Agreement-based Learning

Emmanouil Antonios Platanios
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Model selection is a problem that has occupied machine learning researchers for a long time. Recently, its importance has become evident through applications in deep learning. We propose an agreement-based learning framework that prevents many of the pitfalls associated with model selection. It relies on coupling the training of multiple models by encouraging them to agree on their predictions while training. In contrast with other model selection and combination approaches used in machine learning, the proposed framework is inspired by human learning. We also propose a learning algorithm defined within this framework which manages to significantly outperform alternatives in practice, and whose performance improves further with the availability of unlabeled data. Finally, we describe a number of potential directions for developing more flexible agreement-based learning algorithms.

1 INTRODUCTION

Model selection is a problem that traces itself back to the 14th century when William of Ockham argued that among competing hypotheses, the one with the fewest assumptions should be selected. This principle is known as *Occam's razor* and centuries later, model selection often occupies the minds of machine learning researchers. While developing algorithms that learn patterns, researchers create models that represent the problem being solved. There often exists a large set of promising models and the researcher has to decide which one to use. There exist various methods for making that decision, with cross-validation (Kohavi, 1995) being one of the most frequently used in practice. However, researchers often have to rely on their intuition and expertise, and spend a significant amount of time fine-tuning their models. This is especially true for *deep learning* which is arguably the most popular area of research in machine learning, at present. We propose a new learning framework, *agreement-based learning*, which prevents many of the pitfalls associated with model selection. It relies on coupling the training of multiple models and combining their predictions into a single output. This coupling is performed by encouraging the models to agree with each other on their predictions, while training. We show how the proposed framework is inspired by human learning, in contrast with other approaches commonly used in machine learning. Finally, we provide experimental results which exhibit how our framework successfully manages to outperform cross-validation and other ensemble methods that do not couple the training of the models.

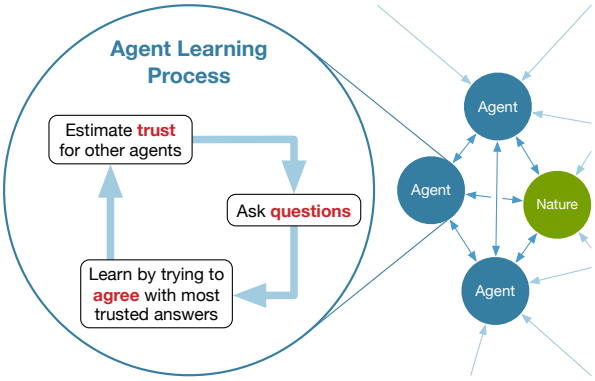

Figure 1: Illustration of the agreement-based learning framework that we propose.

We start by discussing the relationship between machine learning and human learning. Then, we propose a model for natural learning that resembles what happens in the real-world and forms the underlying idea of our agreement-based learning framework, which is formally introduced in section 3.2. In section 2, we discuss existing work in this direction and how they differ

from our proposal. Finally, in 4 we provide an extensive experimental evaluation of the agreement-based learning framework.

1.1 MACHINE VS NATURAL LEARNING

Supervised learning is one of the first and still dominant approaches used to make machines able to learn. It comprises providing the machine a set of input-output pairs and expecting it to learn a *generalized* mapping between inputs and outputs. Thus, the machine must learn to produce correct outputs for new — previously unseen — inputs. This process is inspired by the way in which humans learn. Children ask people to name things they do not recognize. For example, they will point to a mug and ask what that object is. After having seen a few examples they become able to recognize mugs, even if they look different than the mugs they have seen before. However, when people ask questions, the answers are often provided by other humans (as opposed to the surrounding natural environment, for example) and may not always be correct. Furthermore, due to the limited answers that the surrounding natural environment can provide, much of the human ability to generalize could be attributed to the interaction with other humans and this kind of question-answering.

These observations inspired us to view learning from a different perspective. Let us refer to each human and animal in nature as an *agent*. And let us do the same for nature itself (e.g., nature is an agent which tells other agents that if they drop an apple from a building, it will fall down). We argue that learning can be defined as a process of interaction between these agents, through which they try to agree with each other. More specifically, we could argue that each agent: (i) decides how much to *trust* other agents, (ii) asks multiple other agents a question[1], and (iii) forms an idea as to what the true answer to the question may be by weighting the answers based on how much the agents that provided them are trusted. Learning can thus be defined as a process in which agents learn by trying to agree with each other.

