Experiments on Multistrategy Learning by Meta-Learning

Philip K. Chan and Salvatore J. Stolfo

Department of Computer Science
Columbia University
New York, NY 10027
pkc@cs.columbia.edu and sal@cs.columbia.edu

Abstract

In this paper, we propose *meta-learning* as a general technique to combine the results of multiple learning algorithms, each applied to a set of training data. We detail several meta-learning strategies for combining independently learned classifiers, each computed by different algorithms, to improve overall prediction accuracy. The overall resulting classifier is composed of the classifiers generated by the different learning algorithms and a *meta-classifier* generated by a meta-learning strategy. The strategies described here are independent of the learning algorithms used. Preliminary experiments using different strategies and learning algorithms on two molecular biology sequence analysis data sets demonstrate encouraging results. Machine learning techniques are central to automated knowledge discovery systems and hence our approach can enhance the effectiveness of such systems.

1 Introduction

The information age provides us with easy access to a wide range of information, however, it also creates databases of increasing size and complexity. The task of analyzing such vast amounts of data by hand with the goal of gaining new insights becomes an impossible one; automated techniques are therefore necessary to cope with this problem. In the fields of machine learning and knowledge discovery, researchers have been developing automated techniques to learn new knowledge by analyzing data to uncover subtle relationships and patterns of events that are not immediately discernible by direct human inspection or are overlooked due to the sheer volume of data. Various machine learning algorithms have been proposed to learn descriptive relationships and uncover rules that classify and explain what the data mean. These techniques are central to automated knowledge discovery systems [12], which generally also contain statistical data analysis tools. Attaining high prediction accuracy is the primary goal of most of the work on *inductive learning* techniques (techniques that use minimal or no domain knowledge and are further discussed in Section 2) and is also the focus of our approach described here.

Most research in inductive learning focuses on the conception and evaluation of distinct learning strategies embodied by an individual algorithm. Since different algorithms have different representations and search heuristics, different search spaces are being explored and hence potentially diverse results can be obtained from different algorithms. Mitchell [13] refers to this phenomenon as *inductive bias*. That is, the outcome of running an algorithm is biased in a certain direction. Furthermore, different data sets have different characteristics and the performance of different algorithms on these data sets might differ. In other words, to date there is no single algorithm that works best on all kinds of data sets. Hence, we postulate that it is beneficial to build a framework that allows the intelligent integration of different learning algorithms to be used in diverse situations.

Recently, several researchers have proposed implementing learning systems by integrating in some fashion a number of different strategies and algorithms to boost overall accuracy [23, 24, 19]. The basic notion behind this integration is to complement the different underlying learning strategies embodied by different learning algorithms by effectively reducing the space of incorrect classifications of a learned concept. There are many ways of integrating different learning algorithms. For example, work on integrating inductive and explanation-based learning [10, 20] requires a complicated new algorithm that implements both approaches to learning in a single system. However, not much work has been done on combining different learning systems in a loose fashion by essentially learning a new system that knows how to combine the collective outputs of the constituent systems. One advantage of this approach is its simplicity in treating the individual learning systems as black boxes with little or no modification required to achieve a final system. Therefore, individual systems can be added or replaced with relative ease.

Wolpert [23] presents a theory of *stacked generalization* (meta-learning). Several (level 0) classifiers are first learned from the same training set. The predictions made by these classifiers on the training set and the correct classifications form the training set of the next level (level 1) classifier. When an instance is being classified, the level 0 classifiers first make their predictions on the instance. The predictions are then presented to the level 1 classifier, which makes the final prediction. Zhang et al.'s [24] work utilizes a similar approach to learn a *combiner* based on the predictions made by three different classifiers. The work from the two groups demonstrates that meta-learning can improve overall prediction accuracy. Silver et al.'s [18] work also employs multiple learners, but no learning is involved beyond those learners (i.e., at the meta level). The Machine Learning Toolbox project provides a set of learning algorithms as individual tools without much interaction among them.

In this paper we present the concept of *meta-learning*, introduced in [3], and its use in coalescing the results from multiple inductive learning systems to improve accuracy. Meta-learning can also be used to combine results from a set of parallel or distributed learning processes to improve learning speed. In this paper we focus primarily on boosting accuracy, leaving details of our preliminary experiments to improve speed to a companion paper [5]. The ultimate goal of this work is to improve both the accuracy and efficiency of machine learning by means of parallel processing of multiple learning systems applied to massive amounts of training data, which is further discussed in Section 6.

There are many ways one might imagine to combine learned classifiers. For this paper, we detail several approaches and measure their behavior on a set of standard tasks often cited in the literature. Thus, this work may be viewed as exploratory to determine the efficacy of the general approach, but certainly it is not exhaustive. Section 2 discusses meta-learning and how it facilitates a multiple learner framework. Section 3 details our strategies for boosting accuracy by meta-learning. Section 4 discusses our preliminary experiments. Section 5 presents our experimental results and Section 6 discusses our findings and work in progress.

2 Meta-learning

Meta-learning can be loosely defined as learning from information generated by a learner(s). It can also be viewed as the learning of meta-knowledge on the learned information. In our work we concentrate on learning from the output of inductive learning (or learning-from-examples) systems. Given a set of labeled (classified) examples, the task of inductive learning is to form concepts that describe relationships among the data and accurately predict the classification of future unclassified instances. These concepts are also called *classifiers*. In the inductive learning case meta-learning means learning from the classifiers produced by the learners and the *predictions* of these classifiers on *training data*. A classifier (or concept) is the output of an inductive learning system and a prediction (or classification) is the predicted class generated by a classifier when an instance is supplied. That is, we are interested in the output of the learners, not the learners themselves. Moreover, the training data presented to the learners initially are also available to the *meta-learner* if warranted.

