Update $\boldsymbol{\gamma}$ with Equation (3.8)
9: **end while**
10: **return** model parameters θ

3.3 Experiments

We test adversarial label learning on a variety of datasets, comparing it with other approaches for weak supervision. In this section, we describe how we simulate domain expertise to generate weak supervision signals. We then describe the datasets we evaluated with and the compared weak supervision approaches, and we analyze the results of the experiments.

3.3.1 Simulating Weak Supervision

In practice, domain experts provide weak supervision in the form of noisy indicators or simple labeling functions. This weak supervision generates probabilities that the examples in a sample of the data belong to the positive class. Since we do not have explicit domain knowledge for the datasets used in our experiments, we generate the weak signals by training simple, one-dimensional classifiers on subsets of the data. The subset of the data used to train the weak supervision models is referred to as weak supervision data. We train each one-dimensional weak supervision model by selecting a feature and training a one-dimensional logistic regression model using only that feature. We select the weak supervision features based on our non-expert understanding of which features could reasonably serve as indicators of the target class. For datasets whose feature descriptions are not provided, we train the weak supervision models using the first feature, middle feature, and last feature. For the Fashion-MNIST, dataset we used the pixel value at the one-quarter, center, and three-quarter locations along the vertical center line (see Fig. 3.2) to build the respective weak supervision models.

We evaluate one-dimensional classifiers on the training subset, generating the weak signals $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\}$. In our first set of experiments, we measure the true error rate of each weak signal on the training subset and use that as the error bounds $\{b_1, \ldots, b_m\}$. In later experiments,

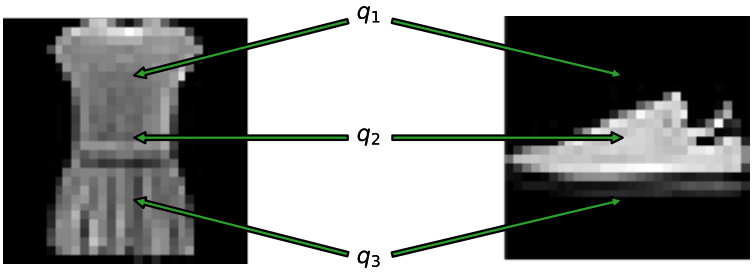


Figure 3.2: Features used to generate weak supervision signals on Fashion-MNIST data.

we set all bounds to 0.3 as an arbitrary guess. We train weak signals from one-dimensional inputs to create realistically noisy weak signals. Training on more features could increase the predictive accuracy of the weak signals and by extension ALL, but such high-fidelity weak signals may be rare in practice. Alternatively, we chose not to hand-design weak supervision signals and bounds, because doing so could inject our own bias into this evaluation. Simulating domain expertise with a small training set provides a neutral evaluation.

3.3.2 Compared Methods

We compare ALL against two baseline models: a modified generalized expectation (GE) method and averaging of weak signals (AVG).

Modified GE GE assigns a score to the value of a model expectation. Given a conditional model distribution and a reference distribution, GE uses a score function to measure the distance between the model expectation and reference expectation. We define a modified GE method to use the label distribution conditioned on each weak signal, i.e.,

$$\hat{p}_\theta\left(\boldsymbol{y}|\boldsymbol{q}_k \geq 0.5\right) = \mathbb{E}_{\hat{\boldsymbol{y}}}\left[\frac{1}{C_k} I(\hat{\boldsymbol{y}}) I\left(\boldsymbol{q}_k \geq 0.5\right)\right] \ , \tag{3.9}$$

and the reference expectation is

$$\tilde{p}\left(\boldsymbol{y}|\boldsymbol{q}_k \geq 0.5\right) = \mathbb{E}_{\boldsymbol{y}}\left[\tfrac{1}{C_k} I(\boldsymbol{y}) I\left(\boldsymbol{q}_k \geq 0.5\right)\right] \ , \tag{3.10}$$

where $\hat{\boldsymbol{y}}$ is the predicted labels and $C_k = \sum_{\boldsymbol{q}_k} I\left(\boldsymbol{q}_k \geq 0.5\right)$ is a normalizing constant and I is an indicator function. We compute these reference distributions on the training subset of the data. Our modified GE objective is then

$$\begin{aligned} \sum_{k=1}^{m} KL\left[\tilde{p}\left(\boldsymbol{y}|\boldsymbol{q}_k \geq 0.5\right) \| \hat{p_\theta}\left(\boldsymbol{y}|\boldsymbol{q}_k \geq 0.5\right)\right] + \\ KL\left[\tilde{p}\left(\boldsymbol{y}|\boldsymbol{q}_k < 0.5\right) \| \hat{p_\theta}\left(\boldsymbol{y}|\boldsymbol{q}_k < 0.5\right)\right] \ . \end{aligned} \tag{3.11}$$

We regularize this objective with an L2 penalty. This modified GE method is able to exploit the same information ALL is provided: the weak signals $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m$ and the reference distributions in Eq. 3.10 are analogous to (though richer than) the error bounds provided to ALL.

Averaging Baseline The input to our weakly supervised learning task includes the weak supervision signals $\boldsymbol{q}$, bounds $\boldsymbol{b}$, and the training set *without labels*. A straightforward approach that a reasonable data scientist could take to this training task is to compute pseudo-labels using the weak signals. Then one can train many classifiers using a standard supervised learning approach. For the averaging method, we generate baseline models by treating the rounded average of weak signals as a label. The averaging baseline tries to mimic the aggregated weak supervision. The averaging model trains a logistic regression classifier using the average of the weak signals' predictions as labels.

3.3.3 Experimental Setup

We run experiments on different datasets to measure the generalization and predictive power of adversarial label learning (ALL). Our first set of experiments measures the generalization of ALL on held out test set for different datasets. Then we provide experiments on crowdsourcing datasets, to show ALL's ability to learn predictive labels for the datasets.

For each dataset in the first experiments, we generate weak supervision signals and estimate their error rates. We then compare the accuracy of the model trained by ALL against (1) the modified GE baseline, (2) the different weak supervision signals and, (3) baseline models trained by treating the average of the weak supervision signals as labels. We randomly split each dataset such that 30% is used as weak supervision data, 40% is used as training data, and 30% is used as test data. For our experiments, we use 10 such random splits and report the mean of the results.

In each of our experiments, we consider three different weak signals. We run ALL on the first weak signal (ALL-1), the first and second weak signals (ALL-2), or all three weak signals (ALL-3). We use the sigmoid function as our parameterized function f_θ for estimating class probabilities of ALL and GE, i.e., $[f_\theta(x_j)]_{j=1}^n = 1/(1+\exp(-\theta^T x)) = \boldsymbol{p}_\theta$.

We compare against the accuracy of GE trained using the first weak signal (GE-1), the first and second weak signals (GE-2), or all three weak signals (GE-3). We also compare directly using the individual weak signals as the classifier (WS-1, WS-2, and WS-3). Additionally, we train models to mimic the average of the first weak signal (AVG-1), the first and second weak signals (AVG-2), all three weak signals (AVG-3). We also report the label accuracy of the weighted average of the weak signals (W-AVG). The accuracies of the weak signals are used as their weights. Finally, we show comparison to supervised learning for reference (SPV-L).

3.3.4 Datasets

We describe the datasets used in the experiments below.

Fashion-MNIST The Fashion-MNIST dataset [60] represents an image-classification task where each example is a 28×28 grayscale image. The images are categorized into 10 classes of clothing types. Each class contains 6,000 training examples and 1,000 test examples. We consider the binary classification between three pairs of classes: dresses/sneakers (DvK), sandals/ankle boots (SvA), and coats/bags (CvB).

Breast Cancer The task in this dataset is to diagnose if the breast cell nuclei are from a malignant (positive) or benign (negative) case of breast cancer [61, 62]. We use the mean radius of the nucleus (WS-1), the radius standard error (WS-2), and worst radius (WS-3) of the cell nucleus as features to train the three different weak supervision models. The dataset contains 569 samples.

OBS Network The classification task for the Burst Header Packet Flooding Attack Detection dataset is to detect network nodes based on their behavior, identifying whether they should be blocked for potentially malicious behavior [63]. We use the percentage of flood per node (WS-1), average packet drop rate (WS-2), and utilized bandwidth (WS-3) as features to train the weak signals. The original dataset contains four classes, so we select the two classes with the most examples, resulting in a total of 795 examples.

Cardiotocography The task for this dataset is to classify fetal heart rate using uterine contraction features on cardiotocograms classified by expert obstetricians [64]. The original dataset contains 10 classes, we select the most common two classes, resulting in a total of

963 examples. We use accelerations per second (WS-1), mean value of long-term variability (WS-2), and histogram median (WS-3) as features to train the weak signals.

Clave Direction The task for the Firm Teacher Clave Direction dataset is to classify the clave direction from rhythmic patterns [65]. The original dataset contains four classes, so we select the two most common classes, resulting in a total of 8,606 examples. We use the first (WS-1), middle (WS-2), and last (WS-3) features to train the weak signals.

Credit Card The Statlog German Credit Card dataset task is to classify people described by a set of attributes as good or bad credit risks [61]. We use the status of an existing checking account (WS-1), installment rate in percentage of disposable income (WS-2), and amount of existing credit at the bank (WS-3) as features to train the weak signals. The dataset contains 1,000 samples.

Statlog Satellite The task of the Statlog dataset is to predict soil class given the multi-spectral values of pixels in 3x3 neighborhoods of satellite images [61]. The original dataset contains seven classes of soil samples, so we select the two most common classes, resulting in a total of 3,041 examples. We use the first (WS-1), middle (WS-2), and last (WS-3) features to train the weak signals.

Phishing Websites The task is to identify phishing websites using different web attributes [66]. The dataset contains 11,055 samples. We use the URL of the anchor (WS-1), web traffic (WS-2), and Google index (WS-3) as features to train the weak signals.

Wine Quality The task is to classify the quality of wine using physiochemical attributes of the wine [67]. The original dataset contains seven classes, so we select the two classes

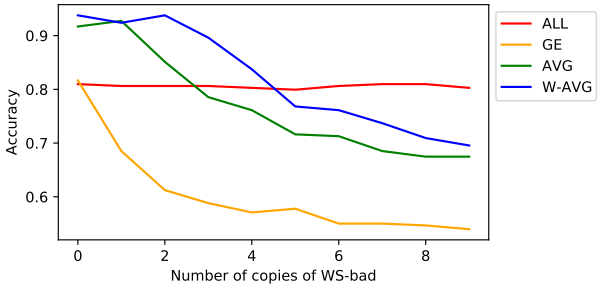


Figure 3.3: Performance of the methods using one good weak signal and repeated erroneous weak signals. ALL is able to learn a consistent classifier as we add redundant low performing weak signals.

with the most examples, resulting in a total of 4974 examples. We use fixed acidity (WS-1), density (WS-2), and pH (WS-3) as features to train the weak signals.

Table 3.1 shows the mean accuracies obtained by running ALL on the different datasets.

Dataset	ALL-1	ALL-2	ALL-3	GE-1	GE-2	GE-3	AVG-1	AVG-2	AVG-3	W-AVG	SPV-L
Fashion MNIST (DvK)	**0.998**	**0.995**	**0.996**	0.975	0.972	0.977	0.506	0.743	0.834	0.838	1.0
Fashion MNIST (SvA)	**0.923**	**0.922**	**0.924**	0.501	0.500	0.500	0.561	0.568	0.719	0.723	0.972
Fashion MNIST (CvB)	0.795	**0.831**	**0.840**	0.497	0.499	0.500	0.577	0.697	0.740	0.742	0.989
Breast Cancer	**0.942**	**0.944**	**0.945**	**0.936**	**0.936**	**0.935**	0.889	0.885	0.896	0.899	0.977
OBS Network	**0.717**	**0.718**	**0.719**	0.708	0.701	0.698	**0.724**	**0.723**	0.698	0.711	0.735
Cardiotocography	0.803	0.803	0.803	0.824	0.675	0.633	**0.942**	**0.947**	**0.942**	**0.945**	0.945
Clave Direction	0.646	**0.837**	0.746	0.646	0.796	0.772	0.646	0.645	0.707	0.711	0.964
Credit Card	**0.697**	**0.696**	**0.697**	**0.695**	0.460	0.424	0.660	0.662	0.607	0.626	0.719
Statlog Satellite	0.470	0.933	0.936	0.521	**0.987**	**0.992**	0.669	0.926	0.916	0.918	0.999
Phishing Websites	**0.896**	**0.895**	**0.895**	**0.898**	**0.894**	0.870	0.846	0.807	0.846	0.849	0.923
Wine Quality	0.572	**0.662**	0.623	0.455	0.427	0.454	0.570	0.573	0.555	0.582	0.685

Table 3.1: Test accuracy of ALL and baseline methods on different datasets with different amounts of weak signals. The best performing methods that are not statistically distinguishable using a two-tailed paired t-test ($p = 0.05$) are boldfaced. ALL is more consistently one of the best performing methods compared to generalized expectation (GE) and averaging (AVG). For reference, we include the accuracy for fully supervised learning (SPV-L).

3.3.5 Learning with True Bounds

Our first experiments allow ALL to use the error bounds computed on the training set. Table 3.1 shows the accuracies of the models evaluated on the held-out test sets of each

task. ALL trains models that perform significantly better than the weak signals and the baselines on the test data. The AVG baselines perform better with an increasing number of weak signals, but their best accuracy score on most datasets is significantly worse than that of ALL. There is not a big performance difference between AVG and W-AVG which uses the accuracies of the weak signals to compute its labels. ALL trains a robust model and is able to learn using noisy weak signals. Despite the fact that the weak signals on the Fashion MNIST dataset have rather low accuracy, ALL trained with these signals is able to achieve high accuracy. This is partly due to the fact that ALL considers the full features of the data in training the model. Hence, it is able to find similarities among neighboring pixels to make better classification decisions compared to the weak signal that considers only one pixel to generate its noisy label. GE also has access to the data during training, but GE only significantly outperforms ALL on the Statlog Satellite dataset. Nevertheless ALL still achieves a high accuracy score. The main failure case is the cardiotocography task, in which the AVG and W-AVG baselines outperforms both GE and ALL. However, in this task and others, we observe that ALL performs well even when the weak signals make dependent errors, while the baseline methods suffer as more signals with dependent errors are introduced. We study this concept further in the next experiment. Interestingly, we observe that ALL performance is close to that of supervised learning (SPV-L) on majority of the datasets.

3.3.6 Robustness against Dependent Errors

We observed from our test results that unlike the baselines, ALL learns a robust model that performs well even in the presence of low-quality weak signals. We isolate this concept using two weak signals from the cardiotocography task, a high-quality weak signal (WS-good) and a low-quality weak signal (WS-bad). We consider the scenario where the low-quality

Dataset	ALL-1	ALL-2	ALL-3	GE-1	GE-2	GE-3	AVG-1	AVG-2	AVG-3	WS-1	WS-2	WS-3
Fashion MNIST (DvK)	**0.998**	**0.995**	**0.996**	0.975	0.972	0.977	0.506	0.743	0.834	0.508	0.750	0.644
Fashion MNIST (SvA)	**0.895**	0.825	**0.901**	0.501	0.500	0.500	0.561	0.568	0.719	0.562	0.535	0.688
Fashion MNIST (CvB)	**0.810**	**0.805**	**0.802**	0.497	0.499	0.500	0.577	0.697	0.740	0.587	0.684	0.643
Breast Cancer	**0.940**	**0.941**	**0.944**	**0.936**	**0.936**	**0.935**	0.889	0.885	0.896	0.871	0.804	0.915
OBS Network	**0.719**	**0.719**	**0.722**	0.708	0.701	0.698	**0.724**	**0.723**	0.698	**0.721**	0.715	0.692
Cardiotocography	0.805	0.794	0.657	0.824	0.675	0.633	**0.942**	**0.947**	**0.942**	**0.946**	0.602	0.604
Clave Direction	0.646	**0.854**	0.727	0.646	0.796	0.772	0.646	0.645	0.707	0.646	0.648	0.625
Credit Card	**0.696**	0.671	0.610	**0.695**	0.460	0.424	0.660	0.662	0.607	0.659	0.572	0.557
Statlog Satellite	0.493	**0.983**	**0.982**	0.521	**0.987**	**0.992**	0.669	0.926	0.916	0.660	0.775	0.880
Phishing Websites	**0.899**	0.835	0.853	**0.898**	**0.894**	0.870	0.846	0.807	0.846	0.846	0.700	0.585
Wine Quality	0.566	0.603	**0.694**	0.455	0.427	0.454	0.570	0.573	0.555	0.571	0.596	0.570

Table 3.2: Test accuracy of ALL and baseline models on different datasets using fixed bounds. The best performing methods that are not statistically distinguishable using a two-tailed paired t-test ($p = 0.05$) are boldfaced. We replicate some baseline results from the previous experiments for convenience; they are unaffected by the change in error bound and additionally show the accuracies of the individual weak signals. We also include in this table the accuracies of each weak signal alone.

signal (WS-bad) is copied multiple times in the weak supervision. We train the models with WS-good and a varying number of copies of WS-bad. We evaluate the performance of the models on each experiment using the test data. Figure 3.3 plots the accuracy of the models under these settings. In the presence of multiple dependent erroneous weak signals, ALL's performance is relatively stable while the baseline accuracies get worse as the poor performing weak signal is repeated. The accuracy of AVG and W-AVG steadily degrades, while GE declines steeply to random performance.

3.3.7 Learning with Fixed, Incorrect Bounds

Instead of using the true training error as the bounds, we consider a more realistic scenario in which the experts are less precise about their error estimates. In practice, the true error rate may be difficult to estimate, so these experiments will validate whether our approach continues to work well when these bounds are inaccurate. We use a fixed upper bound of $b_1 = b_2 = b_3 = 0.3$ and report the performance of the ALL model and baselines in this

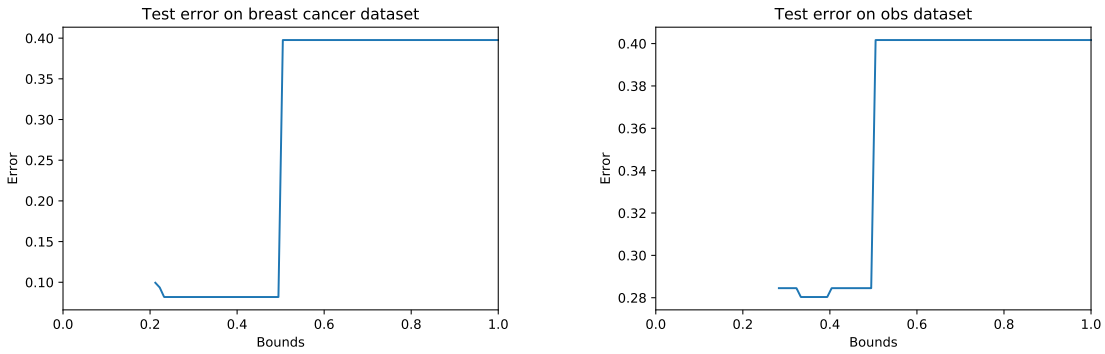


Figure 3.4: Error of the model (ALL-3) when run with different fixed bounds between 0 and 1. Small bound values make infeasible constraints that prevent convergence, and are not plotted here. Large error rate bounds give too much freedom to the adversary, and cause learning quality to significantly deteriorate. The experiments show that any bounds that imply the weak signals are better than random but admit feasible solutions produce similar quality learning.

setting.

Table 3.2 shows the accuracies obtained by the methods using the fixed bounds. The accuracy scores from the Statlog Satellite datasets are marginally higher than the results from the previous experiments, which used the true error rate (see Table 3.1), making it's performance statistically indistinguishable compared to GE.

While we arbitrarily chose a fixed bound of 0.3, we also tried various values of the bound, finding that ALL is not too sensitive to variations of this parameter. The only real challenge in setting this parameter is that when the bound is small enough, the problem becomes infeasible. See Fig. 3.4.

3.3.8 Crowdsourcing Experiments

We evaluate ALL on multiple datasets from the crowdsourcing domain measuring the predictive accuracy compared to the ground truth labels. We compare our method to a sim-

ple majority voting (**MAJ**), regularized minimax conditional entropy by Zhou et al. [30] (**MMCE**) and lastly to Snorkel MeTaL by Ratner et al. [34] (**MeTaL**). We use the true error rates of the weak signals as the bounds for the weak signals. The datasets we used for the experiments are described below.

RTE This dataset contains pairs of sentences labeled by whether the first sentence entails the second sentence [68]. There are 800 examples with 10 different crowd annotator labels for each example in the dataset. We use features provided by pre-trained BERT model [69] for the training data.

Word Similarity The task in this dataset is to label pairs of words as either similar or dissimilar [68]. There are 30 examples with 10 different crowd annotator labels for each example in the dataset. We use features provided by pre-trained BERT model [69] for the training data.

