Estimating Accuracy from Unlabeled Data

Emmanouil Antonios Platanios
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
e.a.platanios@cs.cmu.edu

Avrim Blum
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
avrim@cs.cmu.edu

Tom Mitchell
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
tom.mitchell@cs.cmu.edu

Abstract

We consider the question of how unlabeled data can be used to estimate the true accuracy of learned classifiers. This is an important question for any autonomous learning system that must estimate its accuracy without supervision, and also when classifiers trained from one data distribution must be applied to a new distribution (e.g., document classifiers trained on one text corpus are to be applied to a second corpus). We first show how to estimate error rates exactly from unlabeled data when given a collection of competing classifiers that make independent errors, based on the agreement rates between subsets of these classifiers. We further show that *even when the competing classifiers do not make independent errors, both their accuracies and error dependencies can be estimated* by making certain relaxed assumptions. Experiments on two data real-world data sets produce estimates within a few percent of the true accuracy, using solely unlabeled data. These results are of practical significance in situations where labeled data is scarce and shed light on the more general question of how the consistency among multiple functions is related to their true accuracies.

1 INTRODUCTION

Estimating accuracy of classifiers is central to machine learning and many other fields. Traditionally, one estimates accuracy of a function based on its performance over a set of labeled test examples. This paper considers the question of under what conditions is it possible to estimate accuracy based instead on *unlabeled data*. We show that accuracy can be estimated exactly from unlabeled data in the case that at least three different approximations to the same function are available, so long as these functions make independent errors and have better than chance accuracy. More interestingly, we show that even if one does not assume independent errors, one can still estimate accuracy given a sufficient number of competing approximations to the same function, by viewing the degree of independence of those approximations as an optimization criterion. We present experimental results demonstrating the success of this approach in estimating classification accuracies to within a few percentage points of their true value, in two diverse domains.

We consider a "multiple approximations" problem setting in which we have several different approximations, $\hat{f}_1, \ldots, \hat{f}_N$, to some target boolean classification function, $f : \mathcal{X} \rightarrow \{0, 1\}$, and we wish to know the true accuracies of each of these different approximations, using only unlabeled data. The multiple functions can be from any source - learned or manually constructed. One example of this setting that we consider here is taken from the Never Ending Language Learning system (NELL) [Carlson et al., 2010]. NELL learns classifiers that map noun phrases (NPs) to boolean categories such as fruit, food and vehicle. For each such boolean classification function, NELL learns several different approximations based on different views of the NP. One approximation is based on the orthographic features of the NP (e.g., if the NP ends with the letter string "burgh", it may be a city), whereas another uses phrases surrounding the NP (e.g., if the NP follows the word sequence "mayor of", it may be a city). Our aim in this paper is to find a way to estimate the error rates of each of the competing approximations to f, using only unlabeled data (e.g., many unlabeled NPs in the case of NELL).

2 RELATED WORK

Other researchers have considered variants of this "multiple approximations" setting. For example, [Blum and Mitchell, 1998] introduced the co-training algorithm which uses unlabeled data to train competing approximations to a target function by forcing them to agree on classifications of unlabeled examples. Others have used the disagreement rate between competing approximations as a distance met-

ric to perform model selection and regularization [Schuurmans et al., 2006; Bengio and Chapados, 2003]. Balcan et al. [2013] used disagreement along with an ontology to estimate the error of the prediction vector for multiclass prediction, from unlabeled data, under an assumption of independence of the input features given the labeling. Parisi et al. [2014] proposed a spectral method used to rank classifiers based on accuracy and combine their outputs to produce one final label, also under an assumption of independence of the input features given the labeling. Moreover, there has been work at developing more robust semi-supervised learning algorithms by using the concept of agreement rates [Collins and Singer, 1999] or some task specific constraints [Chang et al., 2007] to decide what should be added to the training data set. However, very few have tried to directly estimate actual per function error rates using agreement rates. In [Dasgupta et al., 2001] the authors PAC-bound the error rates using the pairwise agreement rates only, under the assumption that the functions make independent errors, and [Madani et al., 2004] estimate the average error of two predictors using their disagreements. [Donmez et al., 2010] is one of the few to estimate per-function error rates from unlabeled data. Here, the authors estimate the prediction risk for each function under the assumption that the true probability distribution of the output labels is known. Much of the emphasis of their work is on methods that use the known label distribution to estimate the error rate even of a single classifier, but agreements are used as well, especially under the assumption of conditional independence. In contrast, we propose here several methods for estimating actual function error rates from agreement rates, without making these assumptions.

The main contributions of this paper include: (1) formulating the problem of estimating the error rate of each of several approximations to the same function, based on their agreement rates over *unlabeled data*, as an optimization problem, (2) providing two different analytical methods that estimate error rates from agreement rates in this setting, one based on a set of simultaneous equations relating accuracies, agreements, and error dependencies, and a second, based on maximizing data likelihood, and (3) demonstrating the success of these two methods in two very different real-world problems. We consider our proposed methods a first step towards developing a self-reflection framework for autonomous learning systems.

3 PROPOSED METHODS

We introduce two different methods to estimate the error rates of binary functions in the multiple approximations setting described in section 1. Both methods are based on the idea of looking at the consistency between the different functions' predictions in order to determine the error rates of those functions. The first method consists of matching the sample agreement rates of the functions with the exact formulas of those agreement rates written in terms of the functions' error rates. For the second method, we formulate the functions' predictions consistency as a probabilistic model and solve for the maximum likelihood estimate (MLE) of their error rates. Both methods estimate the individual error rates for each function, as well as the joint error rates of all possible subsets of those functions, based on the predictions made by these functions over a sample of unlabeled instances $X_1, \ldots, X_S$.