Meta-learning is a general technique to coalesce the results of multiple learners. In this paper we concentrate on using meta-learning to combine different learners to improve prediction accuracy. We call this approach *multistrategy hypothesis boosting*. (The term *hypothesis boosting* was introduced by Schapire [17]; his work on hypothesis boosting using only one strategy is discussed in Section 6.) It involves applying multiple algorithms on the same set of data and the results of the learned concepts are combined by meta-learning. Our goal is to achieve an overall accuracy that is higher than the accuracy obtained by any of the individual learning algorithms. Similar ideas were first proposed in [19] in the domain of speech recognition. The aforementioned approaches used by Wolpert [23] and Zhang et al. [24] are examples of this approach. In the next section we will discuss our approach to using meta-learning for multistrategy hypothesis boosting.

3 Multistrategy Hypothesis Boosting

The objective here is to improve prediction accuracy by exploring the diversity of multiple learning algorithms through meta-learning. This is achieved by a basic configuration which has several different *base learners* and one *meta-learner* that learns from the output of the base leaners, as depicted in Figure 1. The meta-learner may employ the same algorithm as one of the base learners or a completely distinct algorithm. Each of the base learners is provided with the entire training set of raw data. However, the training set for the meta-learner (meta-level training data) varies according to the strategies described below, and is quite different from the original training set. Each base-learner generates a *base classifier* and the meta-learner generates a *meta-classifier*. Note that the meta-learner does not aim at picking the "best" base classifier; instead it tries to combine the classifiers. That is, the prediction accuracy of the overall system is not limited to the most accurate base classifier. It is our intention to generate an overall system that outperforms the underlying base classifiers.

There are in general two types of information the meta-learner can combine: the learned *base classifiers* and the *predictions* of the learned base classifiers. The first type of information consists of *concept descriptions* in the base classifiers (or concepts). Some common concept descriptions are represented in the form of decision trees, rules, and networks. Since we are aiming at diversity in the base learners, the learning algorithms chosen usually have different representations for their learned classifiers. Hence, in order to combine the classifiers, we need to define a uniform representation to which the different learned classifiers are trans-

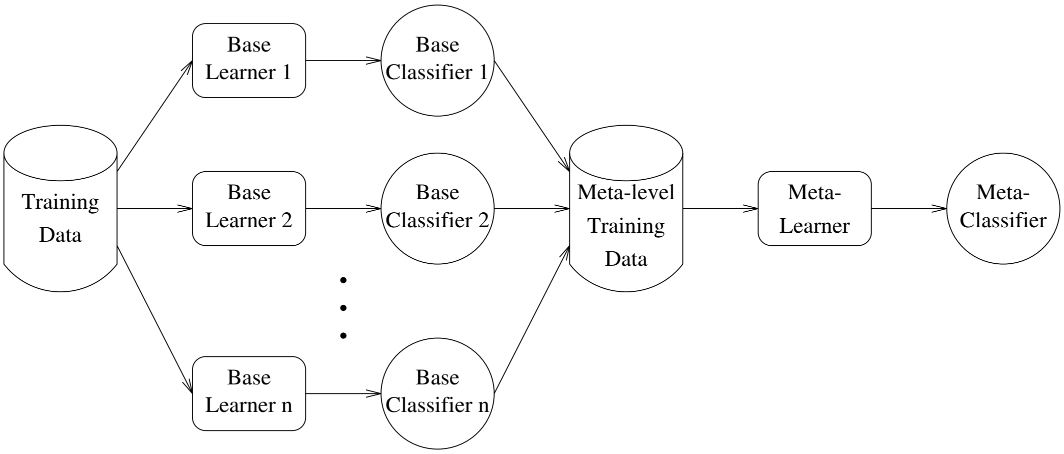

Figure 1: Multistrategy Hypothesis Boosting

lated. However, it is difficult to define such a representation to encapsulate all the other representations without losing a significant amount of information during the translation. For instance, it is very difficult to define a common representation to integrate the distance functions and exemplars computed by a nearest-neighbor learning algorithm with the tree computed by a decision tree learning algorithm. Because of this difficulty, one might define a uniform representation that limits the types of representation that can be supported and hence the choice of learning algorithms.

An alternative strategy is to integrate the *predictions* of the learned classifiers for the training set leaving the internal organization of each classifier completely transparent. These predictions are hypothesized classes present in the training data and can be categorical or associated with some numeric measure (e.g., probabilities, confidence values, and distances). In this case, the problem of finding a common ground is much less severe. For instance, classes with numeric measures can be treated as categorical (by picking the class with the highest value). Since any learner can be employed in this case, we focus our work on combining predictions from the learned classifiers. In addition, since converting categorical predictions to ones with numeric measures is undesirable or impossible, we concentrate on combining categorical predictions.

We experimented with three types of meta-learning strategies (*combiner*, *arbiter*, and *hybrid*) in combining predictions which we discuss in the following sections. For pedagogical reasons, our discussion assumes three base learners and one meta-learner.