Blue Birds The classification task in this dataset is to identify the species of birds from photographs. The birds are either indigo buntings or blue grosbeaks. The dataset contains 108 images of birds with 39 annotator labels provided for each example. We use features provided by a pre-trained ResNet-101 model [70] for the training data.

Table 3.3 lists the accuracies of ALL on estimating labels for the datasets compared to competing approaches. ALL performs better than MeTaL, MMCE, and majority voting on the word similarity and blue birds datasets.

On the RTE dataset, its performance is inferior compared to competing methods. Investigating this, we discovered that the features from the embedding model were confusing for the learner, and using a reduced BERT embedding model as the features increases the accu-

Data	ALL	MAJ	MMCE	MeTaL
RTE	0.834	0.895	0.910	**0.913**
Word Similarity	**0.933**	0.867	0.867	0.9
Blue Birds	**0.935**	0.759	0.889	0.889

Table 3.3: Label accuracy of ALL and baseline models on different datasets. The best performing methods that are boldfaced.

racy to 0.9. We also observe that changing the learner $\boldsymbol{p}(\theta)$ from a simple logistic regression model to a two-layer neural network with non-linear activation function increases the performance. To further test the effect of changing the model family and data representation, we tried a two-layer neural network as the learner $\boldsymbol{p}(\theta)$ on the word similarity dataset, which achieved a better accuracy of 0.967. These extra results suggest that ALL's performance can be suboptimal if the model family and the data representation are mismatched.

3.3.9 Learning with Good Signals

In Section 3.3.6, we examined ALL's robustness to dependent error. In this section, we show how ALL synthesizes information from a combination of good and bad signals on annotator provided signals. We use the word similarity dataset. Starting by randomly selecting a single crowd worker's labels, we then iteratively add more weak crowd workers. For each number of crowd workers, we run 30 trials of randomly sampling that number of workers and calculate the mean accuracy. Figure 3.5 plots the mean accuracies of the models. On average, all methods get better as we increase the number of crowd annotator labels, but ALL gains the best performance compared to MMCE and majority vote. As more weak signals—or crowd workers—are added, each contributes a combination of redundant error and new information. ALL is able to utilize the information while avoiding the negative effects of redundant error to learn better labels for the dataset.

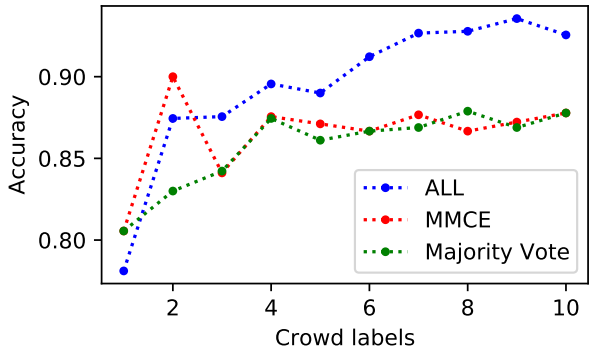


Figure 3.5: Performance of the methods as we increase the number of annotator labels from the crowd annotators in the word similarity dataset. For each number on the x-axis, we run 30 trials by randomly selecting that number of crowd labels from the dataset. We plot the mean accuracies for each of the 30 random trials.

3.4 Multi Adversarial Label Learning

In the previous sections, we covered ALL for the binary case. In this section, we introduce Multi-ALL (also referred to as Stochastic Generalized Adversarial Label Learning or Stoch-GALL), an extension of ALL for multiclass and multilabel cases. Multi-ALL takes as input an unlabeled dataset and a set of constraints. These constraints are consistent with the weak supervision and define the space of possible labelings for the data. We formulate a nonzero-sum game between two agents: the adversary that optimizes inferred labels for the data and the learner that optimizes parameters for the model. In this game, the objective of the adversary is to assign labels that maximize the error of the model subject to the provided constraints. The model objective is to minimize its loss with respect to the adversarial labels. Formally, let the unlabeled training data be $X = \{x_1, \ldots, x_n\}$, and let f be a classifier parameterized by θ. The primal form of the nonzero-sum game we solve for Multi-ALL is

$$\begin{aligned} &\min_{\theta} L^{(f_\theta)}(X) \quad \text{and} \\ &\max_{\boldsymbol{Y} \in \Delta_K^n} L^{(\boldsymbol{Y})}(X) \quad \textbf{s.t.} \;\; g_j(\boldsymbol{Y}) \leq 0, \forall j \in \{1, \ldots, m\}, \end{aligned} \tag{3.12}$$

where $L^{(f_\theta)}$ is a loss function for training the classifier, $L^{(\boldsymbol{Y})}$ is a loss function for the adversarial label perturbations, Δ_K^n is the space of *label matrices* where each row is on the simplex of dimension k (i.e., a matrix that can represent a set of n multinomial distributions), $\boldsymbol{Y}$ is the estimated label matrix, ℓ is a loss function, and $\{g_1, \ldots, g_m\}$ is a set of linear constraint functions on $\boldsymbol{Y}$. The estimated labels are optimized adversarially ($\max_{Y \in \Delta_C^n}$), against the objective of the learning minimization.

3.4.1 Linear Label Constraints

In this section, we describe examples of linear constraints that fit into the Multi-ALL framework. Let $\boldsymbol{q} \in [0,1]^n$ be a weak signal that indicates—in a one-versus-rest sense—the probability that each example is in class c. And let $\boldsymbol{y}_c$ denote the cth column of matrix $\boldsymbol{Y}$, which is the current label's estimated probability that each example is in class c. As shown in the binary case, one set of possible linear constraints that tie the adversarial labels to the true labels is a bound on the error rate of each weak signal. The expected empirical error for the one-versus-rest task under these two probabilistic label probabilities is

$$\text{error}(\boldsymbol{q}, \boldsymbol{y}_c) = \frac{1}{n}\left(\boldsymbol{q}^\top(1-\boldsymbol{y}_c) + (1-\boldsymbol{q})^\top \boldsymbol{y}_c\right) = \frac{1}{n}\left(\boldsymbol{q}^\top \mathbf{1} + \boldsymbol{y}_c^\top(1-2\boldsymbol{q})\right), \tag{3.13}$$

where we use $\boldsymbol{q}^\top \mathbf{1}$ as a vector notation for the sum of $\boldsymbol{q}$ (or its dot product with the ones vector). Combined with an annotator provided estimate of a bound (b_{error}) on reasonable errors for their weak signals, an error-based constraint function for weak signal $\boldsymbol{q}$ on class c would have form

$$g_{\text{error}}(\boldsymbol{Y}) = \frac{1}{n}\left(\boldsymbol{q}^\top \mathbf{1} + \boldsymbol{y}_c^\top(1-2\boldsymbol{q})\right) - b_{\text{error}}. \tag{3.14}$$

This error-based constraint function can be insufficient to capture the informativeness of a

weak signal, especially in cases where there is class imbalance. For multiclass classification, one-versus-rest signals will almost always be class-imbalanced. In such settings, we can allow annotators to indicate their estimates of weak-signal quality by indicating bounds on the precision. In the Multi-ALL setting, expected precision can also be expressed as a linear function of $\boldsymbol{Y}$:

$$\text{precision}(\boldsymbol{q}, \boldsymbol{y}_c) = \frac{\boldsymbol{y}_c^\top \boldsymbol{q}}{\boldsymbol{q}^\top \mathbf{1}}. \tag{3.15}$$

Since $\boldsymbol{q}$ is a constant with respect to the learning optimization, its appearance in the denominator of this expression does not affect the linearity. We can then define a precision constraint function for each weak signal $\boldsymbol{q}$ on class c:

$$g_{\text{prec.}}(\boldsymbol{Y}) = b_{\text{prec.}} - \frac{\boldsymbol{y}_c^\top \boldsymbol{q}}{\boldsymbol{q}^\top \mathbf{1}}. \tag{3.16}$$

Including precision constraints better captures the confusion matrix across different classes. It is also possible to design other linear constraints. As long as the constraints are linear, the feasible region for the maximization over $\boldsymbol{Y}$ remains convex.

3.4.2 Nonzero-Sum Losses

We describe here the loss functions we use to instantiate the Multi-ALL framework. The loss functions for the game must be differentiable; but, the choice of loss function is task-dependent and can have important impact on optimization. For multiclass classification using deep neural networks, our model uses popular cross-entropy loss. However, for the adversarial labeling, we instead use an expected error as the loss function, which the adversary

maximizes. Formally, the model's loss is the cross-entropy

$$L^{(f_\theta)} = -\frac{1}{n}\sum_{c=1}^{K} \boldsymbol{y}_c^\top \log(f_\theta(X)), \tag{3.17}$$

while the adversarial labeler's loss is the expected error

$$L^{(\boldsymbol{Y})} = \frac{1}{n}\sum_{c=1}^{K} f_\theta(X)^\top (1 - \boldsymbol{y}_c). \tag{3.18}$$

We choose this form of error loss because it is linear in the adversarially optimized variable $\boldsymbol{Y}$. This makes the objective for the adversary $\boldsymbol{Y}$ concave, so we are maximizing a concave function subject to linear constraints. This makes the adversarial optimization a linear program with a unique optimum for any fixed f_θ. This form is relevant for the initialization scheme described in Section 3.4.3. We optimize the loss functions using Adagrad [71].

3.4.3 Optimization

We use a primal-dual optimization that jointly solves an augmented Lagrangian relaxation of Eq. (3.12). Since our formulation uses a nonzero-sum game, we have two separate optimizations. The analogous optimizations are

$$\begin{aligned} &\min_{\theta} L^{(f_\theta)} \quad \text{and} \\ &\min_{\boldsymbol{\gamma}\in\mathbb{R}_+^m} \max_{\boldsymbol{Y}\in\Delta_K^n} L^{(\boldsymbol{Y})} - \boldsymbol{\gamma}^\top G(\boldsymbol{Y}) - \frac{\rho}{2}||G(\boldsymbol{Y})||_+^2 := \hat{L}^{(\boldsymbol{Y})}\ , \end{aligned} \tag{3.19}$$

where $\boldsymbol{\gamma}$ is the vector of Karush-Kuhn-Tucker (KKT) multipliers, G is the vector of constraint function outputs (i.e., $G(\boldsymbol{Y})_j := g_j(\boldsymbol{Y})$), ρ is a positive parameter, and $||\cdot||_+^2$ denotes the norm of positive terms. The adversary optimization maximizes the linear loss from Eq. (3.18)

while the learner minimizes the model loss from Eq. (3.17). A primal-dual solver for this problem updates the free variables using interleaved variations of gradient ascent and descent. We do this by updating $\boldsymbol{Y}$ and $\boldsymbol{\gamma}$ with their full gradients while holding θ fixed. Afterwards, we fix $\boldsymbol{Y}$ and $\boldsymbol{\gamma}$ and update θ with its mini-batches for a fixed number of epochs. The number of epochs depends on the size and architecture of the model network.

To preserve domain constraints on the variables $\boldsymbol{Y} \in \Delta_K^n$ and $\boldsymbol{\gamma} \geq 0$, we use projection steps that enforce feasibility. After each update to $\boldsymbol{Y}$ and $\boldsymbol{\gamma}$, we project $\boldsymbol{Y}$ to the simplex using the sorting method [72], and we clip $\boldsymbol{\gamma}$ to be non-negative.

Initialization Scheme

To further facilitate faster convergence toward a local equilibrium, we warm start the optimization with a phase of optimization updating only $\boldsymbol{Y}$ and $\boldsymbol{\gamma}$. The effect of this warm-start phase is that we begin learning with a near-feasible $\boldsymbol{Y}$—one that is nearly consistent with the weak-supervision-based constraints. Since this phase uses the fixed output of a randomly initialized model f_θ, it does not require forward- or back-propagation through the deep neural network, so it is inexpensive, even for large datasets.

Analysis

The advantage of this primal-dual approach is that it enables inexpensive updates for the gaming agents and other variables being optimized, thereby allowing learning to occur without waiting for the solution of the inner optimization. At every iteration, the primal variables take maximization steps and the dual variables take minimization steps. However, for training deep neural networks, the primal-dual approach is not always ideal.

The large datasets needed to fit large models such as deep neural networks often require

stochastic optimization to train efficiently. The key computational benefit of stochastic optimization is that it avoids the $O(n)$ cost of computing the true gradient update. Using a primal-dual approach to optimize Eq. (3.19) would also incur an $O(n)$ cost for each update to $\boldsymbol{Y}$. This cost is why we design our optimization scheme to update $\boldsymbol{y}$ and $\boldsymbol{\gamma}$ only after a fixed number of epochs. Since each epoch costs $O(n)$ computation, the added overhead does not change the asymptotic cost of training.

This optimization scheme has an added benefit that it increases the stability of the learning algorithm. By updating $\boldsymbol{Y}$ and $\boldsymbol{\gamma}$ only after a few epochs of training θ, we are solving the minimization over θ nearly to convergence. We still retain the advantages of primal-dual optimization over the $\boldsymbol{Y}$ variables, but without the added instability of simultaneous nonconvex optimization.

Algorithm 2 Multiclass Adversarial Label Learning

Require: Dataset $X = [x_1, \ldots, x_n]$, vector of constraint functions G, augmented Lagrangian parameter ρ.

Initialize model parameters θ (e.g., deep neural network weights)
Initialize $\boldsymbol{Y} \in \Delta_K^n$ (e.g., uniform probability)
Initialize $\boldsymbol{\gamma} \in \mathbb{R}_{\geq 0}$ (e.g., zeros)
while $G(\boldsymbol{Y}) >$ tolerance **do**
    Update $\boldsymbol{Y}$ with gradient $\nabla_{\boldsymbol{Y}} \hat{L}^{(\boldsymbol{Y})}$ (e.g., $\boldsymbol{Y} \leftarrow \boldsymbol{Y} + \alpha \nabla_{\boldsymbol{Y}} \hat{L}^{(\boldsymbol{Y})}$)
    Project $\boldsymbol{Y}$ to Δ_K^n
    Update $\boldsymbol{\gamma}$ with gradient $\nabla_{\boldsymbol{\gamma}} \hat{L}^{(\boldsymbol{Y})}$ (e.g., $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma} - \rho G(\boldsymbol{Y})$)
    Clip $\boldsymbol{\gamma}$ to be non-negative
end while
while θ not converged **do**
    Update θ with $\nabla_\theta L_{\mathcal{B}}^{(f_\theta)}$ (mini-batches) for a fixed number of epochs
    Update $\boldsymbol{Y}$ with gradient $\nabla_{\boldsymbol{Y}} \hat{L}^{(\boldsymbol{Y})}$
    Project $\boldsymbol{Y}$ to Δ_K^n
    Update $\boldsymbol{\gamma}$ with gradient $\nabla_{\boldsymbol{\gamma}} \hat{L}^{(\boldsymbol{Y})}$
    Clip $\boldsymbol{\gamma}$ to be non-negative
end while
return model parameters θ

3.5 Experiments

We validate our approach on three fine-grained image classification tasks, comparing the performance of models trained with our approach to a baseline averaging method and model trained using labels generated from Snorkel [73]. Each of these methods trains from weak signals, and our experiments evaluate how well they can integrate noisy signals and how robust they are to confounding signals. We also add a supervised learning baseline for reference.

3.5.1 Quality of Constrained Labels

Before using our custom weak annotation framework, we first compare the quality of labels generated by Multi-ALL to existing methods for fusing weak signals. We follow the experiment design from a tutorial[1] designed by Ratner et al. [73] to demonstrate their Snorkel system's ability to fuse weak signals and generate significantly higher quality labels than naive approaches. The experiment uses the *Microsoft COCO: Common Objects in Context* [74] dataset to train detectors of whether a person is riding a bike within each image. We use the 903 images from the tutorial and weak signals generated by the labeling functions based on object occurrence metadata. We calculate the error and precision of each rule and use those to define Multi-ALL constraints. We then run the initialization scheme (the first while loop in Algorithm 2), which finds feasible labels adversarially fit against a random initialization, i.e., arbitrary feasible labels. For an increasing number of weak signals, we compare ALL with error constraints, Multi-ALL with both error and precision constraints, Snorkel, and majority voting.

[1] https://github.com/HazyResearch/snorkel/blob/master/tutorials/images/Images_Tutorial.ipynb

We plot the resulting error rate of the generated labels in Figure 3.6. For all numbers of weak signals, Multi-ALL obtains the highest accuracy labels. The labels generated by Snorkel have the same label error using two and three weak signals, but adding additional weak signals starts to confound Snorkel. Our framework is not confounded by these additional weak signals. Finally, corroborating the results reported by Snorkel's designers, the naive majority vote method has significantly higher error compared to any of the more sophisticated weak supervision techniques.

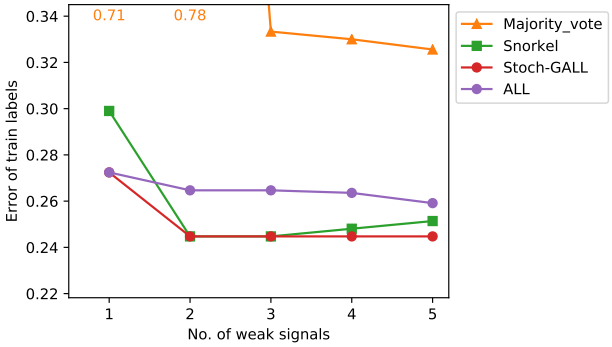


Figure 3.6: Error of MSCOCO bike-riding labels using Multi-ALL initialization compared to other methods.

3.5.2 Multiclass Image Classification

For our other experiments, we train multiclass image classifiers from weak supervision. We are interested in evaluating the effectiveness of the weak supervision approach, so we use the same deep neural network architecture for all experiments: a six-layer convolutional neural network where each layer contains a max-pooling unit, a relu activation unit, and dropout. The final layer is a fully connected layer with softmax output. Table 3.4 lists summary results of the error obtained by each method on each dataset using all the weak signals we provide the learners. The final result for each experiment is that Multi-ALL outperforms both Snorkel and averaging in all settings, showing a strong ability to fuse noisy signals and

to avoid being confounded by redundant signals. We describe our form of weak supervision and each experiment in detail in the rest of the section.

Weak Signals

We ask human annotators to provide weak signals for image datasets. To generate each weak signal, we sample 50 random images belonging to different classes. We then ask the annotators to select a representative image and mark distinguishing regions of the image that indicate its belonging to a specific class. We then calculate pairwise comparisons between the pixels in the region of the reference image selected by an annotator and the pixels in the same region for all other images in the dataset. We measure the Euclidean distance between the pairs of images and convert the scores to probabilities with a logistic transform. Through this process, an annotator is guiding the design of simple nearest-neighbor one-versus-rest classifiers, where images most similar to the reference image are more likely to belong to its class. We ask annotators to generate many of these rules for the different classes, and we provide the computed probabilities as weak labels for the weakly supervised learners.

In practice, we found that these weak signals were noisy. In some experiments, they were insufficient to provide enough information for the classification task. However, our experiments show how different weakly supervised learners behave with informative but noisy signals. We discuss ideas on how to design better interfaces and better weak signals for the image classification task in Section 3.6.

We assume we have access to a labeled validation set consisting of 1% of the available data. We use this validation set to compute the precision and error bounds for the weak signals. This validation set is meant to simulate a human expert's estimate of error and precision. To encourage a fair comparison, we allow all methods to use these labels in addition to weak

Data	Multi-ALL	Average	Snorkel	Supervised
Fashion-MNIST (weak)	**0.335**	0.447	0.401	0.142
Fashion-MNIST (pseudolabels + weak)	**0.228**	0.315	0.320	0.142
SVHN (pseudolabels + weak)	**0.231**	0.435	0.525	0.141

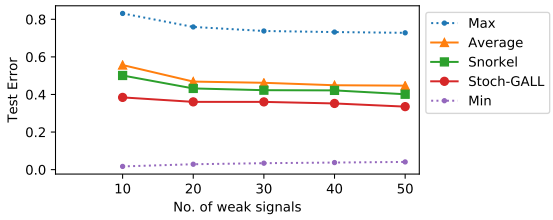
Table 3.4: Errors of models trained using all weak signals. In all three settings, Multi-ALL is able to train higher accuracy models than Snorkel or average labels. The settings include using all human-annotated weak signals (weak) and combining the human signals with pseudolabels (pseudolabels + weak).

signals when training by appending the validation set to the dataset with its true labels. Since these bounds are evaluated on a very tiny set of the training data, they are noisy and prone to the same type of estimation mistakes an expert annotator may make. Therefore, they make a good test for how robust Multi-ALL is to imperfect bounds.