In machine learning, "agents" can refer to different learning algorithms[2] and instead of training each one independently, this model of learning proposes that their training procedures are coupled via the constraint that, while training, the agents try to agree in their predictions. Furthermore, input-output pairs that are known to be true, as traditionally used in supervised learning, can still be provided through an agent that is fixed (i.e., does not learn, such as the previously introduced nature agent). This model of learning is the underlying idea and main motivation for the *agreement-based learning* framework that we propose in this paper. We believe it can improve upon the supervised learning model in multiple ways:

1. Sharing Information: Different agents may have access to different information. For humans, this could be due to growing up in different environments whereas for learning algorithms, it could be due to the fact that different models rely on and incorporate different kinds of information (e.g., some models might use text data as input whereas some others might use image data). Agreement-based learning allows agents to implicitly share the different information they may have by providing answers to questions that other agents cannot.
2. Preventing Overfitting: *Overfitting* is a well-studied concept in machine learning. When an algorithm uses a complex model and there is limited training data, the algorithm may learn a mapping function from inputs to outputs that is much more complex than the true underlying function. This would result in bad generalization and potentially many wrong predictions for inputs the algorithm has not seen during its training. For example, modeling a linear function using a tenth order polynomial function while having observed only 3 points of the underlying line, would likely result in a bad fit. The same is true for humans to some extent. People tend to "overfit" in their own beliefs due to the limited experience they might have and the bias of only acknowledging observations that confirm their earlier beliefs (i.e., *confirmation bias*). Agreement-based learning can help prevent overfitting since it is less likely that multiple agents would overfit and agree simultaneously. This is especially true if they have access to different information.
3. Using Unlabeled Data: In machine learning, we refer to the provided input-output pairs as *labeled data* (since each input is "labeled" with some output). The agreement-based learning framework makes agents capable of also using *unlabeled data*, which consist of inputs without the corresponding outputs. This is due to the fact that agents can strive to make their outputs for these inputs agree, irrespective of the fact that no true outputs are provided. The framework that we propose can thus also be used to convert any supervised learning algorithm to a *semi-supervised learning* algorithm. This will become more clear in section 3.2, when we formally introduce our proposed approach.

[1] "Asking a question" is quite an abstract notion here. For example, dropping an apple from a building and observing what happens could constitute a question for the nature agent.

[2] By learning algorithm we refer to the algorithm and its specific underlying model. For models that have a set of hyperparameters, different values of these hyperparameters result in entirely different learning algorithms, and thus different agents.

Note that there exist situations in which learning can fail in this context; for example, all agents may agree on something that is wrong. Such situations challenge the notion of what is the truth that the agents are trying to learn, and what truth is, more generally. These questions are of a philosophical nature and are beyond the scope of this paper. However, in contrast to traditional machine learning methods, our approach does not assume an inherently true underlying distribution of data and can thus work in settings in which truth is relative. In the setting that we consider, there always exists a *nature agent* that provides answers to some questions and is fixed (i.e., does not learn and thus does not change its answers to questions with time). Since there cannot be total agreement unless other agents agree with the nature agent, the degenerate case described above is avoided.

2 RELATED WORK

The literature covers many projects that are either explicitly or implicitly related to agreement-based learning. A related area of research within the broad spectrum of machine learning is that of ensemble methods (Dietterich, 2000). These are methods that take as input the outputs of multiple models, or in some cases generate these models themselves, and combine them into a single output. The simplest ensemble method is majority voting where the most popular value among the outputs of the models is selected as the combined output. Bagging (Breiman, 1996b) and boosting (Breiman, 1996a) are methods that, instead of just taking the outputs of multiple models and combining them, they also couple their training. They do so by manipulating the distribution of the data provided to the models while training. Freund & Schapire (1997) proposed one of the most influential and widely used boosting algorithms, known as AdaBoost. AdaBoost changes the distribution of the training data such that the training algorithm can focus more on examples for which it has made erroneous predictions in the past. However, in contrast to the framework that we propose, AdaBoost places constraints on the functional form of the ensemble models (i.e., their predictions have to be boolean-valued scalars). Furthermore, bagging and boosting algorithms do not make use of unlabeled data that may be available during training.

Researchers have used the concept of agreement between multiple models in other ways, too. For example, Hanneke (2014) reviews methods that use disagreement between models to perform active learning, and others have used some notion of agreement between models in order to estimate their error rates (Collins & Singer, 1999; Dasgupta et al., 2001; Bengio & Chapados, 2003; Madani et al., 2004; Schuurmans et al., 2006; Balcan et al., 2013; Parisi et al., 2014; Platanios et al., 2014, 2016; **?**). The work most relevant to our proposal, though, is co-training (Blum & Mitchell, 1998). In co-training, a set of models is trained using the following iterative process: (i) each model is trained independently given a small set of initial training data, (ii) the models make predictions on a set of unlabeled data, (iii) their most confident predictions are added to the training dataset, and (iv) the process repeats itself from step (i). Note that, even though the models do not directly attempt to maximize agreement between themselves, they implicitly do so. This is due to the fact that, through the common training dataset that they can append their predictions to, the models effectively train each other. One possible pitfall of co-training is that if a model always produces confident yet wrong predictions, it can negatively affect the performance of other models by corrupting the training dataset. This pitfall is avoided with our framework since the models are not necessarily blindly trusted and their predictions are not equally weighted.