In the following sections we denote the input data by X and the true binary output label by Y. We assume the input data X are drawn from some unknown distribution $P(X) = \mathcal{D}$, and $Y \in \{0, 1\}$. Let us consider N functions, $\hat{f}_1(X), \ldots, \hat{f}_N(X)$ which attempt to model the mapping from X to Y. For example, each function might be the result of a different learning algorithm, or might use a different subset of the features of X as input. We define the error event $E_{\mathcal{A}}$ of a set of functions $\mathcal{A}$ as an event in which every function in $\mathcal{A}$ makes an incorrect prediction:

$$E_{\mathcal{A}} = \bigcap_{i \in \mathcal{A}} \left[\hat{f}_i(X) \neq Y \right], \tag{1}$$

where $\cap$ denotes the set intersection operator and where $\mathcal{A}$ contains the indices of the functions. We define the error rate of a set of functions $\mathcal{A}$ (i.e. the probability that all functions in $\mathcal{A}$ make an error together) as:

$$e_{\mathcal{A}} = \mathbb{P}_{\mathcal{D}}(E_{\mathcal{A}}), \tag{2}$$

where $\mathbb{P}_{\mathcal{D}}(\cdot)$ denotes the probability of an event under the distribution over the input data X.

3.1 AGREEMENT RATES METHOD

Let us define the agreement rate $a_{\mathcal{A}}$, for a set of functions $\mathcal{A}$ as the probability that all of the functions' outputs[1] are the same:

$$a_{\mathcal{A}} = \mathbb{P}_{\mathcal{D}}\left(\left\{ \hat{f}_i(X) = \hat{f}_j(X), \forall i, j \in \mathcal{A} : i \neq j \right\} \right). \tag{3}$$

This quantity can be defined in terms of the error rates of the functions in $\mathcal{A}$. In order to understand how we can write the agreement rate in terms of error rates let us consider a simple example where $\mathcal{A} = \{i, j\}$ (i.e. consider just the pairwise agreement rate between the functions f_i and f_j.). The probability of two functions agreeing is equal to the probability that both make an error, plus the probability that neither makes an error:

$$a_{\{i,j\}} = \mathbb{P}_{\mathcal{D}}\left(E_{\{i\}} \cap E_{\{j\}}\right) + \mathbb{P}_{\mathcal{D}}\left(\bar{E}_{\{i\}} \cap \bar{E}_{\{j\}}\right), \tag{4}$$

where $\bar{\cdot}$ denotes the complement of a set. By using De Morgan's laws and the inclusion-exclusion principle we obtain,

[1] Here, "outputs" is equivalent to "predictions".

using the notation defined in equation (2), an expression for the agreement rate between the two functions, in terms of their individual error rates, and their joint error rate:

$$a_{\{i,j\}} = 1 - e_{\{i\}} - e_{\{j\}} + 2e_{\{i,j\}}. \tag{5}$$

In the same way we obtain the following general result for the agreement rate of a set of functions $\mathcal{A}$ of arbitrary size:

$$\begin{aligned} a_{\mathcal{A}} &= \mathbb{P}_{\mathcal{D}}\Big(\bigcap_{i\in\mathcal{A}} E_i\Big) + \mathbb{P}_{\mathcal{D}}\Big(\bigcap_{i\in\mathcal{A}} \bar{E}_i\Big), \\ &= e_{\mathcal{A}} + 1 - \mathbb{P}_{\mathcal{D}}\Big(\bigcup_{i\in\mathcal{A}} E_i\Big), \\ &= e_{\mathcal{A}} + 1 + \sum_{k=1}^{|\mathcal{A}|}\Big[(-1)^k \sum_{\substack{I\subseteq\mathcal{A} \\ |I|=k}} e_I\Big], \end{aligned} \tag{6}$$

where $\cup$ denotes the set union operator and $|\cdot|$ denotes the number of elements in a set. For the first line we used the fact that the two events, $\{\bigcap_{i\in\mathcal{A}} E_i\}$ and $\{\bigcap_{i\in\mathcal{A}} \bar{E}_i\}$, are mutually exclusive, for the second line we used one of De Morgan's laws and for the last line we used the inclusion-exclusion principle.

In the next section we examine the most basic case, assuming that functions make independent errors and have error rates below 0.5, showing that we can solve exactly for the error rates provided that we have at least 3 different functions. In the subsequent section we examine the most general case, assuming that we have N functions that make errors with unknown inter-dependencies, and show that we can formulate this as a constrained numerical optimization problem whose objective function reflects a soft prior assumption regarding the error dependencies. Experimental results presented in a later section demonstrate the practical utility of this approach, producing estimated error rates that are within a few percentage points of the true error rates, *using only unlabeled data*.

3.1.1 3 Functions That Make Independent Errors

When we have 3 functions that make independent errors we can replace the $e_{\{i,j\}}$ term in equation (5) with the term $e_{\{i\}}e_{\{j\}}$. In this case we have only 3 unknown variables (i.e. the individual function error rates) and we have $\binom{3}{2} = 3$ equations (i.e. equation (5), for $1 \leq i < j \leq 3$). Therefore, we can directly solve for each error rate in terms of the three observed agreement rates:

$$e_{\{i\}} = \frac{c \pm \left(1 - 2a_{\{j,k\}}\right)}{\pm 2\left(1 - 2a_{\{j,k\}}\right)}, \tag{7}$$

where $i \in \{1,2,3\}$, $j,k \in \{1,2,3\} \setminus i$ with $j < k$ and:

$$c = \sqrt{\left(2a_{\{1,2\}} - 1\right)\left(2a_{\{1,3\}} - 1\right)\left(2a_{\{2,3\}} - 1\right)}, \tag{8}$$

where, for a set B and an element of that set b, the notation $B\backslash b$ denotes the set containing all elements in B except b. In practical applications, we can estimate the agreement rates among the competing functions, using a sample of unlabeled data $X_1, \ldots, X_S$, as follows:

$$\hat{a}_{\{i,j\}} = \frac{1}{S}\sum_{s=1}^{S} \mathbb{I}\left\{\hat{f}_i\left(X_s\right) = \hat{f}_j\left(X_s\right)\right\}, \tag{9}$$

where $\mathbb{I}\{\cdot\}$ evaluates to one if its argument statement is true and to zero otherwise.