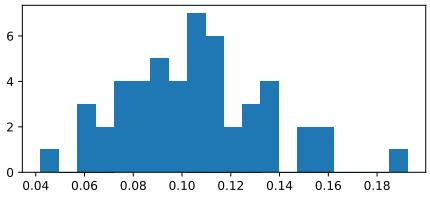
Weakly Supervised Image Classification

In this experiment, we train a deep neural network using only human-provided weak labels as described in Section 3.5.2. We use the *Fashion-MNIST* [60] dataset, which represents an image-classification task where each example is a 28×28 grayscale image. The images are categorized into 10 classes of clothing types with 60,000 training examples and 10,000 test examples. We have annotators generate five one-versus-rest weak signals for each class, resulting in 50 total weak signals.

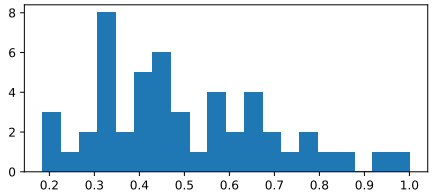
We plot analyses of models trained using weak supervision in Figure 3.7, where Fig. 3.7a plots the test error, and Fig. 3.7b and Fig. 3.7c are histograms of the error and precision bounds for the weak signals evaluated on the validation set. Since our weak signal is a one-versus-rest prediction of an image belonging to a particular class, the baseline precision and error should be 0.1 for training data with balanced classes. The histograms indicate that there is a wide range of precisions and errors for the different weak signals. Note that the order of



(a) Test error



(b) Error bounds



(c) Precision bounds

Figure 3.7: Analyses of experiments using the Fashion-MNIST dataset with human-provided weak signals.

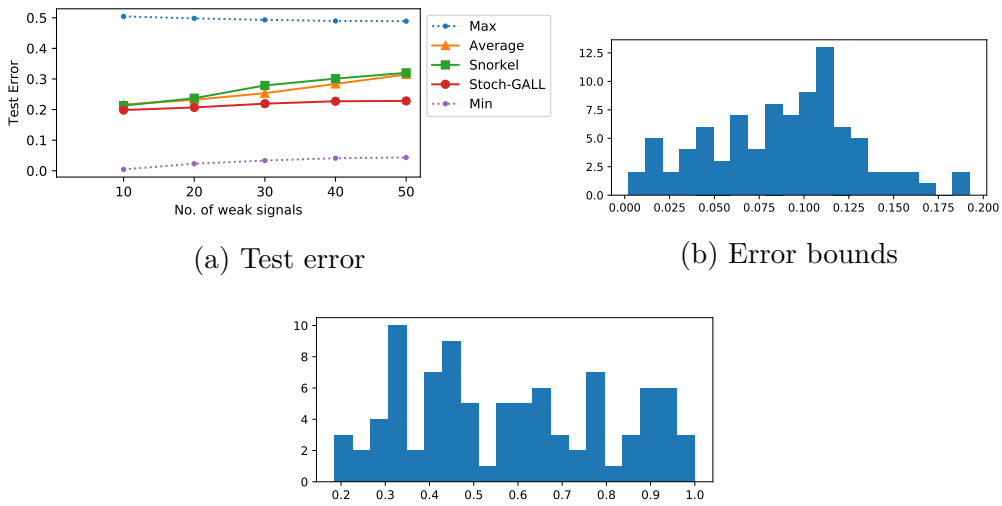
the weak signals in our experiment was fixed as the order provided by the annotators. Thus, the first weak signal for each task is the first weak signal that the annotator generated for that dataset. The error rates in Fig. 3.7a suggest that the test error of the models decreases as we add more weak signals. Multi-ALL with both precision and error bounds outperforms Snorkel and the average baseline for all the weak signals. The min and max curves in the plots represent the best and worst possible label errors for labels that satisfy the provided constraints. The high error in the max curve indicates that the constraints alone still allow highly erroneous labels, yet our Multi-ALL framework trains models that perform well. The min curve indicates how close feasible adversarial labels could be to the true labels. In this experiment, the min curve is close to zero, which suggests that the inaccuracies in the provided bounds are not overly restrictive.

Weak-Pseudolabel Image Classification

In the previous experiments, the human provided weak signals are informative enough to train models to perform better than random guessing, but the resulting error rate is still significantly lower than that of supervised methods. To further boost the performance, we combine the human weak signals with *pseudolabels*: predictions of our deep model trained on the validation set and applied to the unlabeled training data [75]. By training on the available 1% labels and predicting labels for the remaining 99% unlabeled examples, we create a new, high-quality weak signal. We calculate error and precision bounds for the pseudolabels with four-fold cross-validation on the validation set. We report the results of the models trained on the Fashion-MNIST dataset using this combination of pseudolabels and human weak signals.

Fig. 3.8 contains plots of the results. The error and precision histograms now include higher precision bounds and lower error bounds, as a result of the pseudolabel signals being higher quality than the human provided weak signals. Additionally, the min and max error curves have lower values, indicating that we get better quality labels with these signals. The error trends in Fig. 3.8a are quite different compared to the previous experiment (Fig. 3.7a). All the methods have good performance with the pseudolabel signals, but as we add the human signals, Snorkel and the average baseline are confounded and produce increasingly worse predictions. Multi-ALL however is barely affected by the human weak signals. The slight variation in the curve can be attributed to the inaccuracy of estimated bounds for the weak signals.

We hypothesize that this trend occurs because of the nature of our weak supervision. Since the weak signals are based on the selection of exemplar images, they may be effectively subsumed by a fully semi-supervised approach such as pseudolabeling. That is, the information



(a) Test error

(b) Error bounds

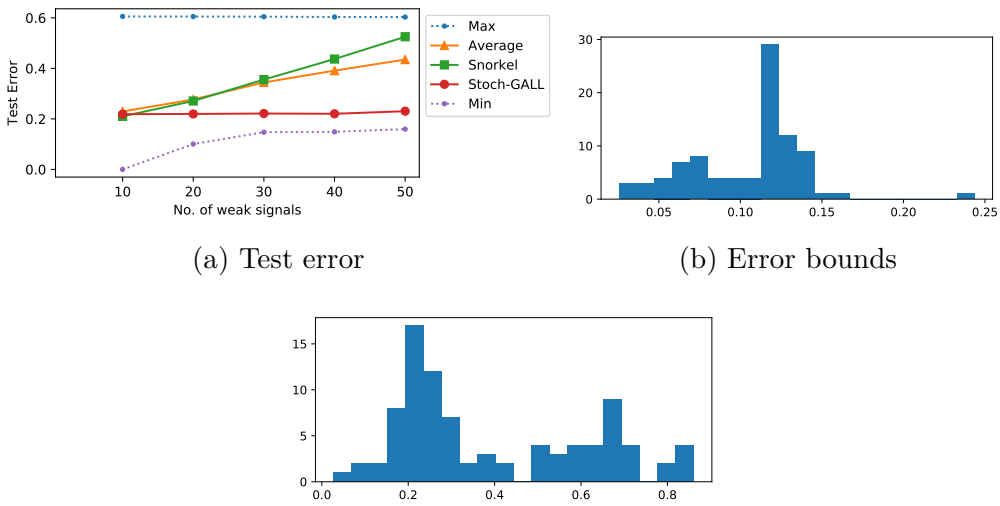
(c) Precision bounds

Figure 3.8: Test error on Fashion-MNIST dataset using pseudolabels and human weak signals.

provided by each human weak signal is already included in the pseudolabeling signal. This type of redundant information is common when using weak supervision. Many signals can have dependencies and redundancies. And despite the Snorkel system's modeling of dependencies among weak signals, it is still confounded by them while Multi-ALL's model-free approach is robust.

Our final experiments test the performance of the different models using pseudolabels and human weak labels on another image classification task. We use the Street View House Numbers (SVHN) [76] dataset, which represents the task of recognizing digits on real images of house numbers taken by Google Street View. Each image is a 32×32 RGB vector. The dataset has 10 classes consisting of 73,257 training images and 26,032 test images.

Figure 3.9 plots the results of the experiment. Figure 3.9a features the same trend as Fig. 3.8a. For this task, the human weak signals perform poorly in labeling the images, so they do not provide additional information to the learners. This fact is evident in the horizontal slope of the max curve. The min curve suggests that the human weak signals are



(a) Test error

(b) Error bounds

(c) Precision bounds

Figure 3.9: Test error on SVHN dataset using pseudolabels and human weak signals.

redundant with poorly estimated bounds, and adding them decreases the space of possible labels for Multi-ALL. Comparing models, Multi-ALL's performance is not affected by the redundancies in the weak signals. Since the human weak signals are very similar, Snorkel seems to mistakenly trust the information from these signals more as we add more of them, thus hurting its model performance. Multi-ALL uses the extra information provided to it as bounds on the weak signals to protect against placing higher emphasis on the redundant human weak signals.

3.6 Discussion

We introduced adversarial label learning (ALL), a method to train robust classifiers when access to labeled training data is limited. ALL trains a model without labeled data by making use of weak supervision to minimize the error rate for adversarial labels, which are subject to constraints defined by the weak supervision. We demonstrated that our method is robust against weak supervision signals that make dependent errors and gets better performance

with better supervision signals. Our experiments confirm that ALL is able to learn models that outperform the weak supervision and baseline models. ALL is also capable of directly training classifiers to mimic the weak supervision.

Subsequently, we introduced Multi-ALL, a generalized adversarial label learning framework that enables users to encode information about the data as a set of linear constraints. We show in our experiments the performance of our method using precision and error constraints. However, Multi-ALL allows for other forms of linear constraints. Our evaluation demonstrates that our adaptive framework is able to generate high-quality labels for a learning task and is also able to combine different sources of weak supervision to increase the performance of a model. Our experiments show that our framework outperforms other weak supervision method on different image-classification tasks and is better at handling redundancies among weak supervision signals. Our work on ALL was published in Association for the Advancement of Artificial Intelligence [77] and a general framework for ALL was published in the Journal for Machine Learning Research [78].

3.6.1 Limitations of ALL

While ALL has several advantages over alternative approaches for weakly supervised learning, it also has some limitations that are important to acknowledge. We discuss these limitations in this section and the open problems associated with them.

Incorrect or loose bounds

ALL trains a model with weak supervision signals and estimated error bounds of the signals. For ALL to learn, we expect the error bounds to be not too far from the true error rates of the weak signals. Using very tight bounds would over-constrain the optimization, thereby

making it infeasible for a solution to be found. Loose bounds—on the other hand—would under-constrain the optimization and make the adversary too powerful. Bounds can be loose even with true error bounds for signals that are only slightly better than random. We have observed this effect in synthetic settings where the features of the data are not informative enough to form informative weak signals. The model training would then contend against arbitrary labels for the training data, causing ALL to fail. Because of the dependence of ALL's performance on the quality of the error bounds, we advise users of ALL to estimate the bounds using a validation set. In cases where a validation set is not available, we recommend using constant bounds or estimating the error bounds of the weak signals using methods that estimate error rates of classifiers without labeled data.

Quality of weak signals

An implicit assumption we make with the weak supervision signals we provide to ALL is that the weak signals are better than random. If this assumption is violated, this information needs to be passed to the learning algorithm by providing accurate estimates of the bound of the weak signals. Additionally, ALL performs better when the weak signals provided are somewhat correlated with the labels of the training data. We see the effect of these assumptions in our first set of experiments (Tables 3.1 and 3.2). We include weak signals whose accuracies are only slightly better than random. Using information from the bound and features of the data, ALL is able to learn high quality models that separate the data. For more complex datasets like image datasets, higher quality weak signals are necessary for the model to learn better. For example, in our work, we used simple nearest-neighbor weak signals provided by human annotators and programmatically generated weak signals. The human-provided weak signals performed well in some experiments (Fashion Mnist), but in other experiments (SVHN), they did not provide adequate information to the learner. One

idea for improving the human signals is by learning latent data representations (e.g., with an autoencoder) and comparing the latent representations, rather than raw pixel values. We surmise that doing this will provide higher quality signal to the learning model.

Noise level

A major motivation for our algorithm is to be robust to noise from redundant weak signals. We showed in Fig. 3.3 experiments that demonstrate ALL's robustness to redundant/noisy signals. However, we have not established what level of noise ALL is robust to. One area of future work is to study the robustness of ALL with relation to random classification noise [79] for binary classification, class-conditional noise [80, 81, 82] for multi-class classification, or instance dependent noise [83, 84]. Recent works [85, 86] have attempted to establish theoretical justifications and connections between weakly supervised learning and robustness to label noise. Our experiments do suggest that ALL is robust to some noise. Its performance reduces with increased noise levels in the weak signals but it does not degrade as much as competing methods in our experiments.

Dependence on Learned Model Parameterization

As we demonstrated in Section 3.3.8 on the RTE dataset, ALL performance can be dependent on the parameterization of the learned model $\boldsymbol{p}(\theta)$. Linear models is adequate for datasets with simple features however when training ALL on complex text or image datasets, non-linear models like neural networks should be preferred to enable $\boldsymbol{p}(\theta)$ learn better parameters for the data.

3.6.2 Comparing Adversarial Modeling with Generative Modeling for Weak Supervision

ALL uses worst-case perturbations to defend against possible correlations among weak signals. However, some existing approaches for weakly supervised learning [34, 56, 73, 87] use generative modeling to model the possible correlations of the weak signals. Unlike these methods, ALL does not depend on the modeling assumptions necessary to do generative modeling, such as on the dependency structure among the weak signals or the parametric form of the distributions. In practice, generative models work well when these modeling assumptions match the real data process. In these cases, generative approaches will provide better results than ALL, since these methods will properly model the uncertainty and correlation among the weak signals. Additionally, generative models have an advantage that they try to estimate the accuracy bounds of the weak signals and then use the estimated accuracies to combine the different weak signals. ALL instead takes in the bounds as an input and thus can be sensitive to incorrect bounds as explained in Section 3.6.1. If the bounds are set appropriately, ALL retains the useful information from the weak signals while also protecting against the possibility of correlated errors. However, if the bounds are too loose, fine-grained structure and information from the weak signals can be lost. Given the differences between ALL and generative models, we recommend using ALL when one is unconfident about generative modeling assumptions and when a reasonable estimate of the bounds of the weak signals can be obtained. In such settings, ALL's approach of protecting against worst-case conditions makes it robust against unknown correlations in weak supervision.

Chapter 4

Constrained Label Learning

4.1 Introduction

In previous chapter, we introduced Adversarial Label Learning and covered some of the limitations of the algorithm. Of particular note is that loose bounds can make the adversary too powerful and cause the algorithm to fail. Additionally, ALL relies on weak supervision that label the whole data examples. In practice, weak signals can abstain on certain data examples.

To fix some of the shortcomings of ALL, we propose *constrained label learning* (CLL), a method that processes various weak supervision signals and combines them to produce high-quality training labels. The idea behind CLL is that, given the weak supervision, we can define a constrained space for the labels of the unlabeled examples. The space will contain the true labels of the data, and any other label sampled from the space should be sufficient to train a model. We construct this space using the expected error of the weak supervision signals, and then we select a random vector from this space to use as training labels. Our analysis shows that, the space of labels considered by CLL improves to be tighter around the true labels as we include more information in the weak signals and that similar to ALL, CLL is not confounded by redundant weak signals.

CLL takes as input (1) a set of unlabeled data examples, (2) multiple weak supervision signals

that label a subset of data and can abstain from labeling the rest, and (3) a corresponding set of expected error rates for the weak supervision signals. While the weak supervision signals can abstain on various examples, we require that the combination of the weak signals have full coverage on the training data. The expected error rates can be estimated if the weak supervision signals have been tested on historical data or a domain expert has knowledge about their performance. In cases where the expected error rates are unavailable, they can be treated as a hyperparameter. Our experiments in Section 4.2.3 show that CLL is still effective when it is trained with a loose estimate of the weak signals. Alternatively, we provide guidelines on how error rates can be estimated.

We implement CLL as a stable, quickly converging, convex optimization over the candidate labels. CLL thus scales much better than many other weak supervision methods. We show in Section 4.3 experiments that compare the performance of CLL to other weak supervision methods. On a synthetic dataset, CLL trained with a constant error rate is only a few percentage points from matching the performance of supervised learning on a test set. On real text and image classification tasks, CLL achieves superior performance over existing weak supervision methods on test data.

4.2 Proposed Method

The goal of *constrained label learning* (CLL) is to return accurate training labels for the data given the weak supervision signals. The estimation of these labels should be aware of the correlation among the weak supervision signals and should not be confounded by it. Toward this goal, we use the weak signals' expected error to define a constrained space of possible labelings for the data. Any vector sampled from this space can then be used as training labels. We consider the setting in which the learner has access to a training

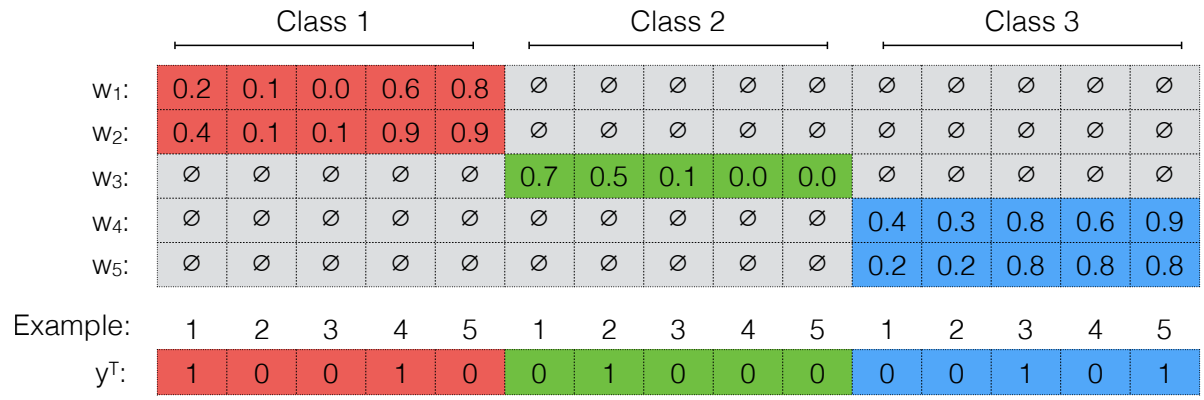


Figure 4.1: Illustration of weak signals and label vectorized structure. For multi-class problems, we arrange the label vector so that it contains indicators for each example belonging to each class. The weak signals use the same indexing scheme. In this illustration, weak signals $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ estimate the probability of each example belonging to class 1 and abstain on estimating membership in all other classes.

set of unlabeled examples, and a set of weak supervision signals from various sources that provide approximate indicators of the target classification for the data. Along with the weak supervision signals, we are provided estimates of the expected error rates of the weak signals. Formally, let the data be $X = [x_1, \ldots, x_n]$. These examples have corresponding labels $\boldsymbol{y} = [y_1, \ldots, y_n] \in \{0, 1\}^n$. For multi-label classification, where each example may be labeled as a member of K classes, we expand the label vector to include an entry for each example-class combination, i.e., $\boldsymbol{y} = [y_{(1,1)}, \ldots, y_{(n,1)}, y_{(1,2)}, \ldots, y_{(n-1,K)}, y_{(n,K)}]$, where y_{ij} is the indicator of whether the ith example is in class j.[1] See Fig. 4.1 for an illustration of this arrangement.

With weak supervision, the training labels $\boldsymbol{y}$ are unavailable. Instead, we have access to m weak supervision signals $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\}$, where each weak signal $\boldsymbol{q} \in [\emptyset, 0, 1]^n$ is represented as a vector of indicators that each example is in each class. The weak signals can choose to abstain on some examples. In that case, they assign a null value $\emptyset$ to that example's entry. In practice, weak signals for multi-class problems typically only label one class at a time,

[1] We represent the labels as a vector for later notational convenience, even though it may be more naturally arranged as a matrix.

such as a one-versus-rest classification rule, so they effectively abstain on all out-of-class entries. The weak signals can be soft labels (probabilities) or hard labels (class assignments) of the data. In conjunction with the weak signals, the learner also receives the expected error rates of the weak signals $\boldsymbol{b} = [b_1, \ldots, b_m]$. In practice, the error rates of the weak signals are estimated or treated as a hyperparameter. The expected empirical error of a weak signal $\boldsymbol{q}_i$ is

$$\begin{aligned} b_i &= \frac{1}{n_i}\left(\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}\boldsymbol{q}_i^\top(1-\boldsymbol{y}_k) + \mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}(1-\boldsymbol{q}_i)^\top\boldsymbol{y}_k\right) \\ &= \frac{1}{n_i}\left(\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}(1-2\boldsymbol{q}_i)^\top\boldsymbol{y}_k + \boldsymbol{q}_i^\top\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}\right), \end{aligned} \tag{4.1}$$

where $\boldsymbol{y}_k$ is the true label for the class k that the weak signal $\boldsymbol{q}_i$ labels, $n_i = \sum \mathbf{1}_{(\boldsymbol{q}_i\neq\emptyset)}$ and $\mathbf{1}_{(\boldsymbol{q}_i\neq\emptyset)}$ is an indicator function that returns 1 on examples the weak signals label (i.e., do not abstain on). Hence, we only calculate the error of the weak signals on the examples they label.