3 PROPOSED METHOD

We consider a traditional machine learning setting in which we have a set of models[3], $f_1, \ldots, f_M$, that, given some input x, produce some output $f_j(x)$. We assume that these models have parameters that are learned by minimizing a loss function of the form:

$$\mathcal{L}_j\left(\{x_i, y_i\}_{i=1}^N\right) = \sum_{i=1}^{N} \ell_j(f_j(x_i), y_i), \qquad (1)$$

where $j = 1, \ldots, M$, $\{x_i, y_i\}_{i=1}^N$ is the set of training examples, and ℓ_j is the loss incurred for a single training example and it penalizes disagreement of the model prediction, $f_j(x_i)$, with the provided label, y_i (e.g., cross-entropy for classification problems or mean squared error for regression problems). This function can be minimized using an appropriate optimization algorithm (e.g., gradient descent). This is a general supervised learning setting that covers both classification and regression problems.

3.1 TRADITIONAL APPROACH

Traditionally, a machine learning researcher would train all these different models separately and then do one of two things:

[3]Note that by "different," here, we do not only refer to structural differences in the models, but we include differences in hyperparameters among instantiations of the same model. For example, in the context of deep learning one could set $f_1, \ldots, f_M$ to be multi-layer perceptrons (MLPs) with different architectures or activation functions, or even completely different neural networks.

1. Model Selection: Choose a single model for making predictions. A commonly used method to perform model selection is cross-validation (Kohavi, 1995), mainly due to its strong theoretical guarantees. Selecting one model is what is frequently done in deep learning when researchers talk about *parameter tuning* (see e.g., the work of Bergstra & Bengio (2012)).
2. Model Combination (Ensemble): Combine the predictions of all models into a single prediction using some kind of ensemble method (Dietterich, 2000). A simple yet powerful and frequently used ensemble method is majority voting, where the predictions are combined by taking their mean.

We propose and describe, in the remainder of this paper, a type of ensemble method in which the training of the models is coupled (and thus the models cannot be trained separately). However, in contrast to other model combination approaches that do this (e.g., boosting (Breiman, 1996a)), our approach does not impose any strict constraints on the functional form of the models, other than that their parameters can be learned by minimizing a loss function of the form presented in equation 1.

3.2 AGREEMENT-BASED LEARNING

Inspired by natural learning (section 1.1), we propose a new approach for training, different than the traditional approaches described in the previous section. In order to consider agreement between models during training, we define an *augmented loss function* that replaces the loss function of equation 1:

$$\mathcal{AL}_j\left(\{x_i, y_i\}_{i=1}^{N}, \{x'_i\}_{i=1}^{N'}\right) = \mathcal{L}_j\left(\{x_i, y_i\}_{i=1}^{N}\right) + \lambda_j \sum_{i=1}^{N'} \ell'_j(f_j(x'_i), \hat{f}(x'_i)), \quad (2)$$

where $\{x'_i\}_{i=1}^{N'}$ is a set of training examples that are unlabeled, $\hat{f}$ represents a combined prediction from all models, which we shall call the *consensus prediction* and which is defined in the next section, ℓ'_j is a loss function that penalizes disagreement of the model prediction with the consensus prediction, and $\lambda_j \geq 0$ is a non-negative number which represents the tendency of model j to agree with the consensus. For simplicity, we set $\ell'_j = \ell_j$ and $\lambda_j = 1$ for all $j = 1, \ldots, M$, even though our framework supports any non-negative value[4]. It remains to: (i) define the consensus prediction (section 3.2.1), (ii) define the training algorithm that minimizes the augmented loss function (section 3.2.2), (iii) define the way in which predictions are made (section 3.2.3), and (iv) point out the relationship between this augmented loss function and the ideas presented in section 1.1, regarding natural learning (section 3.2.4).

This augmented loss function effectively forces the model to not only try and agree with the provided input-output pairs, but to also — at the same time — try and agree with the other models on other inputs for which we do not know the corresponding outputs. This equation also makes it easy to see why our proposed approach helps prevent overfitting, as mentioned in section 1.1. The added term to the loss function can be interpreted as a regularizer to the original optimization problem. It is thus easy to see why the added term can help the optimization solver of the original problem avoid local minima and find a better solution.

3.2.1 Consensus Definition

The consensus prediction, $\hat{f}(x)$, combines the predictions of all models, $f_1(x), \ldots, f_M(x)$, into a single prediction. It could be generated using any existing ensemble method (Dietterich, 2000). For ensemble methods that require no training, such as majority voting, the definition of the consensus prediction is straightforward. However, some consensus methods, including the ones described in the rest of this section, require training. Section 3.2.2 describes the exact way in which they are trained and in which the training procedures of all models are coupled together. In the rest of this section, we describe the consensus methods that we used for our experiments. In our experiments we consider multi-label classification problems and thus, we henceforth assume that the model predictions are vectors containing probabilities (i.e., each element corresponds to the probability of a particular label being positive).