3.1 Combiner Strategy

In the *combiner* strategy, the predictions of the learned base classifiers on the training set form the basis of the meta-learner's training set. A *composition rule*, which varies in different schemes, determines the content of training examples for the meta-learner. From these examples, the meta-learner generates a meta-classifier, that we call a *combiner*. In classifying an instance, the base classifiers first generate their predictions. Based on the same composition rule, a new instance is generated from the predictions, which is then classified by the combiner (see Figure 2). The aim of this strategy is to coalesce the predictions from the base classifiers by learning the relationship between these predictions and the correct prediction.

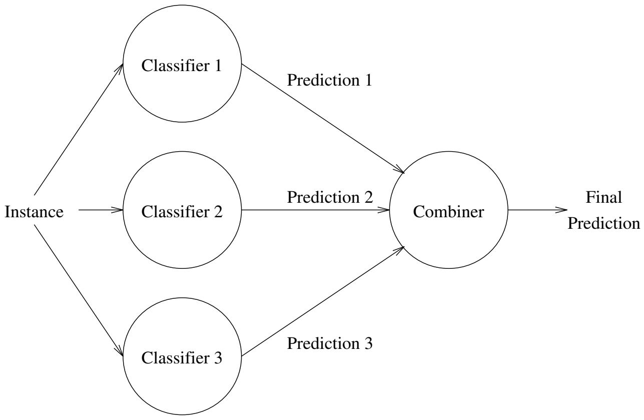

Figure 2: Classification in the *combiner* strategy

We experimented with three schemes for the composition rule. First, three predictions, $C_1(x)$, $C_2(x)$, and $C_3(x)$, for each example x in the original training set of examples, E, are generated by three separate classifiers, C_1, C_2, and C_3. These predicted classifications are used to form a new set of "meta-level training instances," T, which is used as input to a learning algorithm that computes a combiner. The manner in which T is computed varies according to the schemes as defined below. In the following definitions, $class(x)$ denotes the correct classification of example x as specified in the training set, E.

1. Return meta-level training instances with the correct classification and the predictions; i.e., $T =$

Class	Attribute vector	Example	Base classifiers' predictions		
$class(x)$	$attrvec(x)$	x	$C_1(x)$	$C_2(x)$	$C_3(x)$
table	$attrvec_1$	x_1	table	table	table
chair	$attrvec_2$	x_2	table	chair	lamp
lamp	$attrvec_3$	x_3	lamp	chair	chair

Training set for the *meta-class* combiner scheme	
Class	Attribute vector
table	(table, table, table)
chair	(table, chair, lamp)
lamp	(lamp, chair, chair)

Training set for the *meta-different* arbiter scheme	
Class	Attribute vector
chair	$attrvec_2$
lamp	$attrvec_3$

Training set for the *meta-different-class-attribute* hybrid scheme	
Class	Attribute vector
chair	(table, chair, lamp, $attrvec_2$)
lamp	(lamp, chair, chair, $attrvec_3$)

Figure 3: Sample training sets generated by the *combiner*, *arbiter*, and *hybrid* strategies

$\{(class(x), C_1(x), C_2(x), C_3(x)) \mid x \in E\}$. This scheme was also used by Wolpert [23]. (For further reference, this scheme is denoted as *meta-class*.) A sample training set is displayed in Figure 3.

2. Return meta-level training instances similar to those in the first (*meta-class*) scheme with the addition of the original attribute vectors in the training examples; i.e., $T = \{(class(x), C_1(x), C_2(x), C_3(x), attrvec(x)) \mid x \in E\}$. (Henceforth, this scheme is denoted as *meta-class-attribute*.)

3. Return meta-level training instances similar to those in the *meta-class* scheme except that each prediction, $C_i(x)$, has m binary predictions, $C_{i_1}(x), \ldots, C_{i_m}(x)$, where m is the number of classes. Each prediction, $C_{i_j}(x)$, is produced from a binary classifier, which is trained on examples that are labeled with classes j and $\neg j$. In other words, we are using more specialized base classifiers and attempting to learn the correlation between the binary predictions and the correct prediction. For concreteness, $T = \{(class(x), C_{1_1}(x), \ldots, C_{1_m}(x), C_{2_1}(x), \ldots, C_{2_m}(x), C_{3_1}(x), \ldots, C_{3_m}(x)) \mid x \in E\}$. (Henceforth, this scheme is denoted as *meta-class-binary*.)

These three schemes for the composition rule are defined in the context of forming a training set for the combiner. These composition rules are also used in a similar manner during classification after a combiner has been computed. Given an instance whose classification is sought, we first compute the classifications predicted by each of the base classifiers. The composition rule is then applied to generate a single meta-level test instance, which is then classified by the combiner to produce the final predicted class of the original test datum.

3.2 Arbiter Strategy

In the *arbiter* strategy, the training set for the meta-learner is a subset of the training set for the base learners; i.e. the meta-level training instances are a particular distribution of the raw training set E. The predictions of the learned base classifiers for the training set and a *selection rule*, which varies in different schemes, determines which subset will constitute the meta-learner's training set. (This contrasts with the combiner strategy, which has the same number of examples for the base classifier as for the combiner. Also, the meta-level training instances for a combiner incorporate additional information than just the raw training data.) Based on this training set, the meta-learner generates a meta-classifier, in this case called an *arbiter*. In classifying an instance, the base classifiers first generate their predictions. These predictions, together with the arbiter's prediction and a corresponding *arbitration rule*, generate the final prediction (see Figure 4). (This contrasts with the multi-level arbiter trees introduced in [5].) In this strategy one learns to arbitrate among the potentially different predictions from the base classifiers, instead of learning to coalesce the predictions as in the combiner strategy. We first describe the schemes for the selection rule and then those for the arbitration rule.

