Socratic Learning

Paroma Varma*[1], Rose Yu[2], Dan Iter[1], Christopher De Sa[1], and Christopher Ré[1]

[1]Department of Computer Science, Stanford University
[2]Department of Computer Science, University of Southern California

Abstract

Modern machine learning techniques often use discriminative models that require large amounts of labeled data. Since generating labeled training data sets is expensive, an alternative approach is to use a generative model, which leverages a simple heuristic to *weakly* label data. Domain experts prefer using generative models because they "tell a story" about their data. Unfortunately, generative models are typically less accurate than discriminative models. Several recent approaches connect the two types of models to exploit their strengths. In this setting, a misspecified generative model can hurt the performance of subsequent discriminative training. To address this issue, we propose a framework called *Socratic learning* that automatically uses information from the discriminative model to correct generative model misspecification. This process also provides users with interpretable feedback about how to improve their generative model.

1 Introduction

For many machine learning applications, large amounts of training data are difficult to obtain. In such cases, distant supervision approaches have been used to generate a *weakly* labeled training set, where a generative model in the form of a simple heuristic or external knowledge base is used to apply labels to data points. The recently proposed data programming [2] paradigm extends this idea by incorporating user-defined heuristics called *labeling functions* in the generative model. This allows for the programmatic labeling of data to create a large but noisy training set.

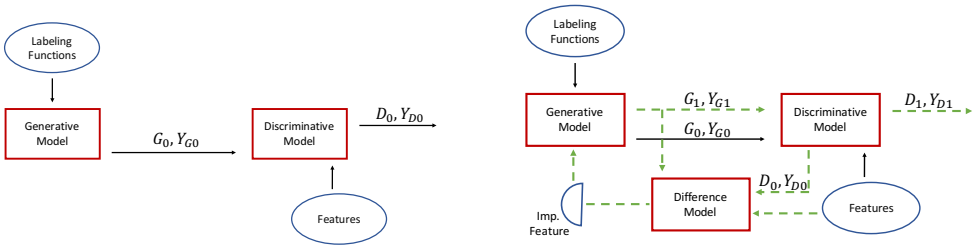
A related framework, the Generative Adversarial Network (GAN) [1] proposes a novel strategy for estimating generative models by simultaneously training two models. In contrast, we address the model misspecification issue of existing frameworks. Users often prefer generative models since they are able to encode domain knowledge through easily explainable rules while discriminative models are more powerful since they have access to the features of the data. Therefore, using the noisy labels from the generative model to train a discriminative model for the desired task takes advantage of the strengths of both kinds of models.

There are two underlying issues with the above approach. First, the generative model being misspecified can affect subsequent discriminative training. Second, the errors in the generative model provide users with little intuition about what went wrong and how it can be fixed. In both cases, there is no methodical approach for identifying and "debugging" the model misspecification issue.

In this paper, we propose the framework of *Socratic learning*, an iterative process that *systematically* improves the generative model by using information from the discriminative model. Since the discriminative model is more powerful than the generative model, it can pass critical knowledge to the generative model to make it more precise and expressive when assigning labels to the training data. This information transfer is performed via features that are easily interpretable and provide

*The authors contributed equally to this work

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.



(a) Data programming pipeline

(b) Socratic learning pipeline

Figure 1: The solid arrows represent the flow of information in the data programming pipeline and the dashed arrows refer to the Socratic learning feedback loop.

users feedback about how they can write more effective labeling functions. Once these features are identified, they are incorporated into the generative model to improve its accuracy.

Summary of Contributions We introduce the *Socratic learning* framework, which addresses the model misspecification issue in distant supervision by enabling knowledge transfer between generative and discriminative models. In Section 2, we outline a specific distant supervision setting and describe how Socratic learning improves that pipeline by adding a feedback loop in the process. In Section 3, we further validate our claims experimentally on a real-world data set related to text relation extraction and achieve an average 2.04 point and 0.95 point F1 score improvement over taking majority vote and data programming, respectively.