Trainable Majority Vote. The majority vote consensus is defined as the mean of the model predictions:

$$\hat{f}_{MV}(x) \triangleq \frac{1}{M} \sum_{j=1}^{M} f_j(x). \quad (3)$$

A simple extension to this consensus method is to consider a weighted combination of the model predictions:

$$\hat{f}_{TMV}(x) \triangleq \frac{1}{M} \sum_{j=1}^{M} w_j f_j(x), \text{ where } \sum_{j=1}^{M} w_j = 1, \quad (4)$$

where the weights can be learned by minimizing a loss function. In the context of our framework, the loss function being minimized is the one defined in equation 1. For our experiments, we used the Adam optimizer (Kingma & Ba, 2014) to learn the values for the weights that minimize that objective function.

[4]Note that, in principle, negative values can also be used, but this would entail that an agent is trying to disagree with the other agents. This is an interesting idea to explore, but is outside the scope of this paper.

Semi-Supervised RBM Consensus. This is a semi-supervised extension of the method proposed by Shaham et al. (2016). The original method that the authors propose is, to the extent of our knowledge, the current state-of-the-art for unsupervised ensemble learning, which is partly why we chose to use it, in addition to ease of implementation. The authors originally propose using a Restricted Boltzmann Machine (RBM) for unsupervised ensemble learning. In a multi-label classification setting, we define one RBM per label which uses the sigmoid activation function; the visible units of each RBM correspond to the model predictions, and there exists a single hidden unit that corresponds to the consensus prediction. More specifically in our setting, we have that:

$$\mathbb{P}(\hat{f}(x), f_1(x), \ldots, f_M(x)) = \frac{1}{Z} \exp\left[a\hat{f}(x) + \sum_{j=1}^{M} b_j f_j(x) + \sum_{j=1}^{M} w_j f_j(x)\hat{f}(x)\right], \quad (5)$$

where $\mathbb{P}$ denotes a probability, a, b_j, and w_j, for $j = 1, \ldots, M$, are trainable parameters, and Z is a normalizing constant to ensure the probability over all prediction values is valid (i.e., sums to 1). The RBM is trained by maximizing the likelihood of the observed data using gradient-based optimization. The consensus prediction can then be computed by taking the mode of the following conditional distribution:

$$\mathbb{P}(\hat{f}(x) = 1 \mid f_1(x), \ldots, f_M(x)) = \sigma\left[a + \sum_{j=1}^{M} w_j f_j(x)\right], \quad (6)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. Our extension of this RBM-based method also uses gradient descent to train the RBM. However, the likelihood function we consider consists of two terms: one using unlabeled data (which is the same as that used by Shaham et al. (2016)), and one using labeled data, which is easy to define, given the definition of the RBM model using the sigmoid activation function.

Note that the agreement-based learning framework that we define is agnostic with respect to the chosen consensus method. This means that one could "plug in" any reasonably performing ensemble method and expect similar results. The choice of consensus method could depend on assumptions made about the models being trained. Some alternative methods that make different assumptions were proposed by Platanios et al. (2014, 2016); **?**.

3.2.2 Coupling the Training Procedures

The consensus prediction, $\hat{f}$, in equation 2 is a function of all model predictions. This means that if we want to train our models using the augmented loss function, they all need to be trained in tandem. For this reason, we propose algorithm 1. The algorithm assumes that an iterative optimization algorithm is used to minimize the loss function of each model. This optimization algorithm is called in the parameter update steps of lines 10 and 12. For example, these steps could represent a gradient descent step. At every iteration of the optimization, the following steps are performed:

1. An unlabeled and a labeled data batch are sampled from the training data. The sizes of those batches can be chosen arbitrarily, but note that the relative sizes of each batch will have a balancing effect similar to λ_j in equation 2. In our experiments, we set them equal.
2. The consensus prediction is computed for the unlabeled data batch, using the current parameter values for each model.
3. The parameters of each model are updated so that the value of the augmented loss function of equation 2 is reduced. Note that for the first few iterations (K_0 in algorithm 1), the original model loss function is used instead; this is due to the fact that in the beginning of training, all models are expected to produce bad predictions and thus, forcing them to agree may not be a good idea. Instead, we can wait until they can perform better. Most consensus methods are only guaranteed to work well if the majority of the models produce predictions better than chance (i.e., random). A simple heuristic is to start using the augmented loss function after at least half of the models have performance on the training data higher than some threshold (e.g., when train accuracy exceeds 50%).
4. The consensus method is re-trained every few iterations (K in algorithm 1). Note that this re-training step could also be performed at every iteration but it is preferable not to do so, since: (i) the model parameters do not generally change dramatically between each iteration, and (ii) it may be prohibitively expensive to do so at every step.

Convergence. From the definition of the augmented loss function in equation 2 and algorithm 1, it is clear that, while training, the model parameters change and thus the function being optimized is not a fixed function. In general, this could mean that the iterative optimization procedure might not converge to a solution. However, we expect that as training proceeds, the model predictions will start agreeing more often[5]. Thus, the augmented loss function will start behaving more like the original model loss function, retaining its convergence

[5]This is expected since minimizing the augmented loss function involves maximizing agreement through the consensus term.

Algorithm 1: Agreement-based learning algorithm.