In most practical applications the competing functions do not make independent errors. We next consider the more difficult problem of estimating the error rates from agreement rates, but without assuming independence of the function error events.

3.1.2 N Functions That Make Dependent Errors

When we have N functions that make dependent errors we rely on the agreement rate equation (6). We consider the agreement rates for all sets $\mathcal{A} = \{\mathcal{A} \subseteq \{1,\ldots,N\} : |\mathcal{A}| \geq 2\}$ of functions (the agreement rate is uninformative for less

KEY IDEA

The significance of equations (5) and (6) is that they relate the different agreement rates $a_{\mathcal{A}}$, which are easily estimated from *unlabeled* data, to the true error rates $e_{\mathcal{A}}$ of the functions, which are difficult to estimate without labeled data. Note that if we have a system of such equations with rank equal to the number of error rates mentioned, then we can solve exactly for these error rates in terms of the observed agreement rates. This is not the case in general, because given a set of functions, $\hat{f}_1, \ldots, \hat{f}_N$, we obtain $2^N - N - 1$ agreement rate equations (one for each subset of two or more functions) expressed in terms of $2^N - 1$ error rates (one for each none-empty subset of functions). However, if we assume that the errors made by the N individual functions are independent, then we can express all of the $2^N - 1$ error rates in terms of N single-function error rates (e.g., $e_{\{i,j\}} = e_{\{i\}}e_{\{j\}}$) and we can then solve exactly for all error rates (given the additional assumption that error rates are better than chance). Furthermore, if we are unwilling to make the strong assumption that errors of individual functions are independent, then we can instead solve for the set of error rates that minimize the dependence among errors (e.g., among the infinite solutions to the underdetermined set of equations, we choose the solution that minimizes $\sum_{i,j}(e_{\{i,j\}} - e_{\{i\}}e_{\{j\}})^2$ - this idea can be easily extended to larger subsets than simply pairs of functions). The key idea in this paper is that the correspondence between easily-observed agreement rates and hard-to-observe error rates given by these equations can be used as a practical basis for estimating true error rates from unlabeled data.

than two functions) and we obtain $2^N - N - 1$ equations by matching equation (6) to the sample agreement rate for each possible subset of functions. Given a sample of unlabeled data $X_1, \ldots, X_S$, the sample agreement rate is defined as:

$$\hat{a}_{\mathcal{A}} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{I}\left\{ \hat{f}_i(X_s) = \hat{f}_j(X_s), \forall i, j \in \mathcal{A} : i \neq j \right\}, \quad (10)$$

and is an unbiased estimate of the true agreement rate. Moreover, our unknown variables are all the individual function error rates along with all of the possible joint function error rates (let us denote the vector containing all those variables by $\boldsymbol{e}$); that is a total of $2^N - 1$ unknown variables.

The set of $2^N - N - 1$ equations involving $2^N - 1$ unknown variables yields an underdetermined system of equations with an infinite number of possible solutions. We therefore cast this problem as a constrained optimization problem where the agreement equations form constraints that must be satisfied and where we seek the solution that minimizes the following objective:

$$c(\boldsymbol{e}) = \sum_{\mathcal{A}:|\mathcal{A}|\geq 2} \left(e_{\mathcal{A}} - \prod_{i \in \mathcal{A}} e_i \right)^2. \quad (11)$$

It can be seen that we are basically trying to minimize the dependence between the error events[2], while satisfying all of the agreement rates constraints. We saw in section 3.1.1 that if we assume that the error events are independent, then we can obtain an exact solution. By defining our optimization problem in this way we are effectively relaxing this constraint by saying that we want to find the error rates that satisfy our constraints and that are, at the same time, as independent as possible. Most existing methods trying to estimate function error rates using only unlabeled data assume that the error events are independent; the main novelty of this method lies in the fact that *we relax all those assumptions* and make no hard or strict assumptions about our functions.

Note that we could also define different objective functions based on information we might have about our function approximations or based on different assumptions we might want to make. For example, one could try minimizing the sum of the squares of all the error rates (i.e. the L_2 norm of $\boldsymbol{e}$) in order to obtain the most optimistic error rates that satisfy the agreement rates constraints. The novelty of our method partly lies in the formulation of the error rates estimation problem using only unlabeled data as a constrained optimization problem.

In this section we defined the model we are using for this method and the optimization problem we wish to solve. We call this method the AR method (i.e. Agreement Rates method). In section 3.3 we define additional constraints that both this method and the maximum likelihood method (described in the next section) use.

[2]That can be seen from the fact that when the error events are independent we have that $e_{\mathcal{A}} = \prod_{i \in \mathcal{A}} e_i$.

3.2 MAXIMUM LIKELIHOOD METHOD

In this section we define a probabilistic model of the consistency in the functions' outputs, considering the most general case of having N functions that make potentially dependent errors. Let us denote the outputs of the functions on an i.i.d. sample of data $X_1, \ldots, X_S$ by $\hat{\boldsymbol{Y}}_s = \left[\hat{f}_1(X_s), \ldots, \hat{f}_N(X_s) \right]$, for $s \in \{1, \ldots, S\}$. The $\hat{\boldsymbol{Y}}_s$'s are independent and therefore we can define the likelihood of our model as:

$$L(\boldsymbol{e}) = \mathbb{P}_{\mathcal{D}}\left(\hat{\boldsymbol{Y}}_1, \ldots, \hat{\boldsymbol{Y}}_S | \boldsymbol{e} \right) = \prod_{s=1}^{S} \mathbb{P}_{\mathcal{D}}\left(\hat{\boldsymbol{Y}}_s | \boldsymbol{e} \right), \quad (12)$$

where the parameter vector $\boldsymbol{e}$ contains all of the possible error events probabilities. More specifically, it contains all the e_I, for all $I \subseteq \{1, \ldots, N\}$ and $|I| \in \{1, \ldots, N\}$.