Analogously to Eq. (4.1), we can express the expected error of all weak signals for the label vector as a system of linear equations in the form $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{c}$. To do this, we define each row in $\boldsymbol{A}$ as

$$\boldsymbol{A}_i = \mathbf{1}_{(\boldsymbol{q}_i\neq\emptyset)}(1-2\boldsymbol{q}_i), \tag{4.2}$$

a linear transformation of a weak signal $\boldsymbol{q}$. Each entry in the vector $\boldsymbol{c}$ is the difference between the expected error of the weak signal and the sum of the weak signal, i.e.,

$$\boldsymbol{c_i} = n_i b_i - \boldsymbol{q}_i^\top\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}. \tag{4.3}$$

Valid label vectors then must be in the space

$$\{\tilde{\boldsymbol{y}} | \boldsymbol{A}\tilde{\boldsymbol{y}} = \boldsymbol{c} \wedge \tilde{\boldsymbol{y}} \in [0,1]^n\}\ . \tag{4.4}$$

The true label $\boldsymbol{y}$ is not known. Thus, we want to find training labels $\tilde{\boldsymbol{y}}$ that satisfy the system of linear equations.

4.2.1 Algorithm

Having defined the space of possible labelings for the data given the weak signals, we explain here how we efficiently sample a vector of training labels from the space. First, we initialize a random $\tilde{\boldsymbol{y}}$ from a uniform distribution $\tilde{\boldsymbol{Y}} \sim U(0,1)^n$. Then we minimize a quadratic penalty on violations of the constraints defining the space. The objective function is

$$\min_{\tilde{\boldsymbol{y}} \in [0,1]^n} \quad \|\boldsymbol{A}\tilde{\boldsymbol{y}} - \boldsymbol{c}\|_2^2 \ . \tag{4.5}$$

The solution to this quadratic objective function gives us feasible labels for the training data. In our experiments, we estimate the error rates $\boldsymbol{b}$ of the weak signals. In cases where the error estimates make an infeasible space, this quadratic penalty acts as a squared slack. We solve Eq. (4.5) iteratively using projected Adagrad [71], clipping $\tilde{\boldsymbol{y}}$ values to $[0,1]^n$ between gradient updates. This approach is fast and efficient, even for large datasets. Our algorithm is a simple quadratic convex optimization that converges to a unique optimum for each initialization of $\tilde{\boldsymbol{y}}$. In our experiments, it converges after only a few iterations of gradient descent. We run the algorithm 3 times with random initialization of $\tilde{\boldsymbol{y}}$ and take the mean of the $\tilde{\boldsymbol{y}}$s as the estimated label. We observed that the labels returned from the different runs are very similar. We fix the number of iterations of gradient descent for each run to 200 for all our experiments. The full algorithm is summarized in Algorithm 3.

Algorithm 3 Constrained Label Learning

Require: Weak signals $[\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m]$, and expected error $\boldsymbol{b} = [b_1, \ldots, b_m]$ for the signals.
1: Define $\boldsymbol{A}$ from Eq. (4.2) and $\boldsymbol{c}$ from Eq. (4.3) using the weak signals and expected errors.
2: Initialize $\tilde{\boldsymbol{y}}$ as $\tilde{\boldsymbol{y}} \sim U(0,1)^n$
3: **while** not converged **do**
4: Update $\tilde{\boldsymbol{y}}$ with its gradient from Eq. (4.5)
5: Clip $\tilde{\boldsymbol{y}}$ to $[0,1]^n$
6: **end while**
return estimated labels $\tilde{\boldsymbol{y}}$

4.2.2 Analysis

We start by analyzing the case where we have the true error $\boldsymbol{b}$, in which case the true label vector $\boldsymbol{y}$ for CLL is a solution in the feasible space. Although the true error rates are not available in practice, this ideal setting is the motivating case for the CLL approach. To begin the analysis, consider an extreme case: if $\boldsymbol{A}$ is a square matrix with full rank, then the only valid label $\tilde{\boldsymbol{y}}$ in the space is the true label, $\tilde{\boldsymbol{y}} = \boldsymbol{y}$. Normally, $\boldsymbol{A}$ is usually underdetermined, which means we have more data examples than weak signals. In this case, there are many solutions for $\tilde{\boldsymbol{y}}$, so we can analyze this space to understand how distant any feasible vector is from the vector of all incorrect labels. Since label vectors are constrained to be in the unit box, the farthest possible label vector from the true labels is $(1 - \boldsymbol{y})$. The result of our analysis is the following theorem, which addresses the binary classification case with non-abstaining weak signals.

Theorem 4.1. *For any $\tilde{\boldsymbol{y}} \in [0,1]^n$ such that $\boldsymbol{A}\tilde{\boldsymbol{y}} = \boldsymbol{c}$, its Euclidean distance from the negated label vector $(1 - \boldsymbol{y}) \in \{0,1\}^n$ is bounded below by*

$$||\tilde{\boldsymbol{y}} - (1 - \boldsymbol{y})|| \geq n||\boldsymbol{A}^+(1 - 2\boldsymbol{b})||, \tag{4.6}$$

where $\boldsymbol{A}^+$ is the Moore-Penrose pseudoinverse of $\boldsymbol{A}$.

Proof. We first relax the constrained space by removing the $[0,1]^n$ box constraints. We can

then analyze the projection onto the feasible space:

$$\min_{\tilde{\boldsymbol{y}}} ||(1-\boldsymbol{y}) - \tilde{\boldsymbol{y}}|| \text{ s.t. } \boldsymbol{A}\tilde{\boldsymbol{y}} = \boldsymbol{c}. \tag{4.7}$$

Define a vector $\boldsymbol{z} := \tilde{\boldsymbol{y}} - \boldsymbol{y}$. We can rewrite the distance as

$$\min_{\boldsymbol{z}} ||(1-2\boldsymbol{y}) - \boldsymbol{z}|| \text{ s.t. } \boldsymbol{A}\boldsymbol{z} = 0. \tag{4.8}$$

The minimization is a projection of $(1-2y)$ onto the null space of $\boldsymbol{A}$. Since the null and row spaces of a matrix are complementary, $(1-2\boldsymbol{y})$ decomposes into

$$(1-2\boldsymbol{y}) = \mathbb{P}_{\text{row}}(1-2\boldsymbol{y}) + \mathbb{P}_{\text{null}}(1-2\boldsymbol{y}),$$

where $\mathbb{P}_{\text{row}}$ and $\mathbb{P}_{\text{null}}$ are orthogonal projections into the row and null spaces of $\boldsymbol{A}$, respectively. We can use this decomposition to rewrite the distance of interest:

$$\begin{aligned} &||(1-2\boldsymbol{y}) - \mathbb{P}_{\text{null}}(1-2\boldsymbol{y})|| \\ &= ||(1-2\boldsymbol{y}) - ((1-2\boldsymbol{y}) - \mathbb{P}_{\text{row}}(1-2\boldsymbol{y}))|| \\ &= ||\,\mathbb{P}_{\text{row}}(1-2\boldsymbol{y})||. \end{aligned} \tag{4.9}$$

For any vector $\boldsymbol{v}$, its projection into the row space of matrix $\boldsymbol{A}$ is $\boldsymbol{A}^{+}\boldsymbol{A}\boldsymbol{v}$, where $\boldsymbol{A}^{+}$ is the Moore-Penrose pseudoinverse of $\boldsymbol{A}$. The distance of interest is thus $||\boldsymbol{A}^{+}\boldsymbol{A}(1-2\boldsymbol{y})||$. We can use the definition of $\boldsymbol{A}$ to further simplify. Let $\boldsymbol{W}$ be the matrix of weak signals

$\boldsymbol{W} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m]^\top$. Then the distance is

$$\begin{aligned} &||A^+(1-2\boldsymbol{W})(1-2\boldsymbol{y})|| \\ &= ||A^+((1-2\boldsymbol{W})\vec{1}_n - 2(1-2\boldsymbol{W})\boldsymbol{y})|| \\ &= ||A^+(n - 2\boldsymbol{W}\vec{1}_n - 2\boldsymbol{A}\boldsymbol{y})||. \end{aligned} \tag{4.10}$$

Because $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{c} = n\boldsymbol{b} - \boldsymbol{W}\vec{1}_n$, terms cancel, yielding the bound in the theorem:

$$\begin{aligned} &||A^+(n - 2\boldsymbol{W}\vec{1}_n - 2n\boldsymbol{b} + 2\boldsymbol{W}\vec{1}_n)|| \\ &= ||A^+(n - 2n\boldsymbol{b})|| = n||A^+(1-2\boldsymbol{b})||. \end{aligned} \tag{4.11}$$

□

This bound provides a quantity that is computable in practice. However, to gain an intuition about what factors affect its value, the distance formula can be further analyzed by using the singular-value decomposition (SVD) formula for the pseudoinverse. Consider SVD $\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^\top$. Then $\boldsymbol{A}^+ = \boldsymbol{V}\Sigma^+\boldsymbol{U}^\top$, where the pseudoinverse Σ^+ contains the reciprocal of all nonzero singular values along the diagonal (and zeros elsewhere). The distance simplifies to

$$n||\boldsymbol{V}\Sigma^+\boldsymbol{U}^\top(1-2\boldsymbol{b})|| = n||\Sigma^+\boldsymbol{U}^\top(1-2\boldsymbol{b})||, \tag{4.12}$$

since $\boldsymbol{V}$ is orthonormal. Furthermore, let $\boldsymbol{p} = \boldsymbol{U}^\top(1-2\boldsymbol{b})$, i.e., $\boldsymbol{p}$ is a rotation of the centered error rates of the weak signals with the same norm as $(1-2\boldsymbol{b})$. From this change of variables, we can decompose the distance into

$$n||\Sigma^+\boldsymbol{p}|| = n\sqrt{\sigma_1^2 p_1^2 \ + \ldots + \ \sigma_m^2 p_m^2}, \tag{4.13}$$

where σ_j is the jth singular value of $\boldsymbol{A}^+$.

As this distance grows toward $\sqrt{n}$, the space of possible labelings shrinks toward zero, at which point the only feasible label vectors are close to the true labels $\boldsymbol{y}$. Equation (4.13) indicates that the distance increases roughly as the rank of $\boldsymbol{A}$ increases, in which case the number of non-zero singular values in Σ^+ increases, irrespective of how many actual weak signals are given. Thus, redundancy in the weak supervision does not affect the performance of CLL. The other key factor in the distance is how far from 0.5 the errors $\boldsymbol{b}$ are. These quantities can be interpreted as the diversity and number of the weak signals (corresponding to the rank) and their accuracies (the magnitude of $\boldsymbol{p}$).

Though the analysis is for length-n label vectors, it is straightforwardly extended to multi-label settings with length-(nK). And with careful indexing and tracking of the abstaining indicators, the same form of analysis can apply for abstaining weak signals.

Figure 4.2 shows an empirical validation of Proposition 4.1 on a synthetic experiment. We plot the error of the labels returned by CLL and majority voting as we change the rank of $\boldsymbol{A}$. We use a synthetic data for a binary classification task with 100 randomly generated examples containing 20 binary features. The weak signals are random binary predictions for the labels where each weak signal error rate is calculated using the true labels of the data. We start with 100 redundant weak signals by generating a matrix $\boldsymbol{A}$ whose 100 columns contain copies of the same weak signal, giving it a rank of 1. We then iteratively increase the rank of $\boldsymbol{A}$ by replacing copies of the weak signal with random vectors from the uniform distribution. The error of CLL labels approaches zero as the rank of the matrix increases while the majority vote error does not improve significantly.

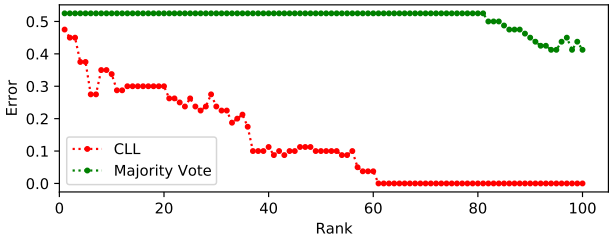


Figure 4.2: Error of CLL estimated labels compared to majority vote as we increase the rank of $\boldsymbol{A}$ by replacing redundant weak signals with linearly independent weak signals.

4.2.3 Error Estimation

In our analysis, we assume that the expected error rates of the weak signals are available. This may be the case if the weak signals have been evaluated on historical data or if an expert provides the error rates. In practice, users typically define weak supervision signals whose error rates are unknown. In this section, we discuss two approaches to handle such situations. We test these estimation techniques on real and synthetic data in our experiments, finding that CLL with these strategies forms a powerful weakly supervised approach.

Agreement Rate Method

Estimating the error rates of binary classifiers using their agreement rates was first proposed by Platanios et al. [5]. They propose two different objective functions for solving the error rates of classifiers using their agreement rates as constraints. Similar to MeTaL [34], we solve a matrix-completion problem to find a low-rank factorization for the weak signal accuracies. We assume that if the weak signals are conditionally independent, we can relate the disagreement rates to the weak signal accuracies. We implemented this method and report its performance in our synthetic experiment (see Section 4.3). The one-vs-all form of the weak signals on our real datasets violates the assumption that each weak signal makes prediction on all the classes, so we cannot use the agreement rate method on our real data.

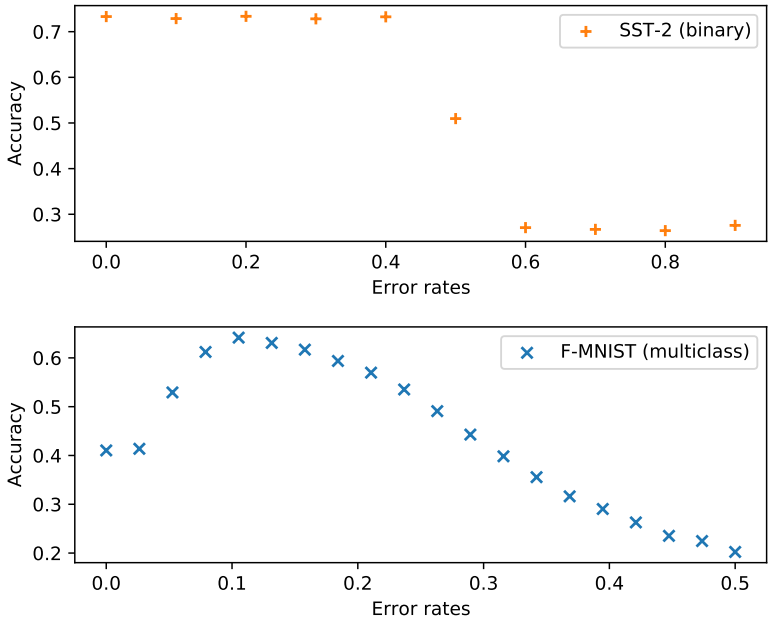


Figure 4.3: Accuracy of constrained label learning as we increase the error rates from 0 to 1 on binary and 0 to 0.5 on multiclass datasets (SST-2 and Fashion-MNIST).

Uniform Error Rate

We showed that ALL can learn as effectively as when using true error bounds by using a fixed constant for the error bounds of all the weak signals. We use this approach in our experiments and extend it to weak supervision signals that abstain and also on multiclass datasets. Figure 4.3 plots the accuracy of generated labels as we increase the error-rate parameter. On the binary-class SST-2 dataset, the label accuracy remains similar if the error rate is set between 0 and 0.5 and drops for values at least 0.5. On the multiclass Fashion-MNIST data, we notice similar behavior where the label accuracies are similar between 0.05 and 0.1 and drop with larger values. We surmise that this behavior mirrors the type of weak supervision signals we use in our experiments. The weak signals in our real experiments are one-vs-all signals; hence a baseline signal (guessing 0 on all examples) will have an error rate of $\frac{1}{K}$. Performance deteriorates when the error rate is worse than this baseline rate.

Method	Test Accuracy
CLL (Agr. rate $\boldsymbol{b}$)	**0.668**± 0.005
CLL (Constant $\boldsymbol{b}$)	0.630± 0.009
Data Programming	0.504± 0.000
Majority Vote	0.504± 0.000
CLL (True $\boldsymbol{b}$)	0.675 ± 0.024
Supervised Learning	0.997± 0.001

Table 4.1: Classification accuracies of the different methods on synthetic data using dependent weak signals. We report the mean and standard deviation over three trials.

4.3 Experiments

We test constrained label learning on a variety of tasks on text and image classification. First, we measure the test accuracy of CLL on a synthetic dataset and compare its performance to that of supervised learning and other baselines. Second, we validate our approach on real datasets.

For all our experiments, we compare CLL to other weakly supervised methods: data programming (DP) [32] and majority-vote (MV) or averaging (AVG). Additionally, on our real datasets we show comparison to regularized minimax conditional entropy for crowdsourcing (MMCE) [30]. For reference, we include the performance of supervised learning baseline. On the image datasets, we show comparison of CLL to Multi-ALL. It is worth noting that DP was developed for binary classification, thus to compare its performance on our multiclass datasets, we run DP on the weak signals that label each class in the datasets. All the weak signals on the real datasets are one-vs-all signals meaning they only label a single class and abstain on other classes.

Method	Test Accuracy
CLL (Agr. rate $\boldsymbol{b}$)	**0.984**± 0.003
CLL (Constant $\boldsymbol{b}$)	0.978± 0.004
Data Programming	0.978± 0.003
Majority Vote	0.925± 0.009
CLL (True $\boldsymbol{b}$)	0.985± 0.0004
Supervised Learning	0.997± 0.001

Table 4.2: Classification accuracies of the different methods on synthetic data using independent weak signals. We report the mean and standard deviation over three trials

Datasets	CLL	MMCE	DP	MV
IMDB	**0.736**± 0.0005	0.573	0.693	0.702
SST-2	**0.678**± 0.0004	0.677	0.666	0.666
YELP-2	0.765± 0.0002	0.685	0.770	**0.775**
TREC-6	0.842± 0.004	0.833	**0.898**	0.273

Table 4.3: Label accuracies of CLL compared to other weak supervision methods on different text classification datasets. We report the mean and standard deviation over three trials. CLL is trained using $\boldsymbol{b} = 0.01$ on the text classification datasets.

4.3.1 Synthetic Experiment

We construct a toy dataset for a binary classification task where the data has 200 randomly generated binary features and 20,000 examples, 16,000 for training and 4,000 for testing. Each feature vector has between 50% to 70% correlation with the true label. We define two scenarios for our synthetic experiments. We run the methods using (1) dependent weak

Datasets	CLL	MMCE	DP	MV	Supervised
IMDB	**0.740**± 0.005	0.551	0.623± 0.007	0.724±0.004	0.820±0.003
SST-2	**0.729**± 0.001	0.727	0.720± 0.001	0.720± 0.0009	0.792± 0.001
YELP-2	**0.840**± 0.0007	0.68	0.760± 0.005	0.798± 0.007	0.879± 0.001
TREC-6	**0.641**± 0.022	0.64	0.627± 0.014	0.605± 0.006	0.700± 0.024

Table 4.4: Test accuracies of CLL compared to other weak supervision methods on different text classification datasets. We report the mean and standard deviation over three trials. CLL is trained using $\boldsymbol{b} = 0.01$ on the text classification datasets

Datasets	CLL	MMCE	DP	AVG	Multi-ALL
SVHN	**0.575**± 0.001	0.1	0.42	0.444	0.196± 0.025
Fashion-MNIST	**0.658**± 0.001	0.147	0.65	0.649	0.488± 0.002

Table 4.5: Label accuracies of CLL compared to other weak supervision methods on image datasets. We report the mean and standard deviation over three trials. CLL is trained using $\boldsymbol{b} = \frac{1}{K}$ on the datasets and it outperforms other baseline approaches.

Datasets	CLL	MMCE	DP	AVG	Multi-ALL	Supervised
SVHN	**0.670**± 0.031	0.1	0.265± 0.004	0.432± 0.001	0.366± 0.003	0.851± 0.002
Fashion-MNIST	**0.695**± 0.002	0.151	0.635± 0.0004	0.666± 0.002	0.598± 0.002	0.852± 0.003

Table 4.6: Test accuracies of CLL compared to other weak supervision methods on image datasets. We report the mean and standard deviation over three trials. CLL is trained using $\boldsymbol{b} = \frac{1}{K}$ on the datasets.

signals and, (2) independent weak signals. In both experiments, we use 10 weak signals that have at most 30% coverage on the data and conflicts on their label assignments. The dependent weak signals were constructed by generating one weak signal that is copied noisily 9 times (randomly flipping 20% of the labels). The original weak signal labeled 30% of the data points and had an accuracy in $[0.5, 0.6]$. So, on average, we expect to perturb 6% of its labels on the copies. The independent weak signals are randomly generated to have accuracies in the range $[0.6, 0.7]$.