Input: Models $f_1, \ldots, f_M$, consensus method $\hat{f}$, unlabeled dataset $\mathcal{D}_U = \{x'_i\}_{i=1}^{N'}$, labeled dataset $\mathcal{D}_L = \{x_i, y_i\}_{i=1}^{N}$, unlabeled data batch size N_U, labeled data batch size N_L, consensus train burn-in iterations K_0, consensus retrain frequency K

1 Initialize all model and consensus method parameters
2 $iteration \leftarrow 0$
3 **while** *at least one model has not converged* **do**
4 Sample unlabeled data batch $\mathcal{B}_U$ of size N_U from $\mathcal{D}_U$
5 Sample labeled data batch $\mathcal{B}_L$ of size N_L from $\mathcal{D}_L$
6 Compute consensus prediction $\hat{f}(x)$, for all $x \in \mathcal{B}_U$
7 **for** $j = 1, \ldots, M$ **do**
8 **if** *model f_j has not converged* **then**
9 **if** $iteration > K_0$ **then**
10 Update model f_j parameters so that the augmented loss, $\mathcal{AL}_j$, of equation 2 decreases
11 **else**
12 Update model f_j parameters so that the loss, $\mathcal{L}_j$, of equation 1 decreases
13 **if** $iteration = 0 \bmod K$ **then**
14 Re-train $\hat{f}$ using $\mathcal{D}_U$ and $\mathcal{D}_L$
15 $iteration \leftarrow iteration + 1$

Output: Trained models $f_1, \ldots, f_M$ and trained consensus $\hat{f}$.

properties. Rather than theoretical convergence guarantees, we provide empirical evidence by noting that the algorithm converged in all of our experiments (presented in section 4).

Scalability. For the proposed algorithm we can only parallelize over the different model parameter updates (i.e., parallelize the "for" loop starting in line 7 of algorithm 1). This makes our method scalable, but we can do even better. Even though in our proposed algorithm, the optimization iterations of each model are synchronous, training could also be done in a completely asynchronous and distributed manner. This way, if one model's optimization iterations are faster than those of other models, it would not have to wait until all others have performed their iterations; it could instead proceed with its own optimization loop. In section 3.2.4 we discuss some extensions that would result in better scalability properties.

Performance. An interesting observation regarding our proposed algorithm is that, as mentioned earlier, it can take a set of supervised models and train them in a semi-supervised manner. We expect this to result in better performance (e.g., better prediction accuracy) relative to a purely supervised approach, as the availability of unlabeled data increases. This is due to the fact that, as discussed in section 1.1, the agreement-based learning framework can improve generalization and help prevent overfitting. Indeed, as shown in section 4, our experiments confirm this expectation.

3.2.3 Making Predictions

As per the definition of the consensus prediction in section 3.2.1, it follows naturally to use the consensus prediction as the output prediction of our algorithm. This also means that the algorithm we propose is a type of ensemble method in which the training of the models is coupled.

Note that the main contribution of our learning framework is the agreement-based coupling of the training of multiple models. A model selection approach could still be used to decide which model is used to make predictions. In principle, the consensus method used could simply be an existing model selection method.

3.2.4 Relationship to Natural Learning

The algorithm defined in the previous section is inspired by the ideas presented in section 1.1. Even though a notion of *trust* is not established in the algorithm, the consensus prediction step and the augmented loss definition can be thought os as implementing both of the first two steps in the natural learning process described in section 1.1. Furthermore, minimizing the augmented loss function of equation 2 is equivalent to maximizing agreement with some nature agent, providing known-to-be-true examples, and at the same time maximizing agreement with the other agents (through the consensus loss term). The λ_j parameter in that equation corresponds to the trade-off between maximizing these two quantities. Therefore, the algorithm presented in the previous section offers a simple first instance of an agreement-based learning framework inspired by the natural process of learning.

In the next few paragraphs, we describe a handful of potential extensions to our algorithm that can bring us even closer to natural learning, as described earlier.

Trust. The trust of one agent for others could be explicitly modeled and used to compute *personalized consensus* predictions for that agent. There exist various unsupervised and semi-supervised accuracy estimation methods (Platanios et al., 2014, 2016; **?**) that could be used for estimating how much an agent can trust another, about a particular question (e.g, about classifying noun phrases as representing cities or not). Then, the obtained trust estimates could be used to weigh each agent's answer to a question and provide a personalized consensus prediction for an agent to use as training data. This would effectively provide an alternative consensus method that generates different consensus predictions for each agent, based on how much they trust the others.

Decentralized Communication. In our algorithm we assume that the consensus is formed by combining all

model predictions together. This can be prohibitively expensive, in practice, when a large number of models is used. Furthermore, it is also not a realistic representation of natural learning in that humans only consult a few other humans when they have questions. One way to deal with this issue would be to make agents individually able to choose which other agents they want to ask, with respect to a particular question (i.e., these could be the agents they trust the most for a particular kind of question), and what questions they want to ask, more generally. This could be achieved by using an *active learning* approach (Settles, 2012). **?** propose an approach that could be useful in this setting. This would provide a more realistic, in the sense that it is closer to natural learning, implementation and would result in a much more scalable and easily distributed approach.