Now, $\hat{\boldsymbol{Y}}_s$ contains all the function outputs given data sample X_s. In order to compute $\mathbb{P}_{\mathcal{D}}(\hat{\boldsymbol{Y}}_s | \boldsymbol{e})$ we need to consider the following two cases:

1. All functions agree with each other (i.e. $\hat{\boldsymbol{Y}}_s$ is a vector of all 1's or all 0's).
2. The functions can be split into two non-empty groups: those that output 1 and those that output 0.

The groups of functions with the same output can also be viewed as maximal cliques in the graph whose nodes consist of the functions and whose edges consist of the agreements between the functions (i.e. when there is an agreement between two functions there is an edge connecting their corresponding nodes in the graph and when there is no agreement between them there is no edge). By using this representation we call the first case the "one clique case" and the second case the "two cliques case". We are now going to consider those two cases separately.

One Clique Case: Let us denote the set of all function indices in the clique by $\mathcal{C}$ (i.e. $\mathcal{C} = \{1, \ldots, N\}$). In this case, either all functions make an error or none of them does. Therefore, for the probability of the current sample we have that:

$$\begin{aligned} \mathbb{P}_{\mathcal{D}}(\hat{\boldsymbol{Y}}_s | \boldsymbol{e}) &= \mathbb{P}_{\mathcal{D}}\left(\bigcap_{i \in \mathcal{C}} E_i \right) + \mathbb{P}_{\mathcal{D}}\left(\bigcap_{i \in \mathcal{C}} \bar{E}_i \right), \\ &= e_{\mathcal{C}} + 1 - \mathbb{P}_{\mathcal{D}}\left(\bigcup_{i \in \mathcal{C}} E_i \right), \\ &= e_{\mathcal{C}} + 1 + \sum_{k=1}^{|\mathcal{C}|} \left[(-1)^k \sum_{\substack{I \subseteq \mathcal{C} \\ |I|=k}} e_I \right], \end{aligned} \quad (13)$$

following an equivalent derivation to the one we used when defining the agreement rates in section 3.1.

Two Cliques Case: Let us denote the set of function indices in the first clique by $\mathcal{C}_1$ and those in the second clique by $\mathcal{C}_2$. Then, we have two possible events:

1. All functions in $\mathcal{C}_1$ make an error and none of the functions in $\mathcal{C}_2$ makes an error.
2. All functions in $\mathcal{C}_2$ make an error and none of the functions in $\mathcal{C}_1$ makes an error.

Let $\mathbb{P}^1_{\mathcal{D}}(\hat{\boldsymbol{Y}}_s|\boldsymbol{e})$ denote the probability of $\hat{\boldsymbol{Y}}_s$ given that the first event of those two occurs, and let $\mathbb{P}^2_{\mathcal{D}}(\hat{\boldsymbol{Y}}_s|\boldsymbol{e})$ denote the probability of $\hat{\boldsymbol{Y}}_s$ given that the second event of those two occurs. It can be easily seen that those two events are mutually exclusive and so we have that $\mathbb{P}_{\mathcal{D}}(\hat{\boldsymbol{Y}}_s|\boldsymbol{e}) = \mathbb{P}^1_{\mathcal{D}}(\hat{\boldsymbol{Y}}_s|\boldsymbol{e}) + \mathbb{P}^2_{\mathcal{D}}(\hat{\boldsymbol{Y}}_s|\boldsymbol{e})$.

Following an equivalent derivation to the one we used when defining the agreement rates in section 3.1 we have that:

$$\begin{aligned}\mathbb{P}^1_{\mathcal{D}}(\hat{\boldsymbol{Y}}_s|\boldsymbol{e}) &= \mathbb{P}_{\mathcal{D}}\Bigg(\Bigg[\bigcap_{i\in\mathcal{C}_1} E_i\Bigg] \cap \Bigg[\bigcap_{j\in\mathcal{C}_2} \bar{E}_j\Bigg]\Bigg), \\ &= \mathbb{P}_{\mathcal{D}}\Bigg(\Bigg[\bigcap_{i\in\mathcal{C}_1} E_i\Bigg] \cap \overline{\Bigg[\bigcup_{j\in\mathcal{C}_2} E_j\Bigg]}\Bigg), \\ &= e_{\mathcal{C}_1} + \sum_{k=1}^{|\mathcal{C}_2|}\Bigg[(-1)^k \sum_{\substack{I\subseteq\mathcal{C}_2\\|I|=k}} e_{\{I\cup\mathcal{C}_1\}}\Bigg].\end{aligned} \tag{14}$$

For the second line we used one of De Morgan's laws and for the last line we used a modified form of the inclusion-exclusion principle. To understand the step we used to obtain the last line in the above equation let us consider a simple case with example events A, B_1 and B_2. It is clear from the Venn diagram in figure 1, on the right, that:

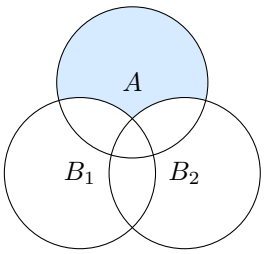

Figure 1: Venn diagram for the simple example used to explain our last step in equation (14).

$$\begin{aligned}\mathbb{P}(A \cap \overline{[B_1 \cup B_2]}) = \mathbb{P}(A) \\ - \mathbb{P}(A \cap B_1) - \mathbb{P}(A \cap B_2) \\ + \mathbb{P}(A \cap B_1 \cap B_2).\end{aligned} \tag{15}$$

Our last step in equation (14) follows from extending this result using the inclusion-exclusion principle. In the same way we also get that:

$$\mathbb{P}^2_{\mathcal{D}}\left(\hat{\boldsymbol{Y}}_s|\boldsymbol{e}\right) = e_{\mathcal{C}_2} + \sum_{k=1}^{|\mathcal{C}_1|}\Bigg[(-1)^k \sum_{\substack{I\subseteq\mathcal{C}_1\\|I|=k}} e_{\{I\cup\mathcal{C}_2\}}\Bigg]. \tag{16}$$

Having defined our likelihood function all that remains is to describe the optimization problem that we are solving to obtain the maximum likelihood estimate for $\boldsymbol{e}$. We use the negative of the natural logarithm of the likelihood function (i.e. the log-likelihood function) as our objective function, which we want to minimize, and the details of how we perform the optimization are described in section 3.3. We call this the MLE method.