We report in Table 4.1 and Table 4.2 the label and test accuracy from running CLL using true error rates for the weak signals, error rates estimated via agreement rate described in Section 4.2.3, and error rates using a maximum error rate constant set to 0.4 as the expected error for all the weak signals. CLL trained using the true $\boldsymbol{b}$ obtains the highest test accuracy compared to the other baselines, and its performance almost matches that of supervised learning in Table 4.2. With the true bounds, CLL slightly outperforms CLL trained using estimated and constant $\boldsymbol{b}$. More interestingly, the results in Table 4.1 show that our method outperforms other baselines that are strongly affected by the dependence

Dataset	No. classes	No. weak signals	Train Size	Test Size	Coverage	Redundancy	Conflict
IMDB	2	10	29,182	20,392	0.174	1.737	0.281
SST-2	2	14	3,998	1,821	0.103	1.445	0.202
Yelp-2	2	14	45,370	10,000	0.177	2.475	0.358
TREC-6	6	30	4,988	500	0.039	1.171	0.821
SVHN	10	50	73,257	26,032	1.0	6.0	-
Fashion-MNIST	10	60	60,000	10,000	1.0	6.0	-

Table 4.7: Summary of datasets and weak signals statistics in our first set of experiments. Coverage is the average number of examples labeled by all the weak signals. Redundancy is the average number of weak signals that label an example in training data. Conflict denotes the fraction of examples covered by conflicting rules in the training data.

in the weak signals. The generative model of data programming assumes that the weak signals are independent given the true labels, but this is not the case in this setup as the weak signals are strongly dependent. Thus the conditional independence violation hurts its performance and essentially reduces it to performing a majority vote on the labels.

Since our evaluation in Fig. 4.3 demonstrated that CLL is not very sensitive to the choice of error rate, we set the error rates $\boldsymbol{b} = 0.01$ on the text datasets and $\boldsymbol{b} = \frac{1}{K}$ on the image datasets. We choose these values because our weak signals in the text dataset tend to label few examples and have low error rates thus we prefer not to under-constrain the optimization by using high error rates values for the one-vs-all weak-signals. In contrast, our human labeled weak signals on the image datasets have high error rates hence we set the error rate value to the baseline value for one-vs-all signals.

4.3.2 Real Experiments

The data sets for our real experiments and their weak signal generation process are described below. Table 4.7 summarizes the key statistics about these datasets. Our code and datasets are provided here.[2]

[2] https://github.com/VTCSML/Constrained-Labeling-for-Weakly-Supervised-Learning

IMDB The IMDB dataset [88] is used for sentiment analysis. The data contains reviews of different movies, and the task is to classify user reviews as either positive or negative in sentiment. We provide weak supervision by measuring mentions of specific words in the movie reviews. We created a set of positive words that weakly indicate positive sentiment and negative words that weakly indicate negative sentiment. We chose these keywords by looking at samples of the reviews and selecting popular words used in them. Many reviews could contain both positive and negative keywords, and in these cases, the weak signals will conflict on their labels. We split the dataset into training and testing subsets, where any example that contains one of our keywords is placed in the training set. Thus, *the test set consists of reviews that are not labeled by any weak signal*, making it important for the weakly supervised learning to generalize beyond the weak signals. The dataset contains 50,000 reviews, of which 29,182 are used for training and 20,392 are test examples.

SST-2 The Stanford Sentiment Treebank (SST-2) is another sentiment analysis dataset [89] containing movie reviews. Like the IMDB dataset, the goal is to classify reviews from users as having either positive or negative sentiment. We use similar keyword-based weak supervision but with different keywords. We use the standard train-test split provided by the original dataset. While the original training data contained 6,920 reviews, our weak signals only cover 3,998 examples. Thus, we used the reduced data size to train our model. We use the full test set of 1,821 reviews.

YELP-2 We used the Yelp review dataset containing user reviews of businesses from the Yelp Dataset Challenge in 2015. Like the IMDB and SST-2 dataset, the goal is to classify reviews from users as having either positive or negative sentiment. We converted the star ratings in the dataset by considering reviews above 3 stars rating as positive and negative otherwise. We used similar weak supervision generating process as in SST-2. We sampled 50,000 reviews for training and 10,000 for testing from the original data set. Our weak signals

only cover 45,370 data points, thus, we used the reduced data size to train our model.

TREC-6 TREC is a question classification dataset consisting of fact-based questions divided into different categories [90]. The task is to classify questions to predict what category the question belongs to. We use the six-class version (TREC-6) from which we use 4,988 examples for training and 500 for testing. The weak supervision we use combines word mentions with other heuristics we defined to analyze patterns of the question and assign a class label based on certain patterns.

SVHN The Street View House Numbers (SVHN) [76] dataset represents the task of recognizing digits on real images of house numbers taken by Google Street View. Each image is a 32×32 RGB vector. The dataset has 10 classes and has 73,257 training images and 26,032 test images. We define 50 weak signals for this dataset. For this image classification dataset, we augment 40 other human-annotated weak signals (four per class) with ten pseudolabel predictions of each class from a model trained on 1% of the training data. The human-annotated weak signals are nearest-neighbor classifiers where a human annotator is asked to mark distinguishing features about an exemplar image belonging to a specific class. We then calculate pairwise Euclidean distances between the pixels in the marked region across images. We convert the Euclidean scores to probabilities (soft labels for the examples) via a logistic transform. Through this process, an annotator is guiding the design of a simple one-versus-rest classifier, where images most similar to the reference image are more likely to belong to its class.

Fashion-MNIST The Fashion-MNIST dataset [60] represents the task of recognizing articles of clothing where each example is a 28×28 grayscale image. The images are categorized into 10 classes of clothing types where each class contains 6,000 training examples and 1,000 test examples. We used the same format of weak supervision signals as in the SVHN dataset (pseudolabels and human-annotated nearest-neighbor classifiers).

Models For the text analysis tasks, we use 300-dimensional GloVe vectors [91] as features for the text classification tasks. Then we train a simple two-layer neural network with 512 hidden units and ReLU activation in its hidden layer. The model for the image classification tasks is a six-layer convolutional neural network model with a 3×3 filter and 32 channels at each layer. We use a sigmoid function as the output layer for both models in our experiment. Thus we train using binary cross-entropy loss with the soft labels returned by CLL, which represent the probability of examples belonging to classes.

Results Tables 4.3 and 4.4 list the performance of the various weakly supervised methods on text classification datasets, while Tables 4.5 and 4.6 list the performance of various weakly supervised methods on image classification datasets. Considering both types of accuracy, CLL is able to output labels for the training data that train high-quality models for the test set. CLL outperforms all competing methods on test accuracy on the datasets. Interestingly, on Yelp and Trec-6 datasets, CLL label accuracy is lower than that of competing baselines yet CLL still achieves superior test accuracy. We surmise that CLL label accuracy is lower than competing methods on some datasets because of the inaccuracy in the error estimates. Generally, CLL is able to learn robust labels from the weak signals, and it seems to pass this information to the learning algorithm to help it generalize on unseen examples. For example, on the IMDB dataset, we used keyword-based weak signals that only occur on the training data. The model trained using CLL labels performs better on the test set than models trained with labels learned from data programming or majority vote. CLL outperforms all competing methods on the image classification tasks. On the digit recognition task (SVHN), CLL outperforms the best compared method (average) by over 13 percentage points for the label accuracy and 23 percentage points on the test data. CLL is able to better synthesize information from the low-quality human-annotated signals combined with the higher-quality pseudolabel signals.

4.4 Discussion

We introduced constrained label learning (CLL), a weakly supervised learning method that combines different weak supervision signals to produce probabilistic training labels for the data. CLL defines a constrained space for the labels of the training data by requiring that the errors of the weak signals agree with the provided error estimates. CLL is fast and converges after a few iterations of gradient descent. Our theoretical analysis shows that the accuracy of our estimated labels increases as we add more linearly independent weak signals. This analysis is consistent with the intuition that the constrained-space interpretation of weak supervision avoids overcounting evidence when multiple redundant weak signals provide the same information, since they are linearly dependent. Our experiments compare CLL against other weak supervision approaches on different text and image classification tasks. The results demonstrate that CLL outperforms these methods on most tasks. Interestingly, we are able to perform well when we train CLL using a worst case uniform error estimate for the weak signals. This shows that CLL is robust and not too sensitive to inaccuracy in the error estimates. Our work on CLL was published at the conference on Uncertainty in Artificial Intelligence [92].

Chapter 5

Data Consistent Weak Supervision

5.1 Introduction

In Chapters 3 and 4, we described two weak supervision algorithms that combine weak signals to produce probabilistic training labels. The algorithms only leverage information from the weak supervision and thus the learned labels may not be consistent with the data. Although the learner in ALL makes use of the data to train the model, the adversary does not explicitly consider the data features in order to estimate adversarial labels. Additionally, ALL and CLL are not able to estimate labels for examples that the weak supervision do not cover.

In this chapter, we propose a novel weak supervision approach that uses input features of the data to aggregate weak signals and produce quality labels for the training data. Our method works by using the features of unlabeled data and the weak signals to define a constrained objective function. We call this paradigm *data-consistency* since the labels produced by the algorithm are a function of the data input features. We define a *label model* that learns parameters to predict labels for the training data, and these labels must satisfy constraints defined by the weak signals. Both the data features and the weak supervision define the space of plausible labelings. Once our algorithm generates data consistent labels, these labels can be used to train any end model as if it were fully supervised.

A major advantage of our approach is that by using features of the data, we are able to estimate labels for examples that have low or no coverage by the weak supervision. In practice, users often define weak signals that do not label some data points. In extreme cases, all the weak signals could abstain on some particular examples. Many existing weak supervision algorithms do not consider these examples for label estimation. Using the features of the dataset enables us to consider fully abstained examples because our algorithm is able to find similarities between covered examples to estimate labels for the abstain examples. Our framework can use various representations for input features and offers flexibility on the choice of parametric model for our label model. We can choose a label model family that best fits each task.

5.2 Proposed Method

The principle behind data consistent weak supervision (DCWS) is that we consider noisy weak supervision labels in conjunction with features of the training data to produce training labels that are consistent with the input data. To do this, we optimize a parametric function constrained by the weak supervision. Formally, let the input data be a set of feature descriptors of examples $X = [x_1, \ldots, x_n]$. These examples have corresponding labels $\boldsymbol{y} = [y_1, \ldots, y_n] \in \{0, 1\}^n$, but they are not available in weakly supervised learning. Instead, we have access to m weak supervision signals $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m\}$, where each weak signal $\boldsymbol{q}_i \in [0, 1]^n$ provides a soft or hard labeling of the data examples. Note that the weak signals can abstain on some examples. In that case, they assign a null value $\emptyset$ to those examples. For multiclass or multi-label classification problems where the labels of the data are a matrix of K classes, each individual weak signal makes one-vs-all predictions for only one class and abstains on other classes. This type of weak supervision is common in practice, since

it is somewhat unnatural for human experts to provide rules that label multiple classes on multi-classification tasks.

The core of DCWS is the following optimization:

$$\begin{aligned} \min_{\theta,\xi\geq 0} \quad & \underbrace{\left\| f_\theta(X) - \hat{Y}_r \right\|_2^2}_{\text{regularization}} + \underbrace{\boldsymbol{C} \sum_{i=1}^{m} \xi_i}_{\text{slack penalty}} \\ \text{s.t.} \quad & \underbrace{\boldsymbol{A} f_\theta(X) \leq \boldsymbol{b} + \boldsymbol{\xi}}_{\text{data consistent constraints}} \end{aligned} \tag{5.1}$$

We describe the components of this optimization the remainder of this section. We first describe the regularization and then we describe how the data consistent constraints are derived.

Regularization The learning objective fits a label model $f_\theta(X)$ to be consistent with weak supervision constraints. We provide a prior labeling $\hat{Y}_r$ that the learning algorithm should default to when given freedom. Choosing a default prior labeling serves as a regularization that can prevent overfitting noisy weak signals or underfitting ambiguous ones. We formulate this regularization as a squared distance penalty

$$\left\| f_\theta(X) - \hat{Y}_r \right\|_2^2. \tag{5.2}$$

In our experiments, we regularize towards majority vote predictions. For comparison, we also show performance of our method in when it trains with uniform regularization and no regularization.

Weak Supervision Bounds We use the same error bound in Eq. (3.2) of ALL extend this to cover multi-class classification tasks where the weak signals can choose to abstain on

the datasets. Our modified bound is

$$\begin{aligned} b_i &\geq \frac{1}{n_i}\left(\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}\boldsymbol{q}_i^\top(\mathbf{1}-\boldsymbol{y}_k)+\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}(\mathbf{1}-\boldsymbol{q}_i)^\top\boldsymbol{y}_k\right) \\ &\geq \frac{1}{n_i}\left(\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}(\mathbf{1}-2\boldsymbol{q}_i)^\top\boldsymbol{y}_k+\boldsymbol{q}_i^\top\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}\right), \end{aligned} \tag{5.3}$$

where $\boldsymbol{y}_k$ is the true label for the class k that the weak signal $\boldsymbol{q}_i$ labels, $n_i = \sum \mathbf{1}_{(\boldsymbol{q}_i\neq\emptyset)}$, and $\mathbf{1}_{(\boldsymbol{q}_i\neq\emptyset)}$ is an indicator function that returns 1 on examples the weak signals do not abstain on. Hence, we only calculate the error of the weak signals on the examples they label.

We make the same transformations in Eqs. (4.2) and (4.3) and express the error bound as a system of linear equations, where $\boldsymbol{A}_i = \mathbf{1}_{(\boldsymbol{q}_i\neq\emptyset)}(\mathbf{1}-2\boldsymbol{q}_i)$ and $\boldsymbol{b} = n_i b_i - \boldsymbol{q}_i^\top\mathbf{1}_{(\boldsymbol{q}\neq\emptyset)}$. From this, we can then rewrite the bound in Eq. (5.3) as

$$\boldsymbol{A}\boldsymbol{y} \leq \boldsymbol{b}. \tag{5.4}$$

Label Model The bounds described so far only consider the weak signals. Using the weak supervision $\boldsymbol{q}$ and user provided bounds $\boldsymbol{b}$, we have constrained the space of possible labeling for the true labels $\boldsymbol{y}$. However, this constrained space can contain label assignments that are unreasonable given the input data. For example, a feasible labeling could give different classifications to two data examples with identical input. To avoid such inconsistencies, we use a parametric model to enforce an additional constraint that the learned labels are consistent with the input features. We estimate $\boldsymbol{y}$ by defining a parametric model $f_\theta(X)$ that reads the data as input and outputs estimated class probabilities. We refer to this model as the *label model*. Our label model is data-consistent by definition because it relates features of the data to the predicted labels. We combine the label model with the weak supervision by finding parameters θ whose predictions of the training labels satisfy the weak supervision constraints. By directly substituting the output of the label model for the estimated labels,

we obtain a constraint on the parameters:

$$\boldsymbol{A} f_\theta(X) \leq \boldsymbol{b}. \tag{5.5}$$

This form of the constraint accommodates many forms of the label model $f_\theta(X)$. Depending on the task, $f_\theta(X)$ could be a linear or non-linear model. This gives our method a flexibility that enables practitioners to adapt it for different problem domains. Lastly, for our model, X could be the training data or any input representation of the data. In Section 5.3, we show experiments with different representations for X.

Slack The constraints are provided by the weak supervision $\boldsymbol{q}$ and the error bounds $\boldsymbol{b}$. If $\boldsymbol{b}$ is too tight, finding solutions to the optimization could be infeasible. In contrast, if $\boldsymbol{b}$ is too loose, then the weak supervision will not adequately constrain the objective and the label estimation will not incorporate information from the weak signals. When the bounds cannot be accurately estimated, we choose a tight bound by setting $\boldsymbol{b} = 0$, and we use a linear slack penalty to adaptively relax the constraints, yielding the constraint $\boldsymbol{A} f_\theta(X) \leq \boldsymbol{b} + \boldsymbol{\xi}$, where $\boldsymbol{\xi}$ is a vector of nonnegative slack variables, and we add to the objective a slack penalty $\boldsymbol{C} \sum_{i=1}^{m} \xi_i$. In this form, setting $\boldsymbol{b} = 0$ becomes equivalent to a weighted majority voting of the weak supervision where the weights are supplied by parameters of the label model.

5.2.1 Optimization

We optimize Eq. (5.1) using Lagrange multipliers. This allows for inexpensive updates to the parameters of the model. At convergence, the Lagrangian function finds a local minimum

that adequately satisfies the constraints. The Lagrangian form of the objective is

$$\begin{aligned} L(\theta, \xi, \boldsymbol{\gamma}) = \left\| f_\theta(X) - \hat{Y}_r \right\|_2^2 + \boldsymbol{C} \sum_{i=1}^{m} \xi_i \\ + \sum_{i=1}^{m} \gamma_i \left(\boldsymbol{A}_i f_\theta(X) - b_i - \xi_i \right), \end{aligned} \tag{5.6}$$

where $\boldsymbol{\gamma}$ is a Karush-Kuhn-Tucker (KKT) multiplier that penalizes violations on the constraints. During training, we use Adam [93] to make gradient updates for the data model $f_\theta(X)$. The Lagrange multiplier $\boldsymbol{\gamma}$ and the slack variable $\boldsymbol{\xi}$ are optimized by gradient ascent and descent respectively. Both the slack variable and the Lagrange multiplier are constrained to be non-negative. This optimization scheme converges when the constraints are satisfied.

5.3 Experiments

Weak supervision algorithms are evaluated on (1) the accuracy of the learned labels on the training data (label accuracy), and (2) the performance of the model on unseen data (test accuracy). Typically, weak supervision algorithms follow a two-stage approach where they first estimate training labels and then use these labels to train an end model that makes predictions on unseen data. DCWS can use a one-stage approach since our label model can itself make predictions on test data, however we use a two-stage approach to measure test accuracy in all our experiments in other to ensure fair comparisons with other methods. To evaluate the effectiveness of DCWS, we design three sets of experiments. First, we train DCWS with the data itself on a synthetic data and measure its label and test accuracy. Secondly, we run DCWS with feature embedding on real data and measure its performance. Lastly, we compare the test accuracy of DCWS to weakly-supervised and semi-supervised methods that train models for label aggregation.

On the first and second set of experiments, we compare the label and test accuracy of DCWS to majority voting, other weak supervision approaches and a crowdsourcing baseline.

Compared Methods The state-of-the-art methods we compare to are FlyingSquid [35], Snorkel MeTaL (MeTaL) [34], regularized minimax conditional entropy for crowdsourcing (MMCE) [30], CLL, and ALL. We show test accuracy on supervised learning as reference. It is worth noting that ALL was developed for binary classification tasks and uses weak signals that do not abstain, hence we only run ALL on datasets that satisfy these requirements. We compare to additional baselines used by Awasthi et al. [1] in our second set of experiments.

Network Architecture and Hyper-parameters In all our runs of DCWS, the label model is a two layer neural network with dropout. The hidden layer uses a ReLU activation function and the output layer is sigmoid for binary classification and softmax on multiclass datasets. We set the slack penalty $\boldsymbol{C} = 10$. We run CLL with the same bounds as ours, $\boldsymbol{b} = 0$, but we run baseline ALL with the true bounds since the constraints in ALL are very sensitive to incorrect bounds.

To test the generalizability of our algorithm on unseen data, we take the labels produced by DCWS and other methods then train an end model to make predictions on a held-out test set. The model is a simple neural network with two 512-dimensional hidden layers and ReLU activation units. We use the Adam optimizer with the default settings for all experiments.

5.3.1 Synthetic Experiment

The aim of this synthetic experiment is to show that DCWS can perform well when the weak signals are noisy and highly dependent. Additionally, we want to show that with additional datapoints that are not covered by the weak supervision, we are able to get performance gains for our method. We construct a synthetic dataset with 32,000 training examples and 8,000

test examples for a binary classification task where the data has 200 randomly generated binary features. Each feature has between 55% to 65% correlation with the true label of the data. We randomly define 10 weak signals where 9 of the signals are close copies of 1 weak signal, 95% overlap between them. That is, one weak signal is copied noisily 9 times by randomly flipping 5% of the labels. The weak signals have 50% coverage on the data and have error rates between $[0.35, 0.45]$. We run experiments in two settings. In the first setting, DCWS and competing baselines train with only datapoints that are covered by at least one weak signal. In the second setting, DCWS trains with all the datapoints in the training data—including data examples that are not covered by any weak supervision. We refer to this setting as DCWS+. Tables 5.1 and 5.2 shows the performance of the different methods in these settings.