We experimented with two schemes for the *selection rule*, which chooses training examples for an arbiter. In essence the schemes select examples that are confusing to the three base classifiers, from which an arbiter is learned. Based on three predictions, $C_1(x)$, $C_2(x)$, and $C_3(x)$, for each example x in a set of training examples, E, each scheme generates a set of training examples, T ($\subseteq E$), for the arbiter. The two versions of this selection rule implemented and reported here include:

1. Return instances with predictions that disagree; i.e., $T =$

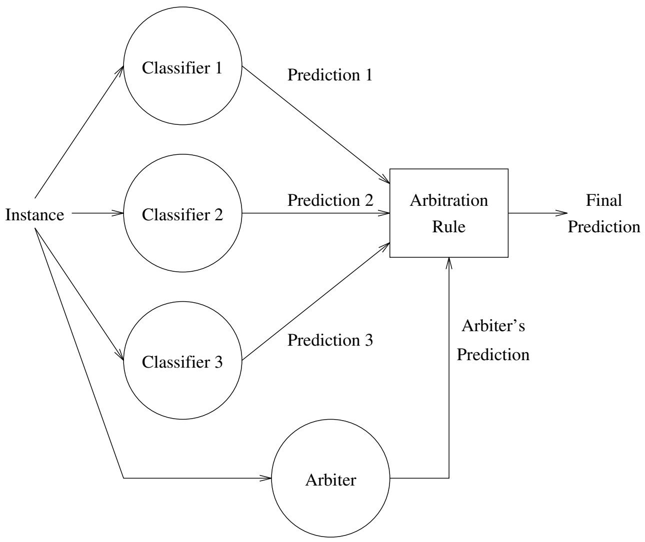

Figure 4: Classification in the *arbiter* strategy

$T_d = \{x \in E \mid (C_1(x) \neq C_2(x)) \vee (C_2(x) \neq C_3(x))\}$. Thus, the arbiter is used to decide between conflicting classifications. However, it does not include classifications that agree but are incorrect. (We refer to this scheme as *meta-different*.) For example, in Figure 3, the three base classifiers do not share the same prediction for examples x_2 and x_3 so these two examples constitute the set T_d.

2. Return instances with predictions that disagree, T_d, as in the first (meta-different) case, but also instances with predictions that agree but are incorrect; i.e, $T = T_d \cup T_i$, where $T_i = \{x \in E \mid (C_1(x) = C_2(x) = C_3(x)) \wedge (class(x) \neq C_1(x))\}$. Note that we compose both cases of data that are incorrectly classified or are in disagreement. (Henceforth, we refer to this case as *meta-different-incorrect*.)

The learned arbiters are trained by some learning algorithm on the particular distinguished distributions of training data and are used in generating predictions. During the classification of an instance, y, an *arbitration rule* and the learned arbiter, A, produce a final prediction based on the three predictions, $C_1(y)$, $C_2(y)$, and $C_3(y)$, from the three base classifiers and the arbiter's own prediction, $A(y)$. The following arbitration rule applies to both schemes for the selection rule described above.

1&2. Return the simple vote of the base and arbiter's predictions, breaking ties in favor of the arbiter's prediction; i.e., if there are no ties, return $vote(C_1(y), C_2(y), C_3(y), A(y))$, otherwise return $A(y)$. For example, if the three classifiers predicts `table`, `chair`, and `table` and the arbiter predicts `chair` (i.e., a tie), the final prediction is `chair`.

3.3 Hybrid Strategy

We integrate the *combiner* and *arbiter* strategies in the *hybrid* strategy. Given the predictions of the base classifiers on the original training set, a *selection rule* picks examples from the training set as in the arbiter strategy. However, the training set for the meta-learner is generated by a *composition rule* applied to the distribution of training data (a subset of E) as defined in the combiner strategy. Thus, the hybrid strategy attempts to improve the arbiter strategy by correcting the predictions of the "confused" examples. It does so by using the combiner strategy to coalesce the predicted classifications of data in disagreement by the base classifiers. A learning algorithm then generates a meta-classifier, effectively a *combiner*, from this training set.

When a test instance is classified, the base classifiers first generate their predictions. These predictions are then composed to form a meta-level instance for the learned meta-classifier using the same composition rule. The meta-classifier then produces the final prediction. The hybrid strategy thus attempts to improve the arbiter strategy by coalescing predictions instead of purely arbitrating among them.

We experimented with two combinations of composition and selection rules, though any combination of the rules is possible:

1. Select examples that have different predictions from the base classifiers and the predictions, together with the correct classes and attribute vectors form the training set for the meta-learner. This integrates the *meta-different* and *meta-class-attribute* schemes. (Henceforth, we refer to this scheme as *meta-different-class-attribute*.) A sample training set is displayed in Figure 3.

2. Select examples that have different or incorrect predictions from the base classifiers and the predictions, together with the correct classes and attribute vectors form the training set for the meta-learner. This integrates the *meta-different-incorrect* and *meta-class-attribute* schemes. (Henceforth, denoted as *meta-different-incorrect-class-attribute*.)