There are many more interesting directions to pursue, such as a *reinforcement learning* approach (where agreement can be part of a reward function) and a game-theoretic interpretation of this model of natural learning, but these discussions are beyond the scope of this paper.

4 EXPERIMENTS

In the following sections we describe: (i) the experimental setup, (ii) the datasets we used for our experiments, and (iii) the results we obtained.

4.1 EXPERIMENTAL SETUP

For our experiments we consider a multi-label classification setting. This means that the predictions that our models produce are real-valued vectors with values in the interval $[0, 1]$. Each value represents the probability of the corresponding label being equal to 1. Before each experiment is run, we shuffle the dataset and split it into train and test parts. We run two experiments for each dataset: one using only 5% of the data to train and one using 50%. This was done to test our hypothesis that agreement-based learning offers greater benefits when less supervised training data is available. Note that the same seed was used for shuffling the datasets, for all experiments, in order to produce comparable results.

Models. For each dataset, the set of models that we consider are multi-layer perceptrons (MLPs) with various architectures. They all use leaky rectified linear activation functions (with the leakage parameter set to 0.01), but we vary the number of hidden layers and number of units per layer to define different architectures, and thus, different models. The architectures used for each dataset are listed in section 4.2.

Methods. We run experiments and compare results for the following methods:

- CV-5: This is simply 5-fold leave-one-out cross-validation and forms our baseline. For this method, we perform the following steps during training:
  1. Split the train dataset into 5 folds.
  2. For each of these folds, train each model on the remaining 4 folds and evaluate its performance on the current fold.
  3. Average these evaluation results over all 5 folds.
  4. Pick the model with the best performance and re-train it using the whole training dataset.

  The final predictions over the testing dataset are made using the best-performing model, selected in step 4.
- TMV: Train each model separately and combine their predictions using the trainable majority vote method presented in section 3.2.1. Note that this is effectively algorithm 1 with $K_0 = \infty$ (i.e., keep using the original model loss function during training, without ever switching to the augmented loss function).
- TMV-AL: Use algorithm 1 with the trainable majority vote method of section 3.2.1 as the consensus.
- RBM: Train each model separately and combine their predictions using the semi-supervised RBM method presented in section 3.2.1. Note that this is effectively algorithm 1 with $K_0 = \infty$ (i.e., keep using the original model loss function during training, without ever switching to the augmented loss function).
- RBM-AL: Use algorithm 1 with the semi-supervised RBM method of section 3.2.1 as the consensus.

For both the trainable majority vote and the semi-supervised RBM methods, we use the Adam optimizer (Kingma & Ba, 2014). Furthermore, we set a limit of $10,000$ training iterations for the first time the consensus methods are trained, but reduce that to $1,000$ for every time they are re-trained. This reduction is due to the use of warm-starts (i.e., the re-train optimization procedure is initialized at the previous solution) deeming a higher number of iterations unnecessary.

For all methods except cross-validation, we set $N_U = N_L = 128$, $K_0 = 10$, and $K = 100$. These parameter values are picked so that our experiments take a reasonable amount of time to run, but varying their values does not seem to affect the results. For the cross-validation method, we still use a batch size of 128 while training each model separately. Furthermore, we limit all experiments to a maximum number of $2,000$ training iterations in order to limit computation time[6].

[6]Note that the training procedures for all experiments converged before that limit was reached.

Table 1: Datasets used in experiments. Number of features corresponds to the dimensionality of the inputs and number of labels corresponds to the dimensionality of the outputs for each dataset. Number of instances corresponds to the total size of the data set. Density corresponds to the proportion of labels that are positive, relative to the total number of labels across all instances.

DATASET	#FEATURES	#LABELS	#INSTANCES	DENSITY (%)
DELICIOUS	501	982	16,105	1.94
MEDIAMILL	120	101	43,907	4.33
RCV1V2	47,236	101	30,000	2.85
YAHOO-ARTS	23,146	26	7,484	6.36
YAHOO-BUSINESS	21,928	27	11,214	6.08
YAHOO-COMPUTERS	34,099	30	12,444	5.60
YAHOO-EDUCATION	27,540	27	12,030	5.42
YAHOO-ENTERTAINMENT	32,001	21	12,730	6.73
YAHOO-HEALTH	30,607	30	9,205	5.99
YAHOO-REFERENCE	39,682	30	8,027	4.01
YAHOO-SCIENCE	37,197	30	6,428	3.50
YAHOO-SOCIAL	52,359	30	12,111	4.60
YAHOO-SOCIETY	31,802	27	14,512	6.28

Evaluation. The evaluation metric we use is the macro-averaged area under the precision-recall curve (we refer to this metric as AUC), computed over the testing dataset. This metric is obtained by: (i) computing the area under the precision-recall curve for each output label separately, and (ii) averaging these values. We decided to use this metric as it can measure multi-label classification performance while accounting for the trade-off between precision and recall, for all possible ways in which the probabilistic predictions can be thresholded and converted to boolean values.