As seen in the table, DCWS achieved the highest performance on both label and test accuracies compared to competing baselines. It outperforms the next best performing method by over 16% percentage points on label accuracy and over 13% points on test accuracy. This improvement is significant because FlyingSquid and MeTaL explicitly model the dependency structure between the weak signal and use this information to learn their labels. DCWS does not solve for weak signal dependency but rather uses the features of the training data and the weak signal constraint to defend against possible redundancies. Additionally, we see that DCWS+ obtains better performance than even DCWS, showing that it is able to leverage features from additional examples that are not covered by the weak supervision to better inform the model. Other weak supervision methods typically remove uncovered data examples, but DCWS can synthesize information from these examples to learn more effectively.

Method	Label Accuracy
DCWS	0.790 ± 0.013
FlyingSquid	0.622
MeTaL	0.622
MMCE	0.625
CLL	0.625
Majority Vote	0.625
DCWS+	**0.821** ± 0.012

Table 5.1: Label accuracies of the different methods on synthetic data. DCWS trains with datapoints that are covered by the weak supervision, while DCWS+ trains with additional data examples that have no weak supervision coverage. We report the mean and standard deviation over three trials.

5.3.2 Real Data

In this section, we describe the datasets we use in the first experiments and the weak supervision we provide to the learning algorithms. For fair comparison, we only consider training examples that are covered by at least one weak signal. We used the same datasets and weak supervision in Section 4.3 for the text datasets. The text classification tasks also uses the same 300-dimensional GloVe vectors features as representation of the data.

For Fashion Mnist dataset, we construct a binary classification task by using two classes from the data. The task is to classify the tops vs. trousers. We define weak signals for the data by choosing 2 images from each class. The chosen images are used as reference data to generate the weak signals and are excluded from the training data. We calculate pairwise cosine similarity between the embedding of the reference images and the images in the training data and use the rounded scores as the weak supervision labels. Each reference image provides a weak signal hence we have 4 weak supervision signals in total for the dataset. We use a pre-trained VGG-19 net [94] to extract features of the data. The extracted features are then used to train the model.

Method	Label Accuracy
DCWS	0.843 ± 0.009
FlyingSquid	0.690
MeTaL	0.668
MMCE	0.679
CLL	0.704
Majority Vote	0.633
DCWS+	**0.867** ± 0.007

Table 5.2: Test accuracies of the different methods on synthetic data. DCWS trains with datapoints that are covered by the weak supervision, while DCWS+ trains with additional data examples that have no weak supervision coverage. We report the mean and standard deviation over three trials.

Datasets	DCWS	FlyingSquid	MeTaL	MMCE	CLL	ALL	Majority
IMDB	**0.811**± 0.004	0.736	0.736	0.573	0.737	-	0.701
SST-2	**0.741**± 0.001	0.660	0.672	0.677	0.678	-	0.674
YELP-2	**0.841**± 0.003	0.780	0.772	0.685	0.765	-	0.775
Fashion Mnist (Binary)	**0.976** ± 0.005	0.951	0.951	0.952	0.952	0.952	0.868

Table 5.3: Label accuracies of DCWS compared to other weak supervision methods on different text and image classification datasets. We report the mean and standard deviation over three trials. We do not list the standard deviations if they are less than 0.001.

5.3.3 Results on Real Datasets

Tables 5.3 and 5.4 show the performance of our method and various baselines on text and image classification datasets. From Table 5.3, we see that DCWS consistently outperforms alternative label aggregation approaches on both label and test accuracies. On the IMDB dataset, DCWS outperforms the next best performing method on label accuracy by over 7.5% percentage points and over 6% on the SST and YELP datasets. We see similar results on the test set in Table 5.4. DCWS's performance is even close to that of supervised learning without using any labeled data. The performance of DCWS in these experiments demonstrates the advantage our method gains by considering features of the data.

Datasets	DCWS	FlyingSquid	MeTaL	MMCE	CLL	ALL	Majority	Supervised
IMDB	**0.779**± 0.002	0.745	0.685	0.555	0.685	-	0.716	0.816
SST-2	**0.731**± 0.004	0.681	0.666	0.682	0.686	-	0.672	0.783
YELP-2	**0.842**± 0.004	0.816	0.780	0.68	0.826	-	0.774	0.877
Fashion Mnist (Binary)	**0.970**± 0.010	0.940	0.940	0.938	0.941	0.942	0.892	0.992

Table 5.4: Test accuracies of DCWS compared to other weak supervision methods on different text and image classification datasets. We report the mean and standard deviation over three trials. We do not report the standard deviations if they are less than 0.001.

Dataset	No. classes	No. weak signals	Train Size	Test Size	Coverage	Redundancy	Conflict
YouTube	2	10	1,380	232	0.187	1.866	0.302
Census	2	83	9,912	16,125	0.054	4.520	0.275
MIT-R	9	27	8,836	1,898	0.040	1.076	0.886

Table 5.5: Summary of datasets and weak signals statistics for data consistent method experiments. Coverage is the average number of examples labeled by the weak signals. Redundancy is the average number of weak signals that label an example in training data. Conflict denotes the fraction of examples covered by conflicting rules in the training data.

The data-free approaches—FlyingSquid, MeTaL, MMCE, and CLL—have accuracy scores that are on par or slightly better than majority voting on some of the datasets. The performance of these methods are greatly affected by the low coverage from the weak supervision. As in the results from the synthetic experiments, DCWS is less affected by the low weak supervision coverage. It is able to synthesize information from the features of the data, which enables it to learn better labels for its model. On all experiments, DCWS outperforms FlyingSquid, which is considered the current state-of-the-art for latent variable weak supervision models. On the Fashion MNIST dataset, ALL is trained with the true bounds of the weak supervision and as such provides an unfair comparison to competing methods, yet DCWS is still able to outperform ALL.

Methods	Datasets				
	Question (Accuracy)	MIT-R (F1)	YouTube (Accuracy)	SMS (F1)	Census (Accuracy)
Only-L	72.9± 0.6	73.5± 0.3	90.9± 1.8	89.0± 1.6	79.4± 0.5
L+Umaj	71.5± 1.5	73.5± 0.3	91.7± 1.9	92.5± 1.2	80.3± 0.1
Noise-tolerant [95]	72.4± 1.1	73.5± 0.2	92.6± 1.1	91.9± 1.2	80.4± 0.2
L2R [96]	73.2± 2.1	58.1± 1.0	93.4± 0.5	91.3± 0.8	82.3± 0.3
L+Usnorkel [32]	72.2± 3.0	73.5± 0.2	93.6± 0.7	92.5± 1.5	80.4± 0.4
Snorkel-Noise-tolerant	71.5± 1.6	73.5± 0.3	92.9± 0.7	91.7± 1.5	79.6± 0.5
Posterior Reg. [97]	72.1± 1.0	73.4± 0.4	88.0± 1.9	90.8± 1.5	78.6± 0.5
ImplyLoss [1]	**84.6**± 1.5	74.3± 0.3	94.1± 1.1	93.2± 1.0	81.1± 0.2
DCWS	78.7± 0.7	**75.2**± 0.6	**94.5**± 0.5	**95.0**± 0.0	**82.4**± 0.2

Table 5.6: Comparison of DCWS with the baselines from [1] on five different datasets. We report standard deviation of DCWS after over trials. The methods with the best accuracy and F1 score on the test data are bold.

5.3.4 Comparison to Model Training Methods

In the previous set of experiments, we showed comparison of our method to weak supervision approaches that do not consider the data for label aggregation (with the only exception being ALL). In this section, we will provide experiments that compare DCWS to other data-dependent weak supervision and semi-supervised approaches.

Baselines The methods we compare against are from Awasthi et al. [1]. We follow similar procedures as the authors in setting up each dataset. We compare against the results reported by the authors' in the paper.

Datasets & Weak Supervision

We used the same datasets and weak signals as Awasthi et al. [1]. Below is a brief description of each dataset. Table 5.5 shows key statistics about the datasets.

SMS Spam Classification The SMS dataset [98] is a binary classification dataset where the task is to classify short texts as either spam or not spam.

YouTube Spam Classification Similar to SMS dataset, YouTube dataset [99] is a binary classification dataset where the task is to classify comments on YouTube videos as either spam or not spam.

Census The Census dataset [100] is a UCI binary classification dataset from the 1994 US Census. The task is to classify if a person earns more than $\$50,000$ income or not.

MIT-R MIT-R [101] is a multi-class dataset for which the the task is to classify sentences on restaurant search as one of 9 tokens.

Question The Question/Trec-6 dataset [90] is a multi-class dataset that classifies a question as one of six categories.

We used the same label and end models from our previous experiments and regularize towards majority vote predictions. As in our previous experiments, we run DCWS with $\boldsymbol{b} = 0$ on binary datasets. However, on the multiclass datasets, we calculate $\boldsymbol{b}$ on validation data. Awasthi et al. [1] used the validation data to tune hyper-parameters for their method and the methods they compare against. Table 5.6 lists the evaluation of DCWS along with the baseline metrics computed by [1] (Table 2).

Results From Table 5.6, DCWS outperforms competing baselines on all datasets except for Question Classification. We achieve the second best result on this dataset. On the binary datasets, we are able to outperform competing methods without using the validation data for parameter tuning. Additionally, we did not tune our model for all experiments, we used the setting from previous experiments for consistency. The results of the methods on the datasets are close to that of supervised learning, hence the improvement offered by DCWS is only marginal compared to the next best performing method. The results from these experiments show that DCWS performs as well as semi-supervised methods and weak supervision methods that train models to learn labels.

5.3.5 Ablation Study

Our data consistency weak supervision approach has different components that enable it to achieve higher quality results in our experiments. In this section, we test the different components to by removing one component at a time and keeping other parts constant to measure relative importance. The various tests we run are training DCWS (i) without slack, (ii) with uniform regularization, (iii) without regularization, (iv) without constraints, and (v) without data consistency. Also, we train DCWS by varying the number of cluster labels for the data representation and also varying the slack parameter and model type. Table 5.7 lists the results from our ablation study.

Slack and Regularization The results indicate that running DCWS without regularization or slack causes a slight drop in performance of the method. This is consistent with our intuition about the roles of the both components. Slack helps us to adaptively loosen the bounds, while regularization makes our method more stable. Interestingly, using uniform regularization performs as well as regularizing towards majority vote labels, indicating that we can substitute one for the other depending on the classification task. Additionally, we notice that as we increase the slack penalty $\boldsymbol{C}$, the performance of our method gets relatively worse. This trend suggests that allowing the constraints to be violated with a small penalty leads to better generalization.

Constraints Running DCWS without constraints, i.e, $\min_\theta \quad \left\| f_\theta(X) - \hat{Y}_m \right\|_2^2$ makes use of only one type of data consistency and the majority vote regularization. We see from the results of SST-2 that removing the constraints can significantly reduce the performance of DCWS because the some important information from the weak supervision is not considered.

Data consistency We disabled data consistency by directly solving for the labels of the training data rather than using features of the data to optimize a parametric model. The

resulting equation is

$$\min_{\tilde{Y}} \quad \left\| \tilde{Y} - \hat{Y}_m \right\|_2^2 + \boldsymbol{C} \sum_{i=1}^{m} \xi_i$$
$$\text{s.t.} \quad \boldsymbol{A}\tilde{Y} \leq \boldsymbol{b} + \boldsymbol{\xi} \quad \text{and} \quad \boldsymbol{\xi} \geq 0,$$

where $\tilde{Y}$ is the label vector we directly solve for. From Table 5.7, we see that doing this results in highest performance drop on both datasets. SST-2 achieves a performance that is slightly better than random, while YELP-2 dataset has a 16.8 percentage points drop compared to DCWS. The results emphasizes the need for data consistency in developing label aggregation algorithms.

Representation Using cluster labels as features is another form of data consistent training for label aggregation. We used the same clustering method as described in the synthetic experiments and run DCWS with different numbers of clusters. The results show a significant drop in label accuracies on both datasets. This is in contrast with the superior performance obtained on the synthetic data using cluster labels. This behavior is likely because real data may not be separable or may require a different clustering algorithm to obtain meaningful clusters. Moreover, selecting the appropriate number of clusters adds an additional hyperparameter to the algorithm. We suggest using cluster labels as a representation for DCWS when the data distribution is known and appropriate assumptions can be made for the clustering choice.

Model For all our experiments, DCWS was trained with the same neural network model, however a user can choose a different model architecture depending on their classification task. For example on vision tasks, a user could use a deeper neural network model as $f_\theta(X)$ to get better performance gains. Selecting the best model for each dataset is beyond the scope of our paper so we do no report results of running DCWS on different model architectures. We varied our model slightly by training without dropout and we achieve a

Ablation tests	Datasets	
	SST-2	YELP-2
DCWS	0.741± 0.002	0.841± 0.004
Without slack	0.724± 0.003	0.829± 0.002
Uniform regularization	0.740± 0.004	0.832± 0.002
Without regularization	0.739± 0.001	0.823± 0.004
Without constraints	0.671± 0.001	0.822± 0.001
Without data consistency	0.504± 0.026	0.667± 0.023
Without dropout	0.728± 0.002	0.828± 0.003
With slack ($\boldsymbol{C} = 0.1$)	**0.750**± 0.002	**0.844**± 0.002
With slack ($\boldsymbol{C} = 1$)	0.748± 0.003	0.829± 0.002
With slack ($\boldsymbol{C} = 100$)	0.726± 0.003	0.828± 0.002
DCWS (10 cluster labels)	0.560± 0.003	0.633± 0.000
DCWS (100 cluster labels)	0.619± 0.000	0.709± 0.000
DCWS (200 cluster labels)	0.628± 0.003	0.705± 0.001

Table 5.7: Results of the ablation study on SST-2 and YELP-2 datasets.

drop in performance on both datasets.

5.4 Discussion

We introduced data consistent weak supervision (DCWS), an approach for weakly supervised learning that combines features of the data together with weak supervision signals generate quality labels for the training data. DCWS uses a parametric model and learns parameters that predict the labels of the data. We showed three data representation approaches that can be used in our framework: (i) training with the data itself on the synthetic experiment, (ii) training with embedding of the data, and (iii) training with cluster labels of the training data. Our experiments showed that our data consistency approach significantly outperforms other methods for label aggregation. We also highlight in our experiments that our approach performs well even when the weak signals are very noisy and have low or no coverage. Lastly, we showed in our ablation tests the importance of the different components of our proposed

approach. We find that, while the different components of our framework each contribute improvements, data consistency contributes the most to the performance gains achieved by our method.

Limitations of DCWS Our experiments have shown that DCWS incorporates information from the weak supervision and features of the data to produce quality training labels that are consistent with the data. While our experiments have demonstrated good performance on the datasets we tested, we note some limitations in using our method. DCWS requires a good feature representation of the data to learn meaningful relationships from the data. Using poor features for the data could hurt the performance, because in such settings, consistency with the data is actually bad. DCWS can only work if there is relevant information in the features to relate to the estimated labels. Another limitation is the scalability of our current optimization scheme. We train using gradient descent on the full dataset to accommodate the fact that the constraint is global. This can be computationally intensive for large datasets with very high dimensional features.

Chapter 6

Weakly Supervised Pre-Training with Active Refinement

6.1 Introduction

In previous chapters, we presented three different weak supervision algorithms. We discussed the benefit of each algorithm and also their limitations. In general, a common limitation amongst weak supervision algorithms is that high level of noise in the weak supervision can cause the algorithm to fail and produce low-quality training labels. Furthermore, if the assumptions made by a weak supervision algorithm are violated, the model performance can suffer. For our constraint-based weak supervision algorithms, we assume that for each task, the error estimates provided by the user are correct within a reasonable margin. If this is not the case, our algorithms could have bad performance and generate labels that fail on the task. In this chapter, we address these limitations by improving the performance of weak supervision models using only a few labeled examples. Specifically, we correct mistakes made by the weak supervision algorithm by iteratively considering examples where the weak supervision predictions are uncertain. Our algorithm pretrains a neural network model with labels from weak supervision, it then uses active learning to choose examples to label. The labels for these examples are used to refine the weights of the model. It repeats the process until the labeling budget is exhausted. We call our approach weakly supervised pre-training

with active refinement (WSPAR). Figure 6.1 shows an illustration of our method.

We benchmark WSPAR on six text classification datasets, comparing its performance to several baselines including active learning and weak supervision by itself. Our experiments show that WSPAR improves on weak supervision and active learning and does so using only few labeled examples. Additionally, we discover that WSPAR is able to match performance of supervised learning on some classification tasks. Interestingly, with WSPAR we are able to use dataset acquired from one model (weak supervision label model) and train a different model (end model). Furthermore, WSPAR almost always outperforms a model trained with i.i.d. sampled labeled data. This is significant because the authors in [102] identified that active learning performance do not always transfer between models and may not always yield gains over random sampling. Our empirical findings, show that weak supervision may be the key for overcoming these obstacles in deploying active learning.

6.2 Related Work

WSPAR is made up of three component parts: unsupervised pre-training, weak supervision and active learning. We already discussed in Chapter 2 literature review about weak supervision. Hence, in this section, we will explore related works in unsupervised pre-training and active learning. We will also discuss research on intersection between weak supervision and active learning.

Unsupervised Pre-training Unsupervised pre-training serves as a basic building block to many deep learning tasks [103] and has enabled the recent success in deep language models [69, 104, 105, 106]. Additionally, pre-training makes it possible for models to transfer their knowledge of one task to other different but related tasks [107]. In our work, we use pre-

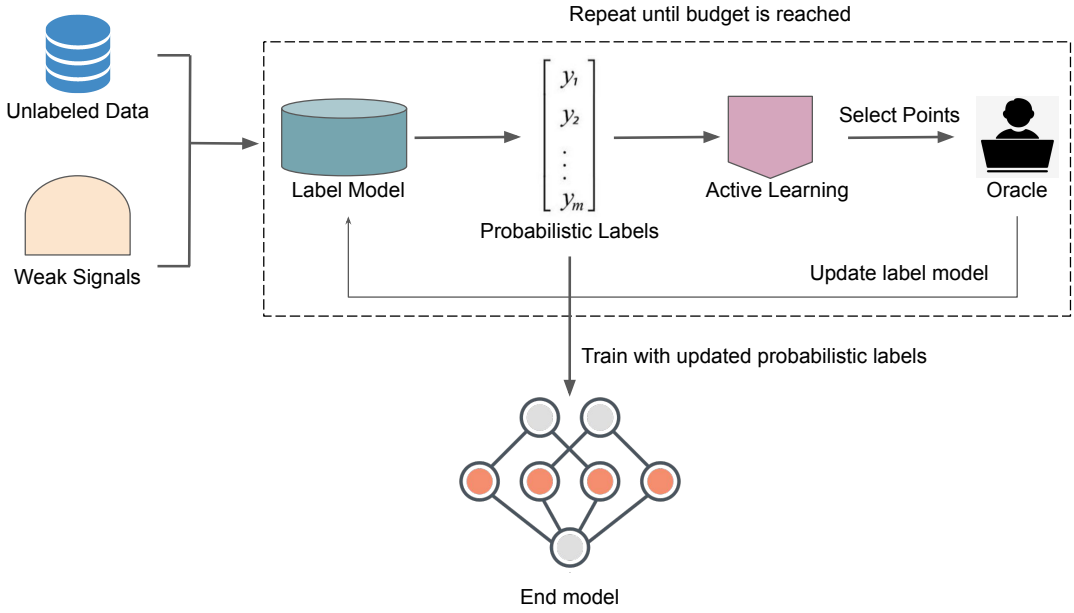


Figure 6.1: Overview of weakly supervised pre-training with active refinement (WSPAR). We pre-train a model with weak supervision and then use active learning to select the most informative examples that improves the model. The refined probabilistic labels are then used to train an end model.

training to initialize the weights of the top layer of our network. Our model is able to gain knowledge from the weak supervision and effectively transfer the gained knowledge in order to improve in the presence of labeled data.

Active Learning Active learning trains a learner by sequentially obtaining labeled data from an oracle labeler. The learner chooses examples to be labeled using a strategy that predicts the most informative examples from unlabeled data. There are various definitions for the most informative data to label, including entropy sampling [108], least-confidence [109], and least-margin [110]. Other approaches apply different query strategies for choosing examples to label [111, 112, 113, 114]. Recent works on active learning have focused on improving its scalability and reducing the number of queries to the oracle [115, 116, 117, 118].

Our work fits this paradigm where we are able to leverage supervision to improve active learning performance thereby improving scalabitlity and ensuring fewer queries to the oracle.