We described the combiner, arbiter, and hybrid strategies for meta-learning in multistrategy hypothesis boosting. It is important to note the difference between the combiner and arbiter strategies. The combiner strategy tries to find relationships among the predictions generated by the classifiers and the correct predictions. A combiner is a "learned function" that determines the final prediction given a set of predictions. For example, given an unlabeled instance x, the combiner may learn a rule stating that if classifier C_1 predicts `table`, C_2 predicts `chair`, and C_3 predicts `chair`, then the combiner predicts `lamp` (i.e., possibly a completely different prediction from either classifier). However, the arbiter strategy attempts to arbitrate among the conflicting predictions and an arbiter is just another classifier, but trained on a biased distribution of the original examples. Here, for example, when C_1, C_2, and C_3's predictions disagree, the arbiter makes its own prediction, which could be completely different from the three base predictions, and a vote determines the final prediction. In the next section we describe our experimental results to explore the effectiveness of these strategies. We applied the strategies to two real-world scientific tasks employing different combinations of several standard machine learning algorithms.

4 Experiments

Four inductive learning algorithms were used in our experiments. We obtained ID3 [16] and CART [1] as part of the IND package [2] from NASA Ames Research Center; both algorithms compute decision trees. WPEBLS is the weighted version of PEBLS [7], which is a nearest-neighbor learning algorithm. BAYES is a Bayesian classifier that is based on computing conditional probabilities (frequency distributions), which is described in [6]. The latter two algorithms were reimplemented in C.

Two molecular biology sequence analysis data sets, obtained from the UCI Machine Learning Database, were used in our studies. The secondary protein structure data set (SS) [15], courtesy of Qian and Sejnowski, contains sequences of amino acids and the secondary structures at the corresponding positions. There are three structures and 20 amino acids (21 attributes because of a spacer [15]) in the data. The amino acid sequences were split into shorter sequences of length 13 according to a windowing technique used in [15]. The sequences were then divided into a disjoint training and test set, according to the distribution described in [15]. The training set, E, for this task has 18105 instances and the test set has 3520. The DNA splice junction data set (SJ) [14], courtesy of Noordewier, Towell, and Shavlik, contains sequences of nucleotides and the type of splice junction, if any, at the center of each sequence (three classes). Each sequence has 60 nucleotides with eight different values each (four base ones plus four combinations). Some 2552 sequences were randomly picked as the training set, E, for this task and the rest, 638 sequences, served as the test set.

The predictions used in the training set of the meta-learner were generated by a two-fold cross validation scheme. The training set is first split in two halves. Each of the three base classifiers were trained on the first half and the second half is used to generate predictions. Similarly, each base classifier is trained on the second half and the first half is used to generate predictions. The predictions from the two halves are merged and then used in constructing the training set for the meta-learner. The objective is to mimic the behavior of the learned classifiers when unseen instances are classified. That is, the meta-learner is trained on predictions of unseen instances in the training set. The base learners are also trained on the entire training set to generate base classifiers, which are then used with the learned meta-classifier in the classification process.

We performed experiments on the different schemes for the combiner, arbiter, and hybrid strategies. Different combinations of three base and one meta-learner are explored on the two data sets and the results are presented in Tables 1 and 2. Each table has four subtables and each subtable presents results from a different combination of base learners. The names of meta-learning schemes are abbreviated in the tables: *m-c* represents *meta-class*, *m-c-a* represents *meta-class-attribute*, and so on. Results for the two data sets with single-strategy classifiers are displayed in Table 3. In addition, we experimented with a windowing scheme used in Zhang's work [24], which is specific to the SS data. This scheme is similar to the *meta-class* scheme described above. However, in addition to the three predictions present in one training example for the meta-learner, the guesses on either side of the three predictions in the sequence (*windows*) are also present in the example. We denote this scheme as *meta-class-window* (or *m-c-w* in the tables).

Furthermore, several non-meta-learning approaches were applied for comparison. *Vote* is a simple voting scheme applied to the predictions from the base classifiers. *Freq* predicts the most frequent correct class with respect to a combination of predictions in the training set[1]. That is, for a given combination of predictions (m^c combinations for m classes and c classifiers), *freq* predicts the most frequent correct class in the training data. *Vote-b* is a simple voting scheme applied to the predictions from the binary base classifiers.

5 Results

There are two ways to analyze the results. First, we look at whether the employment of a meta-learner improves accuracy with respect to the underlying three base classifiers. (The presence of an improvement is denoted by a '+' in the tables.) For both sets of data, improvements were always achieved when BAYES was used as the meta-learner and the other three learning algorithms we used as the base learners, regardless of the meta-learning strategies.

Now let us consider various combinations of meta-learner and strategies with any of base learning algorithms. For the SJ data, a higher or equal accuracy was consistently attained when BAYES was the meta-learner in the *meta-class-attribute* strategy. Similarly, higher accuracy was attained when ID3 served as the meta-learner in the *meta-class* and *meta-class-attribute* strategies, regardless of the base learners used. Improvements were also observed in the *vote* and *freq* strategies. For the SS data, none of the various combinations of meta-learners and strategies attained a consistent improvement in overall accuracy.

Next, we consider whether the use of meta-learning achieves higher accuracy than the most accurate single-strategy learner (BAYES). (The presence of an improvement is denoted by a '*' in the tables.) For the SJ data, an improve-

[1] *Freq* was suggested by Wolpert (personal communication).