2 Methodology

Our goal is to construct a simple and efficient pipeline where we can leverage the discriminative model to programmatically "debug" the heuristic rules or labeling processes that make up the generative model. We focus on the setting of binary classification problems in which we are concerned with a population of objects $\mathcal{O}$, where each object $o \in \mathcal{O}$ is assigned: a hidden true label $Y(o) \in \{-1, 1\}$, a vector of R features $X(o) \in \{-1, 1\}^R$, and a family of M labeling functions $\Lambda(o) \in \{-1, 0, 1\}^M$, each of which encodes a noisy or *weak* guess for the true label $Y(o)$. Given a set of N training examples $O \subset \mathcal{O}$, our goal is to output a classifier that accurately estimates Y. This setting, introduced in [2], differs from the standard supervised learning setting in that, rather than having access to the true labels for training, we need to use the noisy *labeling functions* as heuristics.

2.1 Learning with Generative and Discriminative Models

A generative model G uses labeling functions to *predict* the labels for the training examples. It is represented as a factor graph and uses a distribution in the family in Equation 1 to describe the relationship between the labeling functions Λ and the true class Y. Once it learns ϕ, it can assign predicted labels Y_G to the training set.

These labels Y_G are used in conjunction with the features X to train the discriminative model D shown in Equation1, which takes as input the features X and the marginals from the generative model. This information flow from the generative to the discriminative model is described in Figure 1a.

$$G : \pi_\phi(\Lambda, Y) = \frac{1}{Z} \exp\left(\phi^T \Lambda Y\right) \quad D : L_\phi(\theta) = \mathbb{E}_{(\Lambda,Y)\sim\pi_\phi}[\log(1 + \exp(-\theta^T XY))|\Lambda] \tag{1}$$

2.2 Socratic Learning

Generative Model Misspecification In the data programming setting, it is unlikely for all labeling functions to perform uniformly over the entire dataset. Each labeling function might have a higher accuracy for a certain subset of the data compared to the rest. If critical information that relates such

Algorithm 1 Socratic Learning

Input: labeling function $\Lambda \in \{-1, 0, 1\}^{M \times N}$, $X \in \{-1, 1\}^{R \times N}$
$X_S = \emptyset$
repeat
  Learn generative model $G(\Lambda, X_S, Y)$ and compute labels Y_G
  Learn discriminative model $D(Y|X)$ and compute labels Y_D
  Compute disagreement $V = -Y_G Y_D$
  Select features X_S that are most indicative of V
until performance stops improving

latent classes in the data to these heuristics is overlooked, it can lead to an under-specified generative model.

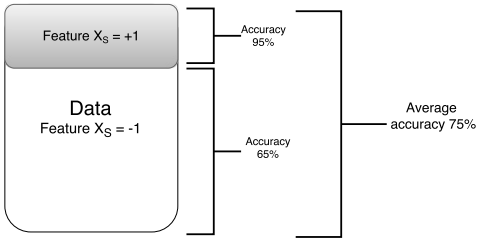


Figure 2: Toy example of a labeling function has different accuracies for different subsets of the data, described using the feature X_S.

Consider the example in Figure 2. The labeling function shown has an accuracy of 75% over the entire dataset, which is what the factor graph model in Equation 1 would ideally learn. However, there can be natural partitions in the data where the labeling functions might have very distinct performances. This suggests that if the generative model takes into account such biases that users are unaware of while writing labeling functions, it could learn more fine-grained accuracies for the heuristics, therefore improving its overall performance.

Updated Generative Model If the generative model wants to learn these latent classes in the data, it needs access to some additional information. It is important to note that these hidden classes are not likely to be random but instead depend on some underlying characteristic of the data itself — characteristics that are encoded by the features of the data. Therefore, we can assume that these latent classes in the data can be identified by some features $X_S \subset X$, as shown in Figure 2.

Socratic learning initializes with a simple generative model from Equation 1, which could be misspecified. It then uses the predictions from the generative and discriminative models, Y_G and Y_D to compute the disagreement of the two models $V = -Y_G Y_D$. The difference model finds the features X_S that are most indicative of the disagreement V via ℓ_1 regularized logistic regression. Finally, it passes these features into the generative model. This feedback loop is shown in Figure 1b. Socratic learning jointly models the relationship between the labeling functions, the true class, and the important features X_S via the factor graph:

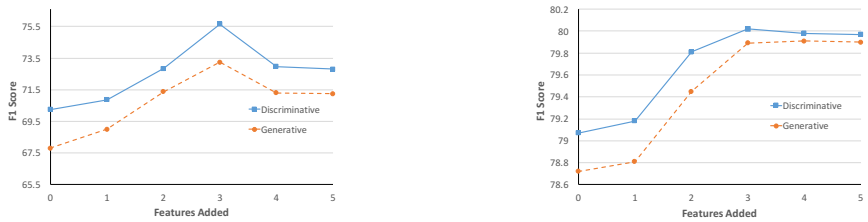
$$\pi_{\phi,\mathbf{W}}(\Lambda, Y, X_S) = \frac{1}{Z} \exp\left(\phi^T \Lambda Y + (\mathbf{W} X_S)^T \Lambda Y\right) \tag{2}$$

In the above equation, X_S represents the $K < R$ features that the generative model depends on. $\mathbf{W} \in \mathbb{R}^{M \times K}$ are weights that quantify the effect of these features on π.