3.2.1 Regularization

In this subsection we define a method which is a slight modification of the above MLE method, in that it uses a modified objective function. The objective function considered in our MLE method is non-convex and hence has many local maxima. In order to avoid getting stuck into one of those local maxima, or at least try to avoid it, we here add a regularization term to the objective function. Following the same argument we used in constructing the objective function of the AR method, we define our new objective function, which we wish to minimize, as:

$$c(\boldsymbol{e}) = -\log L(\boldsymbol{e}) + \lambda \sum_{\mathcal{A}:|\mathcal{A}|\geq 2}\left(e_{\mathcal{A}} - \prod_{i\in\mathcal{A}} e_i\right)^2, \tag{17}$$

where λ is a hyperparameter whose value can be chosen arbitrarily. We call this the maximum a posteriori (MAP) method because the added term is equivalent to adding a Gaussian prior of a special form to the error rates estimates[3]. As we will see in the experiments section the performance of this method will depend on the value chosen for λ.

3.3 OPTIMIZATION

In sections 3.1 and 3.2 we defined the optimization problems corresponding to each of our methods. For all methods we use the TOMLAB Base Module v.7.7 "conSolve" solver. In the following sections we discuss: (1) some additional constraints that apply to all methods, (2) extensions of our approach to the case where multiple approximations are learned for each of several different target functions, and (3) an approximation that can make our methods much faster, more scalable and maybe even more accurate.

3.3.1 Error Rates Constraints

Our unknown variables include both individual function error rates and joint function error rates of those events. We need to impose constraints on the values that the joint function error rates can take. These constraints follow from basic rules of probability and set theory; they represent bounding joint event probabilities using the corresponding marginal event probabilities. These constraints are defined

[3] λ can be interpreted as a function of the variance of that prior.

by the following equation:

$$e_{\mathcal{A}} \leq \min_{i \in \mathcal{A}} e_{\mathcal{A} \setminus i}, \tag{18}$$

for $|\mathcal{A}| \geq 2$. Furthermore, regarding the individual function error rates, it is easy to see that if we transform all e_i, for $i = 1, \ldots, N$, to $1 - e_i$, the resulting agreement rates are equal to the original ones. A similar result holds for the likelihood function. In order to make our models identifiable we add the constraint that $e_i \in [0, 0.5)$, for $i = 1, \ldots, N$, which simply means that our functions/binary classifiers perform better than chance. It is thus a very reasonable constraint[4].

3.3.2 Dealing With Multiple Classification Problems

Up to this point we have assumed that there is a single target function and multiple approximations to that function. More generally though, we might have multiple target functions, or problem settings, and a common set of learning algorithms used for learning each one of those. For example, this is the case in NELL, where the different target functions correspond to different boolean classification problems (e.g., classifying NPs as "cities" or not, as "locations" or not, etc.). Multiple learning methods are utilized to approximate each one of those target functions (e.g., a classifier based on the NP orthography, a second classifier based on the NP contexts, etc.), so that each such classification problem, or target function, corresponds to an instance of our "multiple approximations" problem setting.

Of course we can apply our AR or our MLE methods to estimate accuracies separately for each target classification problem (and that is what we actually did in our experiments described in section 4). However, when we have multiple target functions to be learned and multiple leaning methods shared across each, there is an interesting opportunity to further couple the error estimates across these different target functions. In equations (11) and (17) we introduced terms to minimize the dependency between the error rates of competing approximations. In the case where we have multiple target functions, we might introduce additional terms to capture other relevant assumptions. For example, we could introduce a term to minimize the difference in error dependencies between two learning methods across multiple classification problems (e.g., we could choose to minimize the difference in error dependencies between orthography-based and context-based classifiers trained for different classification problems).

3.3.3 Approximating High Order Error Rates

Once the agreement rate estimates (number of occurrences of each clique formation in the case of the MLE method and the MAP method) have been calculated, the execution time of the optimization procedure for all proposed methods does not depend on the number of provided data samples[5], S. It does however depend on the number of functions, N. This can be easily seen by considering the number of unknown variables we have which is equal to $2^N - 1$. As will be shown in section 4, the performance of all methods, in terms how good the obtained function error rate estimates are, increases with an increasing number of functions, N. It is therefore not a good idea to try to reduce N. So, we instead propose a way to reduce the execution time of the optimization procedure by approximating high order error rates, instead of estimating them directly.

We can estimate high order joint function error rates[6] using lower order function error rates by using the following formula, for $|\mathcal{A}| > M_e$, where M_e is chosen arbitrarily:

$$e_{\mathcal{A}} = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} e_{\mathcal{A} \setminus i} e_i. \tag{19}$$

With a high value of M_e we obtain better estimates but execution time is larger, and vice-versa. This estimate is based on the fact that the higher the order of the function error rates, the less significant the impact of an independence assumption between them.

Furthermore, the only available information regarding high order error rates comes from high order sample agreement rates[6], $\hat{a}_{\mathcal{A}}$, which will likely be very noisy estimates of the true agreement rates. That is because there will be very few data samples were all of the functions in $\mathcal{A}$ will agree and therefore the sample agreement rate will be computed using only a small number of data samples resulting in a noisy estimate of the true agreement rate. This motivates not directly estimating high order error rates, but instead approximating them using low order error rates. In fact, in the case that the sample agreement rates are too noisy, this approximation might even increase the quality of the obtained error rate estimates. By approximating high order error rates in the way described earlier, we are effectively ignoring the corresponding high order sample agreement rates (i.e. they are not used in our estimation) for the AR method.