Table 1: Summary of prediction accuracy for protein secondary structures (%)

Base learners: ID3, CART & WPEBLS

	Meta-learner			
Scheme	BAYES	ID3	CART	WPEBLS
m-c	56.3+	55.8+	55.7+	55.1
m-c-a	60.3+	55.4	48.7	48.5
m-c-b	55.6+	56.6+	56.6+	52.7
m-c-w	56.9+	54.5	49.9	50.6
m-d	60.7+	56.4+	56.1+	53.3
m-d-i	59.8+	56.4+	53.9	52.4
m-d-c-a	60.5+	56.5+	55.7+	54.4
m-d-i-c-a	59.1+	56.4+	54.1	53.1
vote	56.3+	* better than the best single strategy		
freq	56.5+	+ better than the best base classifier		
vote-b	57.1			

Base learners: BAYES, ID3 & CART

	Meta-learner			
Scheme	BAYES	ID3	CART	WPEBLS
m-c	61.4	62.1	62.1	57.3
m-c-a	62.1	61.0	51.0	50.4
m-c-b	61.1	61.9	61.7	54.4
m-c-w	60.7	60.1	52.5	53.3
m-d	62.2+*	57.2	57.6	57.6
m-d-i	60.8	58.3	57.7	56.6
m-d-c-a	62.1	61.5	58.9	58.5
m-d-i-c-a	60.4	58.6	58.0	56.7
vote	60.6			
freq	62.1			
vote-b	60.9			

Base learners: BAYES, ID3 & WPEBLS

	Meta-learner			
Scheme	BAYES	ID3	CART	WPEBLS
m-c	60.4	62.1	62.1	55.9
m-c-a	61.9	60.6	51.0	52.6
m-c-b	60.5	61.8	61.8	55.2
m-c-w	59.9	59.5	51.4	52.9
m-d	62.0	57.4	57.3	55.7
m-d-i	60.8	59.0	56.6	54.4
m-d-c-a	61.6	60.7	57.9	56.8
m-d-i-c-a	60.8	59.3	56.1	54.6
vote	59.3			
freq	62.2+*			
vote-b	59.3			

Base learners: BAYES, CART & WPEBLS

	Meta-learner			
Scheme	BAYES	ID3	CART	WPEBLS
m-c	60.7	62.1	61.8	56.8
m-c-a	61.4	60.5	50.4	50.9
m-c-b	60.5	61.7	61.3	52.6
m-c-w	59.7	57.9	52.6	54.2
m-d	62.0	58.1	57.4	54.6
m-d-i	61.4	58.6	56.8	52.0
m-d-c-a	61.1	60.3	58.0	56.1
m-d-i-c-a	59.4	59.1	59.1	59.2
vote	59.6			
freq	60.7			
vote-b	60.9			

ment was consistently achieved when BAYES served as the meta-learner in the *meta-class-attribute* strategy, regardless of the base learners used. In fact, when the base learners were BAYES, ID3, and CART, the overall accuracy was the highest obtained. For the SS data, almost all the results did not outperform BAYES as a single-strategy learner.

In general, the *combiner* strategies performed more effectively than the *arbiter* and *hybrid* strategies. To our surprise, the hybrid schemes did not improve the arbiter strategies. In addition, Zhang's [24] *meta-class-window* strategy for the SS data did not improve accuracy with the base and meta-learners used here. His study employed a neural net algorithm and different Bayesian and nearest-neighbor learners than those reported here.

The two data sets chosen represent two different kinds of data sets: one is difficult to learn (SS) (50+% accuracy) and the other is easy to learn (SJ) (90+% accuracy). Our experiments indicate that some of our meta-learning strategies improve accuracy in the SJ data and are more effective than the non-meta-learning strategies. However, in the SS data, both meta-learning and non-meta-learning strategies are comparable. This can be attributed to the quality of predictions from the base classifiers for the two data sets. Consider the statistics we gathered from the predictions for the test set from classifiers trained by BAYES, ID3, and WPEBLS (other combinations of learners have similar statistics). In the SJ data set, on 89% of the instances all predictions from the three learned classifiers were correct, on 7% two predictions were correct, on 2% only one, and on 1% none (all incorrect). In the SS data set, on 29% of the instances all three predictions were correct, on 33% only two, on 18% only one, and on 20% none. The high percentage of having one or none correct out of three predictions in the SS data set might greatly hinder the ability of meta-learning to work. One possible solution is to increase the number of base classifiers to lower the percentage of having one or none correct predictions.

6 Discussion

Unlike Wolpert [23] and Zhang et al.'s [24] reports, we present results from all the combinations of presented strategies, base learners, and meta-learners. We have shown that improvements can be achieved consistently with a combination of a meta-learner and collection of base learners across

Table 2: Summary of prediction accuracy for RNA splice junctions (%)

Base learners: ID3, CART & WPEBLS				
	Meta-learner			
Scheme	BAYES	ID3	CART	WPEBLS
m-c	95.1+	94.8	94.8	72.7
m-c-a	96.6+*	95.0+	95.9+	95.5+
m-c-b	95.1+	94.4	94.4	74.1
m-d	96.4+	94.7	95.5+	95.3+
m-d-i	96.6+*	95.8+	95.8+	95.5+
m-d-c-a	96.1+	94.5	94.8	95.3+
m-d-i-c-a	95.9+	94.2	95.0+	95.0+
vote	95.0+	* better than the best single strategy		
freq	95.0+	+ better than the best base classifier		
vote-b	95.1+			