Combination of weak supervision and active learning Combining weak supervision with active learning to increase model performance was first proposed by [119]. They do this by iteratively sampling labels for data points where the weak signals conflict on. They resolve the weak signal conflict by assigning the label as the predictions for all doing the weak signals. Doing this ensures that the weak supervision algorithm predicts the right label for conflicting data points. They implement their method for data programming although the idea can be applied to any weak supervision method. While their method focuses on denoising the noise in the weak signals before it is combined by an label model, our work focuses on improving label model to make better predictions. We showed in our experiments that their methods works well when their is high conflict between the weak signals and the weak signals have high coverage. This is not necessary the case in practice as weak signals typically have low coverage and sometimes conflict on only a handful of examples. The authors recently proposed some modifications to their methods that consider points that the weak signal abstains on [120, 121]. For our method, using a data consistent weak supervision algorithm handles this case. Other methods have also proposed using labeled data or feedback from users to improve labeling functions [122, 123]

A more related approach to is [124]. The authors improve the performance of snorkel MeTal [34] algorithm by modifying the objective function and adding a regularization term that enforces the model parameters to make the right prediction for the data examples for which labels have been sampled. Their work is specific in that it involves modification to the objective function a weak supervision algorithm and thus may not be easy to apply to other weak supervision algorithms. Our method can be applied to any weak supervision algorithm

of choice and is easy to implement.

Some recent works have proposed improving active learning by incorporating weak supervision to its cycle [125, 126]. This is a reverse alternative to what we are doing since we focus on improving performance of weak supervision using active learning. A similarity we share with these works is that our method improves on active learning by itself which is the primary goal of this line of work.

Algorithm 4 Weakly Supervised Pre-Training with Active Refinement

Require: Dataset $X = [x_1, \ldots, x_n]$, weak signals $[\boldsymbol{q}_1, \ldots, \boldsymbol{q}_m]$, oracle labeler $\mathcal{O}$, labeling budget $\mathcal{B}$, sample size z and neural network classifier $f_\theta(X)$.

1: Combine the weak signals using any weak supervision algorithm of choice and generate probabilistic training labels.
2: Pre-train $f_\theta(X)$ using the probabilistic labels.
3: Freeze the weights of the network with exception of the last layer.
4: Initialize empty labeled set $\mathcal{L}$
5: $T \leftarrow \frac{\mathcal{B}}{z}$
6: **for** $i \leftarrow 1$ to T **do**
7: Predict probabilistic labels $\tilde{\boldsymbol{y}}$ for X using $f_\theta(X)$.
8: Use uncertainty sampling to query z examples for $\mathcal{O}$ to label.
9: Add labeled examples to labeled set.
10: Train $f_\theta(X)$ with labeled set updating only the last layer of the model.
11: **end for**
12: Predict new training labels $\tilde{\boldsymbol{y}}$ for X.
return training labels $\tilde{\boldsymbol{y}}$

6.2.1 Proposed Method

The principle behind weakly supervised pre-training with active refinement (WSPAR) is that we improve the performance of weak supervision on classification tasks using labeled examples. Given probabilitic labels from a weak supervision model, we pre-train a neural network classifier $f_\theta(X)$ with the weak supervision labels. After pre-training, we freeze the weights of all the layers of the network with exception of the last layer. We then use active

Method	Weak Supervision	Sampling	Pretraining
WSPAR	DCWS	Uncertainty	✓
Conflict Resolution	DCWS	Conflict	✗
MVPAR	Majority Voting	Uncertainty	✓
MV	Majority Voting	✗	✗
WS	DCWS	✗	✗
Active	✗	Uncertainty	✗
Semi-supervised	✗	Random	✗

Table 6.1: Summary of individual components of WSPAR and baseline methods

learning to sample data points for which the model predictions are uncertain. The sampled labeled examples are used to refine the network continuously until we exhaust the labeling budget. Afterwards, we use $f_\theta(X)$ to generate training labels which can be used to train an end model. The full algorithm is summarized in Algorithm 4.

Our framework accommodates many weak supervision methods. We use DCWS as our base weak supervision algorithm for WSPAR, however we show experiments where we train WSPAR with other weak supervision algorithms.

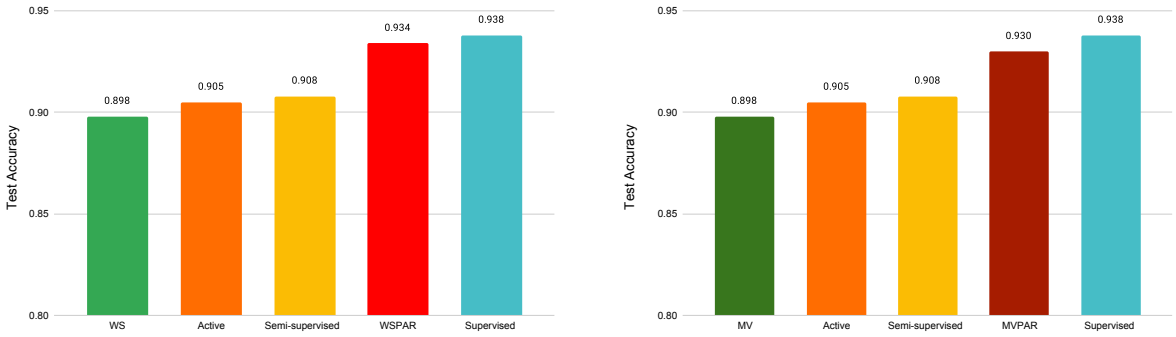
6.3 Experiments

We present a thorough validation of the WSPAR framework together with ablation tests to show performance of each component of the algorithm. We test WSPAR on a variety of tasks for text classification. Like our evaluation in Section 5.3, we use a two-stage approach to separate the label model training from the end model training.

Model For our experiments, our end model is a two-layer neural network with 512 hidden units, a ReLU activation in its hidden layer and a dropout layer. We use the same end model for all the experiments and we use DCWS as our label model.

Datasets	WSPAR	Conflict Resolution	MVPAR	WS	MV	Active	Semi-supervised	Supervised
SST-2	**0.778**± 0.008	0.762± 0.002	0.751± 0.004	0.754± 0.004	0.673± 0.006	0.769± 0.003	0.728± 0.013	0.786± 0.003
IMDB	**0.785**± 0.001	0.745± 0.023	0.768±0.005	0.762±0.010	0.655± 0.039	0.775± 0.002	0.759± 0.012	0.831± 0.002
YELP	**0.861**± 0.001	0.845± 0.002	0.856± 0.004	0.834± 0.006	0.781± 0.001	0.844± 0.003	0.827± 0.019	0.879± 0.001
YouTube	**0.934**± 0.004	0.920± 0.009	0.930± 0.005	0.898± 0.019	0.898± 0.011	0.905± 0.006	0.908± 0.004	0.938± 0.008
Census	**0.832**± 0.002	0.810± 0.002	**0.832**± 0.003	0.809± 0.002	0.796± 0.002	0.818± 0.004	0.800± 0.012	0.838± 0.002
MIT-R	**0.852**± 0.007	0.810± 0.002	0.261± 0.001	0.811± 0.003	0.257± 0.096	0.814± 0.005	0.817± 0.008	0.879± 0.007

Table 6.2: Test accuracies of WSPAR compared to other baseline methods on different text classification tasks. We report the mean and standard deviation over three trials and bold only the best performing method.



(a) WSPAR (b) MVPAR

Figure 6.2: Illustrations showing the benefits of label pre-training with active refinement on YouTube dataset.

Metric To measure performance, we use the test accuracy of the end model evaluated on the test data.

6.3.1 Methods

We compare WSPAR with six representative alternatives. We briefly describe each baseline in details below. Table 6.1 shows the various component of each baseline.

Conflict Resolution We implement the method of [119]. We sample examples where the weak supervision conflicts the most and resolve the conflict by assigning the correct label value to all the weak signals. This way the weak signal predictions agree on the example. We refer to this method as conflict resolution.

Majority Vote Pre-training with Active Refinement (MVPAR) We pre-train the label model with majority vote labels rather than using weak supervision labels.

Majority Voting (MV) We compare to a model trained using only majority voting of the weak supervision labels.

Weak Supervision (WS) We compare to a model trained using just the weak supervision labels.

Active Learning (Active) We compare against an active learning baseline that uses the same uncertainty sampling on binary datasets and entropy sampling on the multi-class dataset. We use 30 seed labeled examples for all the datasets.

Semi-supervised Learning We implement a semi-supervised method *Pseudo-labeling* by training the model with labeled data then making prediction on the unlabeled data [75].

Supervised Learning We also show comparison to supervised learning as reference.

6.3.2 Datasets

We use the same datasets and weak supervision described in Section 4.3.2 and Section 5.3.4. Of the six, five are for binary classification tasks; `SST-2`, `IMDB`, `Yelp`, `YouTube` and `Census`. The last dataset `MIT-R` is a multi-class dataset. We also use the same 300-dimensional GloVe vectors features as representation of the data.

6.3.3 Protocol

We run three sets of experiments. First we compare the performance of our method and the baselines after it trains with the full label data budget and then we evaluate the improvement

of our model as we iteratively increase the labeled data it trains with. Lastly, we run several ablation studies for WSPAR to validate our claims and also test different components of the algorithm. For all sets of experiments, we set $z = 30$, i.e., we sample 30 labeled data points on each iteration of active learning. For the first experiment, we set a labeling budget of 300. That is, we only run 10 iterations of active learning. On all experiments, we use margin sampling as the active learning algorithm on binary classification tasks and on the multi-class dataset we use entropy sampling. For margin sampling, we choose a margin of 0.5 and sample examples whose predictions are close to the margin. This can be thought of as entropy sampling since examples that are closer to 0.5 will have the highest entropy. Our experiments are run over multiple trials and we report the mean and standard deviations of the runs.

6.3.4 Results

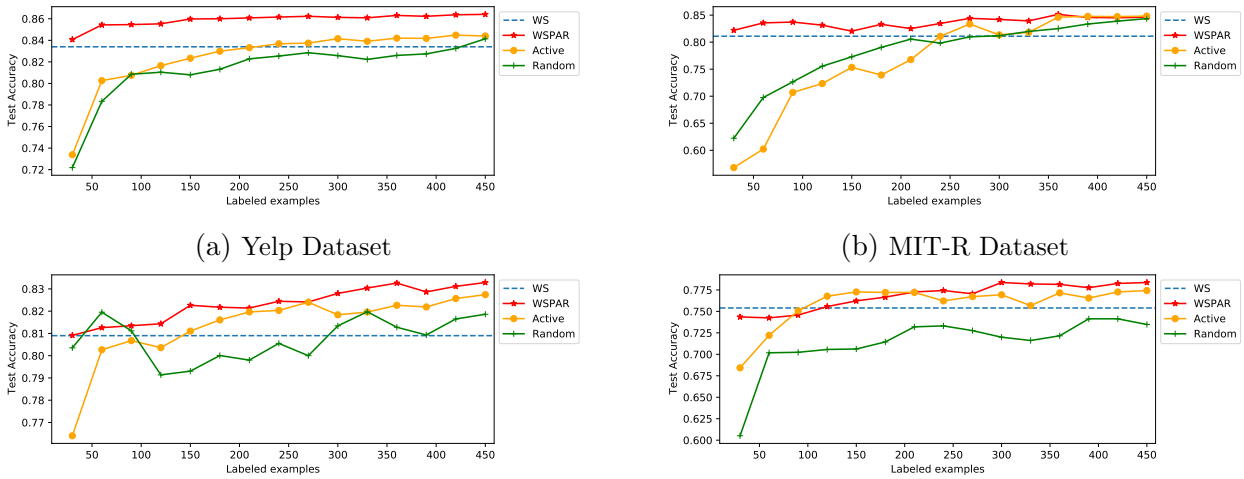
We report in Table 6.2 the performance of WSPAR and other baseline methods on the text classification datasets. From the results, we see that WSPAR outperforms other baselines on all the datasets. On all the experiments, WSPAR shows a significant improvement when compared to weak supervision or active learning by itself. This is especially evident on MIT-R dataset where it improves over the weak supervision and active learning by 4.1 and 3.8 points respectively. Another validation to this claim can be seen from comparing the results of majority voting (MV) by itself to pre-training with majority voting (MVPAR). Across the datasets, we see a huge improvement when active learning is used to refine a model pre-trained with majority voting. On the SST and IMDB datasets, we get improvements of 7.8 points and 11.3 points respectively. Figure 6.2 illustrates this concept on YouTube dataset. We see that both WSPAR an MVPAR improves on the label they are pre-trained with and also outperforms other methods for learning with limited label availability.

Our results also show that active learning performs better than sampling from examples that the weak signals disagree on (conflict resolution). Conflict resolution performs well when there is high conflict amongst the weak supervision but its impact is limited when the weak supervision algorithm has good performance by itself. We surmise that since our method uses a data consistent weak supervision algorithm, it is able to resolve conflicts among the weak supervision, hence sampling labels for data points that the weak signals conflict on do not always improve the performance of the model. This is the case on the IMDB dataset where conflict resolution baseline does not improve the result of weak supervision by itself.

An important observation from our results is that WSPAR is able to close the gap on supervised learning by improving model performance on examples that the label model is uncertain about. We use Fig. 6.2a to show this concept. We see from the plot that the test accuracy of WSPAR is close to supervised learning, performing slightly worse than it by 0.4 points. We observe similar effect on the Census dataset where the performance difference compared to supervised learning differ by 0.6 points. This is very significant since we are able to nearly match supervised learning that trains with 9,912 labeled examples using only 300 labeled points.

Case Study on Accuracy vs. Budget

So far, we have shown and analysed performance of WSPAR after it reaches a fixed labeling budget. In this experiment, we will analyze the performance improvement of WSPAR as we iteratively increase the number of labeled examples we train with. Similar to our previous experiment, we query 30 examples on each iteration but we report the test accuracy of the end model after each iteration. Figure 6.3 shows the performance of WSPAR compared to active learning and random sampling. On the Yelp dataset Fig. 6.3a we see that WSPAR always performs better than weak supervision, active learning and random sampling with as



(a) Yelp Dataset

(b) MIT-R Dataset

(c) Census Dataset

(d) SST-2 Dataset

Figure 6.3: Illustrations showing test accuracy of WSPAR and other baselines as we iteratively increase the number of labeled examples.

little as 50 labeled examples. We observe similar trend on the Census dataset Fig. 6.3c with exception of the low sampling region where i.i.d. random sampling outperforms competing methods with 60 examples. On the MIT-R dataset Fig. 6.3b, WSPAR initially obtains the best performance on low sampling region however as we increase the number of labeled examples the methods attain similar performance on MIT-R dataset. On the SST dataset, WSPAR's performance is slightly better than active learning as we get more data examples. A significant observation is that WSPAR almost consistently surpasses the performance i.i.d. random sampling. Previous studies have shown that a limitation of active learning in practice is that it may not always perform better than random sampling [102]. Our experiments show that by combining weak supervision with active learning, we can overcome this limitation.

6.3.5 Ablation Studies

To test each component of WSPAR, we run three sets of ablation studies: a component study, a weak supervision improvement study and a model transferability study.

Methods	SST-2	IMDB	YELP	YouTube	Census	MIT-R
WSPAR	**0.778**± 0.008	0.785± 0.001	0.861± 0.001	0.934± 0.004	**0.832**± 0.002	0.852± 0.007
WSPAR (no weight freezing)	0.767± 0.009	**0.787**± 0.005	0.852± 0.002	**0.944**± 0.004	0.826± 0.005	0.852± 0.015
WSP RS (random sampling)	0.773± 0.002	0.779± 0.006	0.858± 0.002	0.920± 0.005	0.815± 0.001	**0.856**± 0.003
WSPAR (least confidence)	0.774± 0.002	0.782± 0.015	**0.862**± 0.002	0.934± 0.005	0.829± 0.002	0.843± 0.024
WSP CR (conflict sampling)	0.768± 0.001	0.770± 0.008	0.859± 0.002	0.898± 0.019	0.809± 0.003	0.853± 0.005

Table 6.3: Results of component ablation study on the datasets. We report the mean and standard deviation over three trials and bold only the best performing method.

Component Study The first ablation study tests each component of WSPAR by either removing it or replacing it with a different method in other to measure its relative importance. We test the

1. importance of weight freezing by running WSPAR without freezing any of the weights after pre-training with weak supervision;

2. importance of active learning, by using i.i.d. random sampling to choose labeled examples for WSPAR at each iteration. We refer to this method as WSP RS;

3. benefit of margin/entropy sampling in WSPAR over least confidence sampling;

4. benefit of active learning over conflict sampling in our method. Instead of using active learning to choose training labels, we sample labels for datapoints where the weak supervision conflicts the most. We refer to this as WSP CR.

Table 6.3 shows the results from the study. We observe there is a slight gain in performance on the IMDB and YouTube datasets when the weights of the top layers are not frozen. But the gains are minimal compared to the drop in performance on SST-2 and Yelp dataset from not freezing the weights. Additionally, we observe that using active learning on for our framework outperforms using random sampling for all the datasets with the exception being MIT-R dataset. However, the difference in performance is only marginal. Lastly, margin/en-

Datasets	Models		
	Neural Network	Logistic Regression	Gradient Tree Boosting
SST-2	0.778± 0.008	**0.785**± 0.004	0.765± 0.003
IMDB	**0.785**± 0.001	**0.785**± 0.003	0.764± 0.002
YELP	**0.861**± 0.001	**0.861**± 0.001	0.836± 0.002
YouTube	**0.934**± 0.004	0.911± 0.002	0.912± 0.005
Census	**0.832**± 0.002	0.825± 0.001	0.826± 0.002
MIT-R	**0.852**± 0.007	0.848± 0.009	0.725± 0.014

Table 6.4: Performance of WSPAR on different end models. We report the mean and standard deviation over three trials and bold only the best performing method.

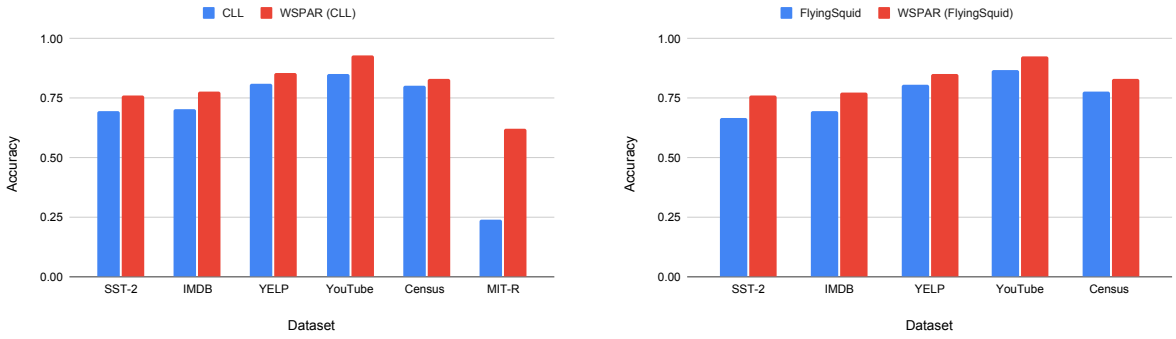


Figure 6.4: Illustrations showing improvement of WSPAR on CLL and FlyingSquid algorithms

tropy sampling has an edge over least confidence sampling for the choice of active learning algorithm and conflict sampling does not improve over active learning on the datasets.

Weak Supervision Improvement Study In this study, we validate our claim that our framework consistently yields gains over the weak supervision. We already observed results in favor of this for DCWS and majority voting in Table 6.2. We test two additional weak supervision algorithms CLL and FlyingSquid [35] to further validate our claim. Figure 6.4 shows the results from the study. We see that on all the datasets, WSPAR (CLL) and WSPAR (FlyingSquid) consistently outperforms their base weak supervision algorithms. On the MIT-R dataset, we run WSPAR (CLL) without freezing the weights of the network

since CLL performance is only slightly better than random on the task and thus its label do not inform the model greatly.

Model Transferability Study Our last ablation study evaluates WSPAR's performance on different end models. In our previous experiments, we validated WSPAR on a neural network end model that is different from the label model from which it was trained. In this experiment, we test WSPAR on a logistic regression model and an ensemble model (gradient tree boosting) to ensure that its performance transfers on different end models. Table 6.4 shows the results from the study. We see that on the different models, WSPAR attains similar performance with exception on the MIT-R dataset. The inferior performance of gradient tree boosting on this dataset is due to the fact that it is an ensemble method and uses a one-vs-rest classification for multiclass classification. Neural network and logistic regression models use softmax and multinomial distributions for multiclass classification making them more ideal for the task. On the IMDB and YELP datasets, logistic regression and neural network models attain identical average performance over three trials but neural network's performance slightly edges that of logistic regression overall on the datasets.

6.4 Discussion

We have introduced weakly supervised pre-training with active refinement (WSPAR), a method for improving the performance of weak supervision models by leveraging labeled data. We showed that by using active learning we can effectively improve the predictive performance of a model trained with weak supervision and we do so with only a few labeled data. Our method is easy and works for any weak supervision algorithm. Additionally, we observed in the experiments that in some cases not freezing the weights of the top layer

achieves better performance than freezing the weights. We recommend weight freezing only when the weak supervision algorithm performs well on that task. A significant observation from our experiments is that WSPAR is able to overcome some of the shortcomings of active learning. It consistently outperforms i.i.d. random sampling and its performance on the label model is able to transfer to different end models. Our performance on the experiments shows that weak supervision may be the key for stable deployment of active learning algorithms in practice.