[4] It is important to understand here that in order for our methods to work in the first place, this constraint *must* hold for the classifiers that we are considering.

[5] Even the execution time of the optimization procedure for the MLE method, which seems to depend on S, does not actually depend on it because there is only a fixed number of possible clique combinations one can obtain for a given number of functions, N. That number is equal to 2^{N-1}. In a large data sample we will have a lot of repeated samples in terms of the maximal cliques that they result in. We can compute the log-likelihood term for each one of those cliques only once and multiply it by the number of samples in which they each appear. This way our algorithm's execution time only depends on N.

[6] By "order" of an error rate, $e_{\mathcal{A}}$, or agreement rate, $a_{\mathcal{A}}$, we mean the number of functions in set $\mathcal{A}$, or simply $|\mathcal{A}|$.

4 EXPERIMENTS

We present here experiments using two very different data sets, to explore the ability of our methods to estimate error rates in realistic settings without domain-specific tuning. For both data sets we used a set of labeled data examples to perform our experiments. We used the data samples without their corresponding labels to estimate agreement rates, and to then estimate error rates using our methods. We used the same examples with their labels to estimate each function's true error rate, which we call from here on the "true error rate" of the function.

NELL Data Set: This data set consists of data samples where we use four binary logistic regression (LR) classifiers to predict whether a NP belongs to a specific category in the NELL knowledge base (e.g. is Monongahela a river?). The domain in this case is defined by the category (e.g. "beverage" and "river" are two different domains) and the four classifiers used were the following: (1) **ADJ**: A LR classifier that uses as features the adjectives that occur with the NP over millions of web pages, (2) **CMC**: A LR classifier that considers orthographic features of the NP (e.g. does the NP end with the letter string "burgh"? - more details can be found in [Carlson et al., 2010]), (3) **CPL**: A LR classifier that uses as features words and phrases that appear with the NP, and (4) **VERB**: A LR classifier that uses as features verbs that appear with the NP. Table 1 lists the NELL categories that we used as the domains in our experiments, along with the number of labeled examples available per category. Note the NP features used by these four classifiers are somewhat independent given the correct classification label.

Brain Data Set: Functional Magnetic Resonance Imaging (fMRI) data were collected while 8 subjects read a chapter from a popular novel [Rowling, 2012], one word at a time. The classification task is to find which of two 40 second long story passages correspond to an unlabeled 40 second time series of fMRI neural activity. For this binary classification task, we consider eight different classifiers, each making its prediction based on a different representation of the text passage (e.g., the number of letters in each word of the text passage, versus the part of speech of each word, versus emotions experienced by characters in the story, etc.). In this case different domains correspond to 11 different locations in the brain and we have 924 labeled examples per location. Additional details can be found in [Wehbe et al., 2014].

Our experimental results for both data sets are presented and discussed in the following two sections. As a performance measure we use the mean absolute deviation (MAD) between the true function error rates and the function error rates estimated from unlabeled data (i.e. we sum the absolute values of the element-wise differences of the true error rates vector and the estimated error rates vector). We compute the MAD for the individual function error rates alone, for the pairwise function error rates (i.e. for $|\mathcal{A}| = 2$) alone and for all function error rates together. Note the higher order error rates are quite small, because it is rare for every one of the competing functions to simultaneously err. Therefore, we consider the individual and pairwise function error rates to be most diagnostic of how well our approach is working.

Table 1: A listing of the 15 NELL categories we used as the domains in our experiments, along with the number of labeled examples available per category.

Category	# Examples	Category	# Examples
animal	20,733	food	19,566
beverage	18,932	fruit	18,911
bird	19,263	muscle	21,606
bodypart	21,840	person	21,700
city	21,778	protein	21,811
disease	21,827	river	21,723
drug	20,452	vegetable	18,826
fish	19,162		

4.1 NELL DATA SET RESULTS

We initially applied the AR method using only the ADJ, the CPL and the VERB classifiers, while assuming that they make independent errors. The method for estimating error rates in this case is described in section 3.1.1. In this case we estimate only the individual function error rates. The resulting MAD is 2.82×10^{-2}; that is, the average error estimate is within a few percent of the true error. Although encouraging, this MAD is poor in comparison to our less restricted methods described below, and indicates that the assumption that the classifiers make independent errors is an incorrect in this case (and in most other cases as a matter of fact). Some of the obtained error rates are not even within the interval $[0, 0.5]$ and are thus obviously incorrect, since we know by the construction of the problem that the true error rates lie in this interval. From now on we consider only the more general case of N functions that make dependent errors, thus making no independence assumptions.

Table 2 presents results for all three of our methods used with the entire NELL data set. It includes the results obtained when using all available data samples (i.e. the numbers shown in table 1) and when using only 50 data samples per category. It is clear from this table that the more data samples we have the better our methods perform, presumably due to the the more accurate estimates of the true agreement rates for the AR method, and for the other two methods, to the larger volume of evidence we have to incorporate into our likelihood. Furthermore, we see that the AR method performs significantly better than the other two methods. This could possibly be attributed to the fact that

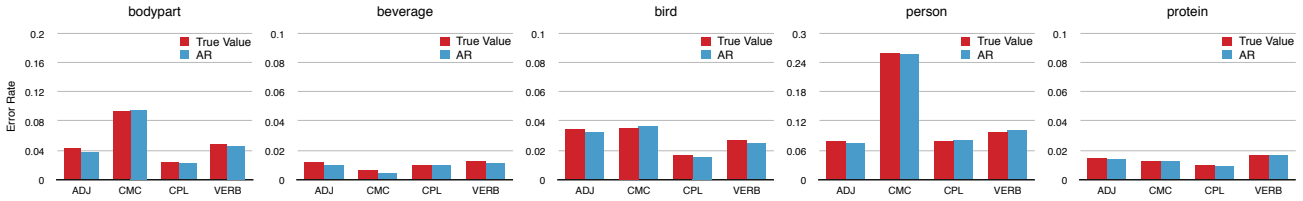

Figure 2: True errors (red bars) versus errors estimated from unlabeled data using AR method (blue bars), for four competing function approximations (ADJ, CMC, CPL and VERB), to five different target function domains (i.e. "bodypart", "beverage", "bird", "person" and"protein") using the NELL data set. Note each plot uses a different vertical scale to make it easier to observe the accuracy of the error rates estimates.