Implementation. We implemented our agreement-based learning framework in Python, using the TensorFlow library (Abadi et al., 2016) as the backend for performing all computations. We have released our implementation as a standalone general purpose library that can be used for training arbitrary TensorFlow models.

4.2 DATASETS

We use several datasets obtained from the Mulan library website (available at `http://mulan.sourceforge.net/datasets-mlc.html`). Table 1 lists the datasets. The DELICIOUS dataset contains textual data of web pages as inputs, along with a set corresponding tags as outputs (Tsoumakas et al., 2008), and is used for automated tag suggestion. It was extracted from the `del.icio.us` social bookmarking site on the 1st of April, 2007. The MEDIAMILL dataset is taken from the MediaMill video indexing challenge (Snoek et al., 2006). The inputs consist of visual features derived from video sequences and the outputs consist of semantic concepts (e.g., indoors vs. outdoors). The YAHOO datasets contain data of web pages that were obtained from Yahoo's top-level categories (e.g., arts, business, etc.) and that are automatically being classified into a number of second-level categories (Ueda & Saito, 2003).

The multi-layer perceptron (MLP) architectures used for each dataset are shown in the following list. Please refer to the "Models" paragraph of the previous section for information about how these architectures are used in our experiments. The format is a comma-separated list of architectures with each architecture being encoded as a list of numbers. Each number corresponds to the number of units in the corresponding hidden layer. Therefore, `[32 16]` corresponds to two hidden layers with 32 and 16 units, respectively.

1. Delicious: `[16]`, `[256]`, `[16 16]`, `[256 256]`, `[512 256]`, `[1024 512 256]`, `[16 16 16 16]`, `[256 256 256 256]`.
2. MediaMill: `[1]`, `[8]`, `[16 8]`, `[32 16]`, `[256 128]`, `[1024 1024]`, `[2048 2048]`, `[128 32 8]`, `[128 64 32 16]`.
3. RCV1v2 and Yahoo (all datasets): `[1]`, `[8]`, `[16 8]`, `[32 16]`, `[256 128]`, `[128 32 8]`, `[128 64 32 16]`.

Note that these architectures were chosen arbitrarily in order to have both very simple and very complex models that can overfit easily.

4.3 RESULTS

The results from our experiments are shown in table 2. Our first observation is that the *RBM-AL method always outperforms all of the competing methods*. This is a very strong result for our proposed agreement-based learning framework. However, it is interesting to also note that TMV-AL does not always outperform TMV. In fact, for the experiments that we ran, it almost always gets outper-

Table 2: Area under the precision-recall curve (AUC), computed over the test dataset, for all our experiments. All values are scaled by 10^{-2} and each column corresponds to a dataset. There are two sets of experiments: one using 5% of the data as training data and one using 50%. Higher AUC values are better.

$\times 10^{-2}$	Delicious	MediaMill	RCV1v2	Yahoo									
				Arts	Business	Computers	Education	Entertainment	Health	Reference	Science	Social	Society
5% Train Data / 95% Test Data													
CV-5	11.09	15.22	52.30	13.36	14.87	14.25	13.30	22.37	15.87	9.47	10.79	12.85	16.06
TMV	11.35	16.69	46.20	16.31	12.24	16.08	18.80	29.44	20.06	23.42	15.64	21.87	19.91
TMV-AL	12.01	20.19	53.39	12.04	7.29	9.08	11.33	22.20	12.31	7.82	8.16	8.97	14.54
RBM	12.66	28.02	55.63	26.93	34.89	30.08	34.41	37.88	42.48	41.96	29.16	30.29	30.51
RBM-AL	23.30	39.86	59.87	38.26	49.07	49.86	46.48	39.12	50.65	44.63	47.37	49.55	30.61
50% Train Data / 50% Test Data													
CV-5	14.45	16.41	94.28	31.54	32.42	40.07	32.54	43.02	43.77	25.75	34.48	44.81	34.96
TMV	12.84	22.55	89.56	29.29	30.86	32.85	27.05	43.77	36.93	30.08	29.41	33.51	31.29
TMV-AL	13.35	20.38	92.34	25.13	21.02	27.10	25.86	34.63	24.30	22.43	21.63	31.15	30.97
RBM	28.64	41.00	96.43	39.89	40.00	48.92	43.58	54.32	49.82	30.72	41.00	38.32	39.44
RBM-AL	42.21	46.99	98.87	41.52	49.42	50.30	47.72	57.27	53.76	41.53	48.86	51.23	44.45

formed by TMV. The most likely reason for that is that the consensus prediction that TMV generates is not very accurate and is also potentially too sensitive to dependencies among the models[7]. Thus, a necessary condition for our proposed algorithm to work is that the consensus method used performs well. This condition seems to be satisfied by the RBM consensus. It would also probably be satisfied by other state-of-the-art ensemble methods (Platanios et al., 2014, 2016; **?**) that use similar intuition for the dependencies among the models. On a related note, we, more generally, expect that an agreement-based learning framework would always work better when the models being used are not highly dependent.