Base learners: BAYES, ID3 & CART				
	Meta-learner			
Scheme	BAYES	ID3	CART	WPEBLS
m-c	95.6	96.6+*	95.3	74.3
m-c-a	97.2+*	96.4	95.0	96.9+*
m-c-b	95.8	96.2	96.2	75.2
m-d	96.9+*	95.3	95.6	95.9
m-d-i	96.9+*	95.5	95.9	96.2
m-d-c-a	95.8	94.5	95.1	95.9
m-d-i-c-a	95.3	94.2	94.8	95.5
vote	95.6+			
freq	95.9+			
vote-b	95.6			

Base learners: BAYES, ID3 & WPEBLS				
	Meta-learner			
Scheme	BAYES	ID3	CART	WPEBLS
m-c	97.2+*	96.9+*	96.9+*	73.7
m-c-a	97.6+*	96.9+*	95.9	96.1
m-c-b	96.6+*	96.1	96.1	75.6
m-d	96.2	95.6	95.8	96.1
m-d-i	96.1	95.3	96.6+*	95.6
m-d-c-a	95.6	94.4	94.5	95.0
m-d-i-c-a	95.1	94.4	94.2	95.0
vote	97.0+*			
freq	97.0+*			
vote-b	96.1			

Base learners: BAYES, CART & WPEBLS				
	Meta-learner			
Scheme	BAYES	ID3	CART	WPEBLS
m-c	97.0+*	96.6+*	95.3	73.7
m-c-a	97.2+*	96.4	95.3	96.2
m-c-b	96.1	96.2	96.2	76.2
m-d	96.7+*	95.0	96.2	96.2
m-d-i	96.6+*	95.0	95.8	96.2
m-d-c-a	94.8	94.2	94.7	94.5
m-d-i-c-a	94.7	94.2	94.5	94.8
vote	97.0+*			
freq	96.9+*			
vote-b	96.2			

Table 3: Prediction accuracy of single-strategy classifiers (%)

Data Set/Algorithm	BAYES	ID3	CART	WPEBLS
Secondary Structure (SS)	62.1	55.4	50.8	48.1
Splice Junction (SJ)	96.4	93.9	94.8	94.4

various strategies in both data sets. Similarly, better results were achieved for various combinations of different strategies and meta-learners across all base learners in the SJ data set. Improvements on the already high accuracy obtained from the base learners in the SJ data set reflects the viability of the meta-learning approach.

As mentioned in the previous section, the combiner schemes generally performed more effectively than the arbiter or hybrid schemes. This suggests that coalescing the results is more beneficial than arbitrating among them. In addition, the training set for the combiner strategy includes examples derived from the entire original training set, whereas the one for the arbiter or hybrid strategy includes only examples chosen by a selection rule from the original set. That is, the training set for the arbiter or hybrid strategy is usually smaller than the one for the combiner strategy and hence contains less information. (This crucial fact may not be exhibited in larger learning tasks with massive amounts of data.)

Among the combiner schemes, the *meta-class-attribute* scheme generally performed more effectively than the others. This might be due to the additional information (attribute vectors) present in the training examples, suggesting that information from the predictions alone is not sufficient to achieve higher prediction accuracy. To our surprise, the *meta-class-binary* scheme did not perform more effectively than the *meta-class* scheme. We postulated that more specialized binary classifiers would provide more precise information for the meta-learner. However, that was not the case in our experiments.

We also postulated that a probabilistic learner like BAYES would be a more effective meta-learner due to the relatively low regularity in the training data for meta-learners and its probabilistic means of combining evidence. Our empirical results indeed show that BAYES is a more effective meta-learner.

An anonymous reviewer of another paper proposed an “optimal" formula based on Bayes Theorem to combine the results of classifiers, namely, $P(x) = \sum_c P(c) \times P(x|c)$, where x is a prediction and c is a classifier. $P(c)$ is the prior which represents how likely classifier c is the true model and $P(x|c)$ represents the probability classifier c guesses x. Therefore, $P(x)$ represents the combined probability of prediction x to be the correct answer. Unfortunately, to be optimal, Bayes Theorem requires the priors $P(c)$'s to be known, which are usually not, and it also requires the summation to be over all possible classifiers, which is almost impossible to achieve. However, an approximate $P(x)$ can still be calculated by approximating the priors using various established techniques on the training data and using only the classifiers available. This technique is essentially a "weighted voting scheme" and does not consider possible relationships among predictions generated by the classifiers. This and the aforementioned strategies and issues are the subject matter of ongoing experimentation.

Hypothesis boosting using only a single strategy has recently attracted attention in the theoretical learning community. The pioneering work in this area is due to Schapire [17]. Based on an initial learned hypothesis for some concept derived from a random distribution of training data, Schapire's scheme iteratively generates two additional distributions of examples. The first newly derived distribution includes randomly chosen training examples that are equally likely to be correctly or incorrectly classified by the first learned classifier. A new classifier is formed from this distribution. Finally, a third distribution is formed from the training examples on which both of the first two classifiers disagree. A third classifier (in effect, an arbiter) is computed for this distribution. The predictions of the three learned classifiers are combined using a simple arbitration rule similar to the one of the rules we presented above. Schapire rigorously proves that the overall accuracy is higher than the one achieved by simply applying the learning algorithm to the initial distribution under the PAC learning model [21]. In fact, he shows that arbitrarily high accuracy can be achieved by recursively applying the same procedure. Although his approach is limited to the PAC model of learning, some success was achieved in the domain of character recognition, using neural networks [9]. Freund [11] has a similar approach, but with potentially many more distributions.