Table 2: Mean absolute deviation (MAD) of individual (Ind.), pairwise (Pair.) and all function error rates for the NELL data set, for all three proposed methods and for the cases where we use all of the available data samples and only 50 data samples per domain.

$\times 10^{-2}$	All Data Samples			50 Data Samples		
	Ind.	Pair.	All	Ind.	Pair.	All
AR	**0.49**	**0.31**	**0.29**	0.82	0.39	0.40
MLE	2.77	2.19	1.84	20.06	19.96	15.42
MAP	1.54	1.30	1.08	13.11	15.17	11.14

for this method we solve a convex optimization problem, whereas for the other two we solve a non-convex one and we possibly get stuck in local minima. Better numerical optimization solvers could possibly help with that. Finally, the MAP method performs better than the MLE method, presumably reflecting the correctness of our prior which attempts to minimize dependencies in errors across competing approximations. We did discover that the performance of the MAP method depends strongly on the choice of the λ parameter. In this case we selected $\lambda = 10$ simply because that value gave the regularization term the same order of magnitude as the log-likelihood term in the objective function. Moreover, now it becomes clear why the 2.82×10^{-2} MAD that we obtained when we assumed independent error events is quite a bad result. The AR method manages to achieve an MAD that is almost 6 times better than that.

We also run an experiment by using the approximation described in section 3.3.3 and setting $M_e = 2$ (i.e. considering only pairwise agreement rates). The individual functions MAD in this case was 0.52×10^{-2}, the pairwise one was 0.35×10^{-2} and the overall one was 0.31×10^{-2}. These results are worse than the ones we obtained without using this approximation, as expected, but they are still very good. This is important because it shows that this proposed approximation method is useful (there was a significant speedup as well - the code run 3 times faster).

From these results it is clear that the AR method, which also happens to be the simplest and fastest of the three methods we propose, performs better than the other proposed methods, for this data set, and also does not require tuning any parameters (as opposed to the MAP method).

INDEPENDENCE ASSUMPTION WEAKNESS

In order to make it more clear that the independence assumption is not very appropriate even in the case of NELL where a significant amount of effort has been put into having the NELL classifiers make independent errors, we provide here a measure of that dependence. We compute the following quantity for each domain:

$$\frac{1}{Z}\sum_{i,j}\left|\frac{e_{\{i,j\}}}{e_{\{i\}}e_{\{j\}}} - 1\right|, \qquad (20)$$

where Z is the total number of terms in the sum, and we average over all domains. That gives us a measure of the average dependence of the functions error rates across all domains. If the functions make independent errors, then this quantity should be equal to 0. We computed this quantity for the NELL data set using the sample error rates, which are an estimate of the true error rates (a pretty accurate estimate since we have about $20,000$ data samples per domain), and we obtained a value of 8.1770, which is indeed quite far from 0. That indicates why our methods, and especially the AR method, do so much better than the exact solution when assuming independent errors.

Figure 2 provides a plot of the estimated error rates for the AR method, along with the true error rates for five randomly selected NELL classification problems (plots for all regions are not included in this paper due to space constraints). This plot gives an idea of how good the AR estimates are, and helps to make sense of the reported MAD values. As is easily seen in this plot, irrespective of the exact error estimate *the ranking of the competing function approximations based on error rate is recovered exactly* by using the AR method. And that is in fact true for each of the 15 NELL target function classification problems we evaluated – not only for the five shown in this figure.

4.2 BRAIN DATA SET RESULTS

Table 3 presents results for the brain data set, obtained when using 4 of the 8 competing function approximations (randomly selected to be classifiers 1, 3, 4 and 5) and when using all 8 of them. It is clear from that table that the more competing classifiers used, the better the quality of the resulting estimates. When using all 8 classifiers the AR method performs significantly better than the other two

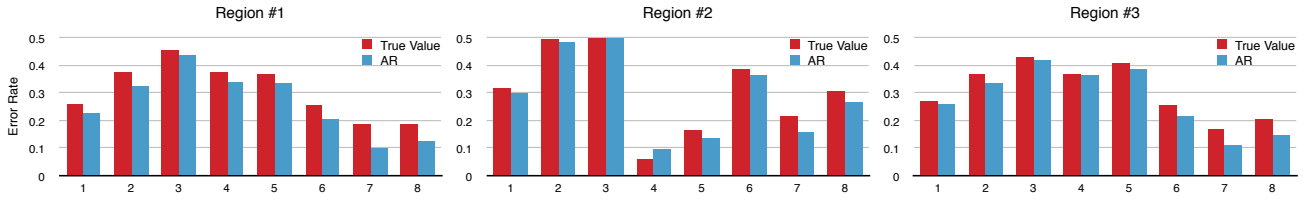

Figure 3: True errors (red bars) versus errors estimated from unlabeled data using AR method (blue bars), for eight competing function approximations (based on different story features), to three different target function domains (using neural activity from three different brain regions) using the brain data set. Note estimates from unlabeled data are quite close to true errors.

Table 3: Mean absolute error (MAE) of individual (Ind.), pairwise (Pair.) and all function error rates for the brain data set, for all three proposed methods and for the cases where we use 4 classifiers and 8 classifiers.

