A Comparative Evaluation of Combiner and Stacked Generalization

David W. Fan, Philip K. Chan and Salvatore J. Stolfo

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

Abstract

Combiner and *Stacked Generalization* are two very similar *meta-learning* methods that combine predictions of multiple classifiers to improve accuracy of any single classifier. In this paper, we compare stacked generalization and combiner from the perspective of training efficiency versus accuracy. We show that both methods improve the accuracy of any single classifier roughly at an equivalent level. Moreover, we also see that the cost of stacked generalization is very large and may prevent it from being used on very large data sets.

1 Introduction

Combiner (Chan & Stolfo 1993a) and *stacked generalization* (Wolpert 1992) are two similar *meta-learning* methods. Meta-learning (Chan & Stolfo 1993a) can be loosely defined as learning from information generated by a set of learners. It can also be viewed as the learning of meta-knowledge on the learned information. In meta-learning, the focus is on learning from the output of inductive learning 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, in meta-learning we are interested in the output of the learners, not the learners themselves. Meta-learning is a general technique to coalesce the results of multiple learners. Combiner and stacked generalization are specifically proposed techniques to combine different learners to improve prediction accuracy. They involve the application of multiple algorithms on the same set of data and the results of the learned concepts are combined by meta-learning. This process is called *multistrategy hypothesis boosting* in (Chan & Stolfo 1993a). The goal is to achieve an overall accuracy that is higher than the accuracy obtained by any of the individual learning algorithms. In addition to combiner and stacked generalization, other meta-learning strategies are discussed in (Chan & Stolfo 1993a; 1993b; 1993c), including learning in distributed and parallel contexts.

This paper focuses on combiner and stacked-generalization and a comparative evaluation of training efficiency versus accuracy. In Section 2, we compare the differences between stacked generalization and combiner. Sections 3 and 4 present our preliminary experimental results that show there is essentially no difference in accuracy between them, but there is a big difference in training efficiency. Section 5 presents our findings and work in progress.

2 Combiner vs. Stacked Generalization

The objective of both combiner and stacked-generalization is to improve the overall predictive 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 learners, as depicted in Figure 1. Each of the base learners is provided with the set of raw training data. However, the training set for the meta-learner (meta-level training data or meta data) varies according to different strategies. Each base-learner generates a *base classifier* and the meta-learner generates a *meta-classifier*. The meta-learner does not aim at picking the "best" base classifier; instead it tries to combine the predictions of the classifiers by learning their biases. That is, the prediction accuracy of the overall system is not limited to the most accurate base classifier. The goal is to generate an overall system that outperforms the underlying base classifiers by learning how they correlate with each other. In both combiner and stacked generalization, the meta-learner combines the predictions of the learned classifiers (by the base learners), as shown in Figure 2. The predictions of the learned base classifiers on the training