Furthermore, we observe that, even though RBM-AL always outperforms RBM, which is the second best-performing method in our experiments, it does so by a larger margin when more unlabeled data are used (i.e., 95% test data vs. 50% test data). This observation agrees with our expectation of section 3.2.2.

Finally, it is interesting to point out that RBM-AL always outperforms cross-validation. Perhaps even more interestingly, using RBM-AL with 5% of the dataset as training data outperforms using cross-validation with 50% of the dataset as training data. This is a significant result that makes agreement-based learning seem like a very strong alternative to cross-validation.

[7] A highly dependent scenario would be one where we have one model and nine copies of another model. Such a scenario could be detrimental to a bad consensus method.

5 CONCLUSION

In this paper, we have proposed an agreement-based learning framework that prevents many of the pitfalls associated with model selection. This framework is inspired by human learning and it relies on coupling the training of multiple models by encouraging them to agree on their predictions while training. We also proposed an algorithm defined within this framework which was shown to significantly outperform alternative methods in practice, and whose performance was shown to improve further with the availability of unlabeled data. There exist many potential future directions for this work. We intend to pursue the directions related to trust and decentralized communication that were discussed in section 3.2.4.

References

Abadi, Martín, Barham, Paul, Chen, Jianmin, Chen, Zhifeng, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Irving, Geoffrey, Isard, Michael, Kudlur, Manjunath, Levenberg, Josh, Monga, Rajat, Moore, Sherry, Murray, Derek G., Steiner, Benoit, Tucker, Paul, Vasudevan, Vijay, Warden, Pete, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: A System for Large-scale Machine Learning. In *USENIX Conference on Operating Systems Design and Implementation*, pp. 265–283, 2016.

Balcan, Maria-Florina, Blum, Avrim, and Mansour, Yishay. Exploiting Ontology Structures and Unlabeled

Data for Learning. *International Conference on Machine Learning*, pp. 1112–1120, 2013.

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

Bergstra, James and Bengio, Yoshua. Random Search for Hyper-parameter Optimization. *J. Mach. Learn. Res.*, 13:281–305, February 2012. ISSN 1532-4435.

Blum, Avrim and Mitchell, Tom. Combining Labeled and Unlabeled Data with Co-training. In *Annual Conference on Computational Learning Theory*, pp. 92–100, 1998.

Breiman, Leo. Bias, Variance, and Arcing Classifiers. Technical Report 460, Statistics Department, University of California at Berkeley, 1996a.

Breiman, Leo. Bagging Predictors. *Mach. Learn.*, 24(2): 123–140, August 1996b. ISSN 0885-6125.

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

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

Dietterich, Thomas G. Ensemble Methods in Machine Learning. In *International Workshop on Multiple Classifier Systems*, pp. 1–15, 2000.

Freund, Yoav and Schapire, Robert E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, 55(1): 119–139, August 1997. ISSN 0022-0000.

Hanneke, Steve. Theory of Disagreement-Based Active Learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014. ISSN 1935-8237. doi: 10.1561/2200000037.

Kingma, Diederik P. and Ba, Jimmy. Adam: A Method for Stochastic Optimization. *Computing Research Repository*, 2014. URL `https://arxiv.org/abs/1412.6980`.

Kohavi, Ron. A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence - Volume 2*, pp. 1137–1143, 1995.

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

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

Platanios, Emmanouil Antonios, Blum, Avrim, and Mitchell, Tom M. Estimating Accuracy from Unlabeled Data. In *Conference on Uncertainty in Artificial Intelligence*, 2014.

Platanios, Emmanouil Antonios, Dubey, Avinava, and Mitchell, Tom M. Estimating Accuracy from Unlabeled Data: A Bayesian Approach. In *International Conference on Machine Learning*, pp. 1416–1425, 2016.

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

Settles, Burr. *Active Learning*, volume 6. Morgan & Claypool Publishers, June 2012.

Shaham, Uri, Cheng, Xiuyuan, Dror, Omer, Jaffe, Ariel, Nadler, Boaz, Chang, Joseph T., and Kluger, Yuval. A Deep Learning Approach to Unsupervised Ensemble Learning. In *International Conference on Machine Learning*, pp. 30–39, 2016.

Snoek, Cees G. M., Worring, Marcel, van Gemert, Jan C., Geusebroek, Jan-Mark, and Smeulders, Arnold W. M. The Challenge Problem for Automated Detection of 101 Semantic Concepts in Multimedia. In *ACM International Conference on Multimedia*, pp. 421–430, 2006.

Tsoumakas, Grigorios, Katakis, Ioannis, and Vlahavas, Ioannis P. Effective and Efficient Multilabel Classification in Domains with Large Number of Labels. In *ECML/PKDD 2008 Workshop on Mining Multidimensional Data*, 2008.

Ueda, Naonori and Saito, Kazumi. Parametric mixture models for multi-labeled text. In *Advances in Neural Information Processing Systems*, pp. 721–728, 2003.