$\times 10^{-2}$	4 Classifiers			8 Classifiers		
	Ind.	Pair.	All	Ind.	Pair.	All
AR	10.97	6.60	6.50	**4.36**	**4.14**	**2.01**
MLE	10.60	8.34	7.64	32.02	12.33	4.50
MAP	9.61	18.19	11.16	27.95	18.60	7.26

methods. We have also included a plot of the estimated error rates for the AR method, along with the true error rates, for three randomly selected brain regions (i.e. domains), in figure 3 (it is clear from the figure that we can recover the ranking of the classifiers based on error rate, using the AR method, for this data set as well).

Note for this data set, for the case when we use 8 classifiers, the MLE method and the MAP method both perform poorly. This can probably be attributed to the optimization algorithm not being able to deal with those problems very well, due to their high dimensionality and non-convexity. These results could probably be improved by choosing a different optimization algorithm better suited for those problems. It is interesting to note that when we use only 4 classifiers the MLE and the MAP methods perform slightly better than the AR method in estimating the individual function error rates. However, they perform significantly worse when dealing with higher order error rates and so, overall, the AR method still dominates. Note than for this data set, for the MAP method, we also selected $\lambda = 10$ for the same reasons as for the NELL data set.

We also ran an experiment using the approximation described in section 3.3.3 and setting $M_e = 2$ (i.e. considering only pairwise agreement rates). The individual functions MAD in this case was 4.40×10^{-2}, the pairwise one was 4.06×10^{-2} and the overall one was 1.90×10^{-2}. These results are slightly better than the ones we obtained without using this approximation. This is important because it shows once again that this proposed approximation method is useful (there was a significant speedup as well - the code run 8 times faster). The better accuracy could possibly be attributed to two factors: (i) the problem is of much lower dimensionality and so the optimization algorithm might be dealing better with it and (ii) the high order sample agreement rates might have been bad estimates of the true agreement rates due to insufficient data and so they might have affected our methods negatively.

5 CONCLUSION

We have introduced the concept of estimating the error rate of each of several approximations to the same function, based on their agreement rates over *unlabeled data* and we have provided three different analytical methods to do so: the AR method, the MLE method and the MAP method. Our experiments showed that the AR method performs significantly better than the other two methods for both data sets we considered. Our results are very encouraging and suggest that function agreement rates are indeed very useful in estimating function error rates. We consider this work to be a first step towards developing a *self-reflection framework* for autonomous learning systems.

There are several directions we would like to pursue to further improve upon the methods introduced here. Firstly, we wish to explore other interesting natural objectives one can aim to optimize, as described in section 3.1.2. It would also be very interesting to explore possible generalizations of our models to non-boolean, discrete-valued functions, or even to real-valued functions. Finally, apart from simply estimating function error rates, we want to explore how the obtained error rate estimates can be used to improve the learning ability of a system such as NELL, for example. In this context, we could try using our estimates in order to develop a more robust co-training framework. One very direct application of our methods would be to use the estimated error rates and their dependencies in order to combine the functions' outputs and obtain one final output.

Acknowledgements

We thank Leila Wehbe for providing us with the brain data set and Alan Ritter and Siddharth Varia for providing us with the NELL data set. Finally, we thank the previously mentioned people, Jayant Krishnamurthy and the anonymous reviewers for their helpful comments. This research has been supported in part by DARPA under contract number FA8750-13-2-0005 and in part by NSF grants IIS-1065251 and CCF-1116892.

References

Maria-Florina Balcan, Avrim Blum, and Yishay Mansour. Exploiting Ontology Structures and Unlabeled Data for Learning. *International Conference on Machine Learning*, pages 1112–1120, 2013.

Yoshua Bengio and Nicolas Chapados. Extensions to Metric-Based Model Selection. *Journal of Machine Learning Research*, 3:1209–1227, March 2003.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, 1998. doi: 10.1145/279943.279962.

Andrew Carlson, Burr Settles, Justin Betteridge, Bryan Kisiel, Estevam R Hruschka Jr, and Tom M Mitchell. Toward an Architecture for Never-Ending Language Learning. In *Conference on Artificial Intelligence (AAAI)*, pages 1–8, 2010.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. Guiding Semi-Supervision with Constraint-Driven Learning. In *Annual Meeting of the Association of Computational Linguistics*, pages 280–287, Prague, Czech Republic, June 2007.

Michael Collins and Yoram Singer. Unsupervised Models for Named Entity Classification. In *Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 1–11, 1999.

Sanjoy Dasgupta, Michael L Littman, and David McAllester. PAC Generalization Bounds for Co-training. In *Neural Information Processing Systems*, pages 375–382, 2001.

Pinar Donmez, Guy Lebanon, and Krishnakumar Balasubramanian. Unsupervised Supervised Learning I: Estimating Classification and Regression Errors without Labels. *Journal of Machine Learning Research*, 11:1323–1351, April 2010.

Omid Madani, David M Pennock, and Gary W Flake. Co-Validation: Using Model Disagreement on Unlabeled Data to Validate Classification Algorithms. In *Neural Information Processing Systems*, pages 1–8, 2004.

Fabio Parisi, Francesco Strino, Boaz Nadler, and Yuval Kluger. Ranking and combining multiple predictors without labeled data. *Proceedings of the National Academy of Sciences*, pages 1–28, January 2014.

J.K. Rowling. *Harry Potter and the Sorcerer's Stone*. Harry Potter US. Pottermore Limited, 2012. ISBN 9781781100271.

Dale Schuurmans, Finnegan Southey, Dana Wilkinson, and Yuhong Guo. Metric-Based Approaches for Semi-Supervised Regression and Classification. In *Semi-Supervised Learning*, pages 1–31. 2006.

Leila Wehbe, Brian Murphy, Partha Talukdar, Alona Fyshe, Aaditya Ramdas, and Tom Mitchell. Predicting brain activity during story processing. *in review*, 2014.