In addition to applying meta-learning to coalescing results from multiple different algorithms applied to the same set of data, meta-learning can also be used to combine results from a set of parallel or distributed learning processes to improve speed. Much of the research in inductive learning concentrates on problems with relatively small amounts of data. With the coming age of very large network computing, it is likely that orders of magnitude more data in databases will be available for various learning problems of real world importance. The Human Genome Project [8] and Grand Challenges of HPCC [22] are perhaps the best examples of large scale efforts demanding efficient knowledge discovery systems. Parallel and distributed processing would substantially increase the speed of processing data and hence the amount of data a knowledge discovery system can handle effectively. Our approach is to apply learning processes to disjoint subsets of the original training set concurrently (a *data reduction* technique) and the results from the processes are then combined through meta-learning. Preliminary results reported in [5] are encouraging. Since the ultimate goal of this work is to improve both the accuracy and efficiency of machine learning, we have been working on combining ideas in multistrategy hypothesis boosting, described in this paper, with those in parallel learning. We call this approach *multistrategy parallel learning*. Preliminary results reported in [4] are encouraging. To our knowledge, not much work in this direction has been attempted by others.

7 Concluding Remarks

The schemes presented here constitute a step toward the multistrategy parallel learning approach and the preliminary results obtained so far indicates certain meta-learning strategies and algorithm for meta-learning are more effective than others. More experiments are being performed to ensure that the results we have achieved to date are indeed statistically significant, and to study how meta-learning scales with much larger data sets. We intend to further explore the diversity and possible "symbiotic" effects of multiple learners to improve our meta-learning schemes in a parallel and distributed environment.

Acknowledgements

This work has been partially supported by grants from New York State Science and Technology Foundation, Citicorp, and NSF CISE. We thank David Wolpert for many useful and insightful discussions that substantially improved the ideas presented in this paper. We also thank Pat Langley and the anonymous reviewers for comments on improving the presentation of this paper.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, Belmont, CA, 1984.

[2] W. Buntine and R. Caruana. *Introduction to IND and Recursive Partitioning*. NASA Ames Research Center, 1991.

[3] P. Chan and S. Stolfo. Meta-learning for multistrategy and parallel learning. In *Proc. Second Intl. Work. on Multistrategy Learning*, pages 150–165, 1993.

[4] P. Chan and S. Stolfo. Toward multistrategy parallel and distributed learning in sequence analysis. In *Proc. First Intl. Conf. Intel. Sys. Mol. Biol.*, pages 65–73, 1993.

[5] P. Chan and S. Stolfo. Toward parallel and distributed learning by meta-learning. In *Working Notes AAAI Work. Know. Disc. Databases*, pages 227–240, 1993.

[6] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–285, 1987.

[7] S. Cost and S. Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78, 1993.

[8] C. DeLisi. The human genome project. *American Scientist*, 76:488–493, 1988.

[9] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *Intl. J. Pat. Recog. Art. Intel.*, 1993. To appear.

[10] N. Flann and T. Dietterich. A study of explanation-based mehtods for inductive learning. *Machine Learning*, 4:187–266, 1989.

[11] Y. Freund. Boosting a weak learning algorithm by majority. In *Proc. 3rd Work. Comp. Learning Theory*, pages 202–216, 1990.

[12] C. Matheus, P. Chan, and G. Piatesky-Shapiro. Systems for knowledge discovery in databases. *IEEE Trans. Know. Data. Eng.*, 1993. To appear.

[13] T. M. Mitchell. The need for biases in learning generalizaions. Technical Report CBM-TR-117, Dept. Comp. Sci., Rutgers Univ., 1980.

[14] M. Noordewier, G. Towell, and J. Shavlik. Training knowledge-based neural networks to recognize genes in DNA sequences. In *Proc. NIPS-91*, pages 530–536, 1991.

[15] N. Qian and T. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *J. Mol. Biol.*, 202:865–884, 1988.

[16] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[17] R. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–226, 1990.

[18] B. Silver, W. Frawley, G. Iba, J. Vittal, and K. Bradford. ILS: A framework for multi-paradigmatic learning. In *Proc. Seventh Intl. Conf. Machine Learning*, pages 348–356, 1990.

[19] S. Stolfo, Z. Galil, K. McKeown, and R. Mills. Speech recognition in parallel. In *Proc. Speech Nat. Lang. Work.*, pages 353–373. DARPA, 1989.

[20] G. Towell, J. Shavlik, and M. Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *Proc. AAAI-90*, pages 861–866, 1990.

[21] L. Valiant. A theory of the learnable. *Comm. ACM*, 27:1134–1142, 1984.

[22] B. Wah et al. High performance computing and communications for grand challenge applications: Computer vision, speech and natural language processing, and artificial intelligence. *IEEE Trans. Know. Data. Eng.*, 5(1):138–154, 1993.

[23] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[24] X. Zhang, J. Mesirov, and D. Waltz. A hybrid system for protein secondary structure prediction. *J. Mol. Biol.*, 225:1049–1063, 1992.