Socratic learning iteratively refines the generative model by updating the set of features X_S. As more features are added to the generative model, the risk of overfitting to the training set increases. Adding in all the features ($R = K$) to the generative model would lead the model to fit more parameters than data points in most real-world examples where $N < R \times M$. An additional hold-out validation set is used to track the accuracy of the generative model. As soon as the performance on this validation set drops, the Socratic learning process is stopped. This iterative procedure is described in Algorithm 1.

Table 1: F1 scores for Socratic Learning and baseline methods. kF refers to adding k features.

		Baseline F1			Socratic learning F1	
Application	# of LFs	FS	MV	DP	1F	3F
Disease Tagging-32	32	86.47	84.42	85.28	85.30	85.29
Disease Tagging-16	16	86.47	77.98	79.07	79.18	80.02



(a) Simulation

(b) Disease Tagging-16

Figure 3: Improvement in F1 Score for generative and discriminative models with addition of features. Best improvement is achieved at 3 features in both scenarios.

3 Experimental Results

In the general relation extraction setting, we are given a set of candidates, which are substrings of a sentence, and the goal is to classify whether those candidates are in a given relationship. We use the data from the BioCreative CDR Challenge [4], where mentions of diseases are extracted from PubMed abstracts and the labeling functions are written by a team of domain experts. We compare our performance against (1) fully supervised (FS), (2) majority vote (MV) and (3) data programming (DP) F1 scores. FS uses the true labels in training, MV uses majority vote across all labeling functions as the generative model, and DP uses the data programming paradigm without considering features [2]. Table 1 displays the performance of different methods with two sets of labeling functions. Disease Tagging-32 uses 32 heavily engineered labeling functions, which suggests that any underlying biases that might have been present in the first few iterations of writing these functions were eventually alleviated. As expected, Socratic learning does not show much improvement in this case.

Disease Tagging-16 uses labeling functions that rely only on using dictionaries and regex rules. These labeling functions tend to either be more noisy or have lower coverage than the extensively hand-tuned labeling functions. Here, Socratic learning is able to increase the F1 score by 0.95 points, on average, compared to the data programming approach — an improvement which could otherwise require weeks of effort by a domain expert. This iterative improvement is shown in Figure 3b. Socratic learning finds that the presence of the phrase "for induction of..." is negatively correlated with the accuracy of labeling function *non-common diseases*, which works by searching through a dictionary of predefined non-common diseases and returns a -1 if it finds a match. Phrases like "for induction of anesthesia" show up a fair amount in a Google/PubMed query, which does not indicate a true negative disease relation. This serves as a reminder for users to consider the presence of "for induction of..." and other such common phrases when designing the labeling function *non-common diseases*. It is important to note that having access to this feature provides users an easier path to debugging and improving their generative model since they can manually add labeling functions that are aware of such situations.

4 Conclusion

We introduced Socratic learning, a novel framework that can initiate a cooperative dialog between the generative and the discriminative model. We demonstrated how the generative model can be further improved, using feedback from the discriminative model. Finally, we showed how Socratic learning works with relation extraction where it improved over the best baseline by 0.95 points. We showed how Socratic learning can educate domain experts about the hidden classes in the data that can help them write better heuristics. Additional details and experiments are in an extended version of the paper [3].

References

[1] 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.

[2] Alexander Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. *arXiv preprint arXiv:1605.07723*, 2016.

[3] Paroma Varma, Rose Yu, Dan Iter, Christopher De Sa, and Christopher Ré. Socratic learning. *arXiv preprint arXiv:1610.08123*, 2016.

[4] CH Wei, Y Peng, R Leaman, AP Davis, CJ Mattingly, J Li, TC Wiegers, and Z Lu. Overview of the biocreative v chemical disease relation (cdr) task. In *Proceedings of the Fifth BioCreative Challenge Evaluation Workshop*, pages 154–166, 2015.