Chapter 7

Summary and Outlook

7.1 Contributions

In this dissertation, we introduced several algorithms for learning with constraint-based weak supervision. We first introduced Adversarial Label Learning (ALL), an algorithm that estimates probabilistic labels by solving a game between an adversary and a learner. In the binary case, the game is zero-sum but in the multi-class case, we solve a non-zero sum game. In both games, the adversary finds the worst case labeling for the data under the constraint provided by the weak supervision and the learner minimizes the error of the adversarial labels. We showed that ALL is robust to dependent errors in the weak supervision and can train with incorrect estimates of the error rates.

Using ideas from ALL, and in an attempt to solve some of the shortcomings of ALL, we proposed Constrained Label Learning (CLL). CLL is a simple yet effective method that samples a random label from a constrained space and trains a model with the random labeling. We provided theoretical guarantees that showed that any random labeling from the space will have good performance on the task under some conditions. Our theoretical analysis showed that the more diverse and independent the weak signals are, the better the performance of the model.

Next, we introduced the idea of data consistency which is the notion that labels generated by

weak supervision models should be consistent with the data. We presented Data Consistent Weak Supervision (DCWS), a method that takes as input features of the data together with the weak supervision and combines them to produce better training labels for the data. Our experiments showed that DCWS achieved the best performance on most tasks and outperforms other state-of-the-art weak supervision methods.

Lastly, we proposed a framework for improving the general performance of weak supervision models. We introduced Weakly Supervised Pre-Training with Active Refinement (WSPAR), a method that uses active learning to choose labels for data points where the weak supervision predictions are uncertain. We showed that WSPAR outperforms weak supervision and active learning by themselves and overcomes some of the limitations of active learning.

In the next section, we will discuss some of the limitations of our work and prospects for future work.

7.2 Open Problems

Our constraint-based weak supervision algorithms takes as input the error estimate of the weak signals. It uses these error rates or error bounds to constrain the possible labels of the weak supervision. Typically, an expert can provide reliable estimates of the error rates or use different methods to estimate it. However, accurately estimating error rates of weak signals isn't always obtainable in practice. When this happens, the incorrect error rates could cause our algorithms to fail and produce low quality training labels for the data. A possible direction of future work is to theoretically analyze the behavior of our weak supervision algorithms in such settings where the error rates are unreliable. A theoretical analysis will inform how each algorithm will be affected and could suggest how to apply the algorithms when the error rates are incorrectly estimated.

Similarly, we showed empirically that our algorithms are robust to noise in the weak supervision. We did this by running experiments that showed that our methods perform well for different ranges of error rates. A more grounded approach and a possible direction for future research is to establish theoretical conditions of noise levels for which the algorithms are robust to. Each of our weak supervision algorithms have slight modifications to their constraints thus a theoretical analysis can inform the noise level that each algorithm is robust to. Additionally, theoretical understanding could suggest new approaches that are even more robust.

Another possible direction for future work is to develop a theoretical framework for convergence analysis of our algorithms. This will be especially useful for ALL where the optimization is a min-max game for a non-convex objective function. A convergence analysis of our algorithms will provide information on how long to run the algorithms and also what hyper-parameter settings are required for the convergence of the algorithms.

Lastly, weakly supervised pre-training with active refinement showed that we can improve upon weak supervision and active learning using only a few labeled data examples. Our ablation tests showed that while our approach works, its performance can be surpassed on some tasks with slight modifications to our algorithm. An interesting prospect for future work in this line of research will be to investigate other approaches for combining weak supervision with active learning. Our results show a lot of promise and opportunities for applying this line of research.

Bibliography

[1] Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. Learning from rules generalizing labeled exemplars. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeuexBtDr.

[2] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[3] Ariel Jaffe, Ethan Fetaya, Boaz Nadler, Tingting Jiang, and Yuval Kluger. Unsupervised ensemble learning with dependent classifiers. In *Artificial Intelligence and Statistics*, pages 351–360, 2016.

[4] Omid Madani, David M Pennock, and Gary W Flake. Co-validation: Using model disagreement on unlabeled data to validate classification algorithms. In *Advances in Neural Information Processing Systems*, pages 873–880, 2005.

[5] Emmanouil Antonios Platanios, Avrim Blum, and Tom Mitchell. Estimating accuracy from unlabeled data. In *Proceedings of the Thirtieth Conf. on Uncertainty in Artificial Intelligence*, pages 682–691, 2014.

[6] Emmanouil Antonios Platanios, Avinava Dubey, and Tom Mitchell. Estimating accuracy from unlabeled data: A bayesian approach. In *International Conference on Machine Learning*, pages 1416–1425, 2016.

[7] Jacob Steinhardt and Percy S Liang. Unsupervised risk estimation using only con-

ditional independence structure. In *Adv. in Neural Information Processing Systems*, pages 3657–3665, 2016.

[8] Kuzman Ganchev, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul): 2001–2049, 2010.

[9] Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual Intl. ACM SIGIR Conf. on Research and Dev. in Information Retrieval*, pages 595–602, 2008.

[10] Gideon S Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research*, 11:955–984, 2010.

[11] Gideon S Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. *Proceedings of ACL-08: HLT*, pages 870–878, 2008.

[12] Andrew McCallum, Gideon Mann, and Gregory Druck. Generalized expectation criteria. *Computer science technical note, University of Massachusetts, Amherst, MA*, 94 (95):159, 2007.

[13] Daniel Lowd and Christopher Meek. Adversarial learning. In *Proceedings of the ACM SIGKDD Intl. Conf. on Knowledge Discovery in Data Mining*, pages 641–647. ACM, 2005.

[14] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial

training for free! In *Advances in Neural Information Processing Systems*, pages 3353–3364, 2019.

[15] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[16] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2107–2116, 2017.

[17] Mohamad Ali Torkamani and Daniel Lowd. Convex adversarial collective classification. In *Intl. Conf. on Machine Learning*, 2013.

[18] Mohamad Ali Torkamani and Daniel Lowd. On robustness and regularization of structural support vector machines. In *Intl. Conf. on Machine Learning*, pages 577–585, 2014.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[20] Gauthier Gidel, Reyhane Askari Hemmat, Mohammad Pezeshki, Remi Lepriol, Gabriel Huang, Simon Lacoste-Julien, and Ioannis Mitliagkas. Negative momentum for improved game dynamics. *arXiv preprint arXiv:1807.04740*, 2018.

[21] Guojun Zhang and Yaoliang Yu. Convergence of gradient methods on bilinear zero-sum games. *arXiv preprint arXiv:1908.05699*, 2020.

[22] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, pages 20–28, 1979.

[23] Peter Welinder, Steve Branson, Pietro Perona, and Serge Belongie. The multidimensional wisdom of crowds. *Advances in Neural Information Processing Systems*, 23: 2424–2432, 2010.

[24] Bob Carpenter. Multilevel bayesian models of categorical data annotation. *Unpublished manuscript*, 17(122):45–50, 2008.

[25] Huiji Gao, Geoffrey Barbier, and Rebecca Goolsby. Harnessing the crowdsourcing power of social media for disaster relief. *IEEE Intelligent Systems*, 26(3):10–14, 2011.

[26] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in Neural Information Processing Systems*, pages 1953–1961, 2011.

[27] Ashish Khetan, Zachary C Lipton, and Anima Anandkumar. Learning from noisy singly-labeled data. *arXiv preprint arXiv:1712.04577*, 2017.

[28] Qiang Liu, Jian Peng, and Alexander T Ihler. Variational inference for crowdsourcing. In *Advances in neural information processing systems*, pages 692–700, 2012.

[29] Emmanouil Antonios Platanios, Maruan Al-Shedivat, Eric Xing, and Tom Mitchell. Learning from imperfect annotations. *arXiv preprint arXiv:2004.03473*, 2020.

[30] Dengyong Zhou, Qiang Liu, John C Platt, Christopher Meek, and Nihar B Shah. Regularized minimax conditional entropy for crowdsourcing. *arXiv preprint arXiv:1503.07240*, 2015.

[31] Yao Zhou and Jingrui He. Crowdsourcing via tensor augmentation and completion. In *IJCAI*, pages 2435–2441, 2016.

[32] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Info. Proc. Sys.*, pages 3567–3575, 2016.

[33] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, and Houman Alborzi. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In *Intl. Conf. on Manag. of Data*, pages 362–375, 2019.

[34] Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771, 2019.

[35] Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*, pages 3280–3291. PMLR, 2020.

[36] Mayee F Chen, Daniel Y Fu, Frederic Sala, Sen Wu, Ravi Teja Mullapudi, Fait Poms, Kayvon Fatahalian, and Christopher Ré. Train and you'll miss it: Interactive model iteration with weak supervision and pre-trained embeddings. *arXiv preprint arXiv:2006.15168*, 2020.

[37] Mayee Chen, Benjamin Cohen-Wang, Stephen Mussmann, Frederic Sala, and Christopher Ré. Comparing the value of labeled and unlabeled data in method-of-moments latent variable estimation. In *International Conference on Artificial Intelligence and Statistics*, pages 3286–3294. PMLR, 2021.

[38] Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexan-

der Ratner. Wrench: A comprehensive benchmark for weak supervision. *arXiv preprint arXiv:2109.11377*, 2021.

[39] Zhaobin Kuang, Chidubem Arachie, Bangyong Liang, Pradyumna Narayana, Giulia DeSalvo, Michael Quinn, Bert Huang, Geoffrey Downs, and Yang Yang. Firebolt: Weak supervision under weaker assumptions. In *International Conference on Artificial Intelligence and Statistics*, page to appear. PMLR, 2022.

[40] Akshay Balsubramani and Yoav Freund. Scalable semi-supervised aggregation of classifiers. In *Advances in Neural Information Processing Systems*, pages 1351–1359, 2015.

[41] Akshay Balsubramani and Yoav Freund. Optimally combining classifiers using unlabeled data. *arXiv preprint arXiv:1503.01811*, 2015.

[42] A. Mazzetto, C. Cousins, D. Sam, S. H. Bach, and E. Upfal. Adversarial multiclass learning under weak supervision with performance guarantees. In *International Conference on Machine Learning (ICML)*, 2021.

[43] Alessio Mazzetto, Dylan Sam, Andrew Park, Eli Upfal, and Stephen Bach. Semi-supervised aggregation of dependent weak supervision sources with performance guarantees. In *International Conference on Artificial Intelligence and Statistics*, pages 3196–3204. PMLR, 2021.

[44] Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. Self-training with weak supervision. *arXiv preprint arXiv:2104.05514*, 2021.

[45] Daiki Tanaka, Daiki Ikami, Toshihiko Yamasaki, and Kiyoharu Aizawa. Joint optimization framework for learning with noisy labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5552–5560, 2018.

[46] Songzhu Zheng, Pengxiang Wu, Aman Goswami, Mayank Goswami, Dimitris Metaxas, and Chao Chen. Error-bounded correction of noisy labels. In *International Conference on Machine Learning*, pages 11447–11457. PMLR, 2020.

[47] Robert E Schapire, Marie Rochery, Mazin Rahim, and Narendra Gupta. Incorporating prior knowledge into boosting. In *Intl. Conf. on Machine Learning*, volume 2, pages 538–545, 2002.

[48] Razvan C. Bunescu and Raymond Mooney. Learning to extract relations from the web using minimal supervision. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, volume 45, pages 576–583, 2007.

[49] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proc. of the Annual Meeting of the Assoc. for Comp. Ling.*, 2009.

[50] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Joint Euro. Conf. on Mach. Learn. and Knowledge Disc. in Databases*, 2010.

[51] Limin Yao, Sebastian Riedel, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 1013–1023, 2010.

[52] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proc. of the Annual Meeting of the Assoc. for Comp. Linguistics: Human Language Tech.*, pages 541–550, 2011.

[53] Liang-Chieh Chen, Sanja Fidler, Alan L. Yuille, and Raquel Urtasun. Beat the mturkers: Automatic image labeling from weak 3d supervision. In *Proc. of the IEEE Conf. on Comp. Vis. and Pattern Recognition*, pages 3198–3205, 2014.

[54] Jia Xu, Alexander G. Schwing, and Raquel Urtasun. Tell me what you see and I will show you where it is. In *Proc. of the IEEE Conf. on Computer Vis. and Pattern Recog.*, pages 3190–3197, 2014.

[55] Jason A Fries, Paroma Varma, Vincent S Chen, Ke Xiao, Heliodoro Tejeda, Priyanka Saha, Jared Dunnmon, Henry Chubb, Shiraz Maskatia, Madalina Fiterau, et al. Weakly supervised classification of aortic valve malformations using unlabeled cardiac mri sequences. *Nature communications*, 10(1):1–10, 2019.

[56] Yoni Halpern, Steven Horng, and David Sontag. Clinical tagging with joint probabilistic models. In *Proceedings of the Conference on Machine Learning for Healthcare*, pages 209–225, 2016.

[57] Zhaobin Kuang, Frederic Sala, Nimit Sohoni, Sen Wu, Aldo Córdova-Palomera, Jared Dunnmon, James Priest, and Christopher Ré. Ivy: Instrumental variable synthesis for causal inference. In *International Conference on Artificial Intelligence and Statistics*, pages 398–410. PMLR, 2020.

[58] Magnus R Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

[59] Simon S Du and Wei Hu. Linear convergence of the primal-dual gradient method for convex-concave saddle point problems without strong convexity. *arXiv preprint arXiv:1802.01504*, 2018.

[60] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[61] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL http://www.ics.uci.edu/mlearn/MLRepository.html.Irvine.

[62] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical Image Processing and Biomedical Visualization*, volume 1905, pages 861–871. Intl. Society for Optics and Photonics, 1993.

[63] Adel Rajab, Chin-Tser Huang, Mohammed Al-Shargabi, and Jorge Cobb. Countering burst header packet flooding attack in optical burst switching network. In *Intl. Conf. on Information Security Practice and Experience*, pages 315–329. Springer, 2016.

[64] Diogo Ayres-de Campos, Joao Bernardes, Antonio Garrido, Joaquim Marques-de Sa, and Luis Pereira-Leite. Sisporto 2.0: a program for automated analysis of cardiotocograms. *Journal of Maternal-Fetal Medicine*, 9(5):311–318, 2000.

[65] Mehmet Vurkaç. Clave-direction analysis: A new arena for educational and creative applications of music technology. *Journal of Music, Technology & Education*, 4(1): 27–46, 2011.

[66] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. An assessment of features related to phishing websites using an automated technique. In *Internet Technology And Secured Transactions, 2012 Intl. Conf. for*, pages 492–497. IEEE, 2012.

[67] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[68] Rion Snow, Brendan O'connor, Dan Jurafsky, and Andrew Y Ng. Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, 2008.

[69] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[70] 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.

[71] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

[72] Mathieu Blondel, Akinori Fujino, and Naonori Ueda. Large-scale multiclass support vector machine training via Euclidean projection onto the simplex. In *Intl. Conf. on Pattern Recognition*, pages 1289–1294. IEEE, 2014.

[73] Alexander J Ratner, Stephen H Bach, Henry R Ehrenberg, and Chris Ré. Snorkel: Fast training set generation for information extraction. In *Proceedings of the 2017 ACM Intl. Conf. on Management of Data*, pages 1683–1686. ACM, 2017.

[74] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *Proceedings of the IEEE Intl. Conf. on computer vision*, pages 2641–2649, 2015.

[75] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, 2013.

[76] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and A Ng. The street view house numbers (SVHN) dataset. Technical report, Accessed 2016-08-01.[Online]., 2018.

[77] Chidubem Arachie and Bert Huang. Adversarial label learning. In *Proc. of the AAAI Conf. on Artif. Intelligence*, pages 3183–3190, 2019.

[78] Chidubem Arachie and Bert Huang. A general framework for adversarial label learning. *Journal of Machine Learning Research*, 22(118):1–33, 2021.

[79] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2 (4):343–370, 1988.

[80] Guillaume Stempfel and Liva Ralaivola. Learning svms from sloppily labeled data. In *International conference on artificial neural networks*, pages 884–893. Springer, 2009.

[81] Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep K. Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems*, pages 1196–1204, 2013.

[82] Clayton Scott, Gilles Blanchard, and Gregory Handy. Classification with asymmetric label noise: Consistency and maximal denoising. In *Conference On Learning Theory*, pages 489–511, 2013.

[83] Naresh Manwani and PS Sastry. Noise tolerance under risk minimization. *IEEE Transactions on Cybernetics*, 43(3):1146–1151, 2013.

[84] Aritra Ghosh, Naresh Manwani, and PS Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

[85] Giorgio Patrini, Frank Nielsen, Richard Nock, and Marcello Carioni. Loss factorization, weakly supervised learning and label noise robustness. In *International Conference on Machine Learning*, pages 708–717, 2016.

[86] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen 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*, pages 1944–1952, 2017.

[87] Paroma Varma, Frederic Sala, Ann He, Alexander Ratner, and Christopher Ré. Learning dependency structures for weak supervision models. *arXiv preprint arXiv:1903.05844*, 2019.

[88] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[89] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[90] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[91] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014.

[92] Chidubem Arachie and Bert Huang. Constrained labeling for weakly supervised learning. In *Uncertainty in Artificial Intelligence*, pages 236–246. PMLR, 2021.

[93] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[94] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[95] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018.

[96] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343. PMLR, 2018.

[97] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

[98] Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262, 2011.

[99] Túlio C Alberto, Johannes V Lochter, and Tiago A Almeida. Tubespam: Comment spam filtering on youtube. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 138–143. IEEE, 2015.

[100] Dheeru Dua and Casey Graff. Uci machine learning repository, 2017. *URL http://archive. ics. uci. edu/ml*, 7(1), 2019.

[101] Jingjing Liu, Panupong Pasupat, Yining Wang, Scott Cyphers, and Jim Glass. Query understanding enhanced by hierarchical parsing structures. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 72–77. IEEE, 2013.

[102] David Lowell, Zachary C Lipton, and Byron C Wallace. Practical obstacles to deploying active learning. *arXiv preprint arXiv:1807.04801*, 2018.

[103] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[104] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[105] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[106] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[107] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

[108] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

[109] Mingkun Li and Ishwar K Sethi. Confidence-based active learning. *IEEE transactions on pattern analysis and machine intelligence*, 28(8):1251–1261, 2006.

[110] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer, 2007.

[111] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 208–215, 2008.

[112] Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. *Advances in Neural Information Processing Systems*, 23:892–900, 2010.

[113] Ruth Urner, Sharon Wulff, and Shai Ben-David. Plal: Cluster-based active learning. In *Conference on Learning Theory*, pages 376–397. PMLR, 2013.

[114] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE, 2009.

[115] Gordon V Cormack and Maura R Grossman. Scalability of continuous active learning for reliable high-recall text classification. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 1039–1048, 2016.

[116] Er-Chen Huang, Hsing-Kuo Pao, and Yuh-Jye Lee. Big active learning. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 94–101. IEEE, 2017.

[117] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

[118] Meng Fang, Yuan Li, and Trevor Cohn. Learning how to active learn: A deep reinforcement learning approach. *arXiv preprint arXiv:1708.02383*, 2017.

[119] Mona Nashaat, Aindrila Ghosh, James Miller, Shaikh Quader, Chad Marston, and Jean-Francois Puget. Hybridization of active learning and data programming for labeling large industrial datasets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 46–55. IEEE, 2018.

[120] Mona Nashaat, Aindrila Ghosh, James Miller, and Shaikh Quader. Asterisk: Generating large training datasets with automatic active supervision. *ACM Transactions on Data Science*, 1(2):1–25, 2020.

[121] Mona Nashaat, Aindrila Ghosh, James Miller, and Shaikh Quader. Wesal: Applying active supervision to find high-quality labels at industrial scale. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*, 2020.

[122] Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. Interactive weak supervision: Learning useful heuristics for data labeling. *arXiv preprint arXiv:2012.06046*, 2020.

[123] Benjamin Cohen-Wang, Stephen Mussmann, Alex Ratner, and Chris Ré. Interactive programmatic labeling for weak supervision. In *Proc. KDD DCCL Workshop*, 2019.

[124] Samantha Biegel, Rafah El-Khatib, Luiz Otavio Vilas Boas Oliveira, Max Baak, and Nanne Aben. Active weasul: Improving weak supervision with active learning. *arXiv preprint arXiv:2104.14847*, 2021.

[125] Clemens-Alexander Brust, Christoph Käding, and Joachim Denzler. Active and incremental learning with weak supervision. *KI-Künstliche Intelligenz*, 34(2):165–180, 2020.

[126] Julius Gonsior, Maik Thiele, and Wolfgang Lehner. Weakal: Combining active learning and weak supervision. In *International Conference on Discovery Science*, pages 34–49. Springer, 2020.