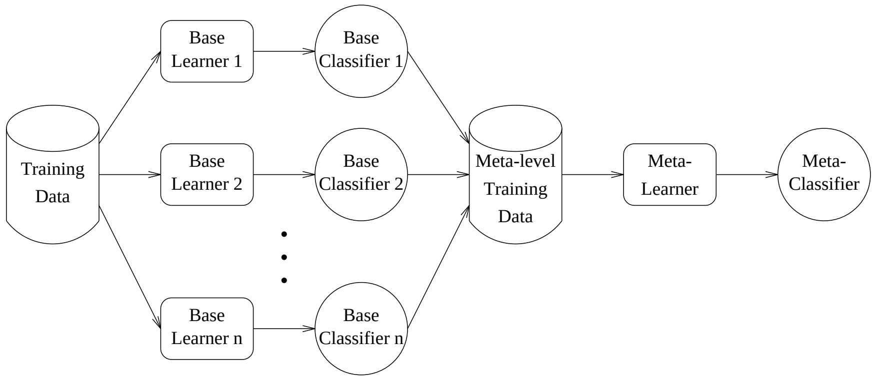

Figure 1: Meta-Learning

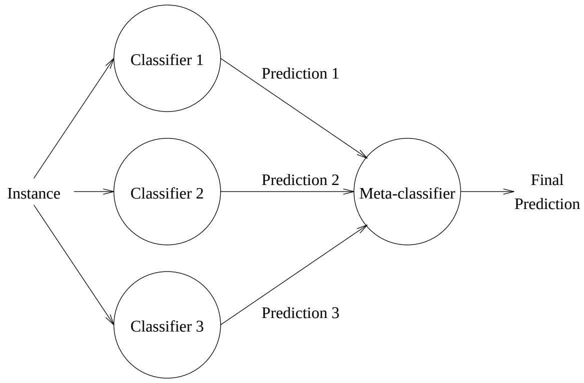

Figure 2: Classification in the *combiner* and *stacked generalization* strategy

set form the basis of the meta-learner's training set. A *composition rule* determines the content of training examples for the meta-learner. From these examples, the meta-learner generates a meta-classifier. 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 meta-classifier.

In combiner, we partition the training set T into two parts, T_1 and T_2. T_1 is half of T, and T_2 is the remaining half of T. Each partition preserves the original distribution of class labels as in T. Then we form two pairs, (T_1, T_2) and (T_2, T_1). We apply the learning algorithms (those chosen to generate the base classifiers) on T_1, and apply the following scheme [1] on every item of the unseen T_2 with the learned classifiers to form half of the meta-learner training data. Similarly, we apply the base level learning algorithms on T_2 and apply the same scheme on every item of T_1 to form the other half of the meta-learner training data. Finally, base classifiers are generated on the whole set T.

[1] In another paper (Chan & Stolfo 1993a), other composition schemes are addressed.

- Return a meta-level training instance with the correct classification and the predictions; i.e., $(class(x), C_1(x), C_2(x), C_3(x))$, where x is an item on which this scheme is applied, C_i is a learned classifier, $C_i(x)$ is its prediction on item x, $class(x)$ is the correct classification of x. (For further reference, this scheme is denoted as *class-combiner*.) A sample meta-level training set obtained by applying this scheme is displayed in Figure 3.

The scheme for the composition rule is defined in the context of forming a training set for the meta-classifier. It is also used in a similar manner during classification after a meta-classifier has been computed. Given an instance whose classification is sought, we first compute the classifications predicted by each of the base classifiers (generated on the whole set T). The composition rule is then applied to generate a single meta-level instance, which is then classified by the meta-classifier to produce the final predicted class of the original test datum.

In stacked-generalization, the training set T (of size n) is made into a set of n pairs T'={ (x_1, y_1), (x_2, y_2), ..., (x_n, y_n) }, where each x_i is a single element in T and y_i is the remaining n-1 elements in T (excluding x_i). Each x_i is unique, so they completely cover T. This is called CVPS- partitioning (Wolpert 1992). We train the base learning algorithms on y_i, and apply the above scheme (*class-combiner*) on the left-out x_i with the learned classifiers to form one meta-level training data item. This process is repeated n times. Finally, like in combiners, we generate the same base classifiers on all of T. The way to classify a data item is the same as that of combiner.

It is important to note the connection between combiner and stacked-generalization. Generally speaking, to form the meta-learner training set, we partition the original training set T into k disjoint subsets of equal size or at most differ by 1, p_i (i=1...k). Each part preserves the original distribution of class labels as in T. We then form k pairs, (x_i, y_i), where x_i is p_i, and y_i is the remaining data in T excluding x_i (or $T - x_i$). To

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 *class-combiner* scheme	
Class	Attribute vector
`table`	(`table`, `table`, `table`)
`chair`	(`table`, `chair`, `lamp`)
`lamp`	(`lamp`, `chair`, `chair`)

Figure 3: Sample training sets generated by the *combiner* and *stacked generalization*

obtain the meta-learner training data, we apply the learning algorithms to y_i of each pair and apply the *class-combiner* scheme on every items in corresponding unseen x_i to form meta-learner data. This process will be carried out for a total of k times. In combiner, k is equal to 2; and in stacked generalization, k is equal to n or the size of T. We call this process k-fold meta-learning.

It will be very interesting to see how this k will affect the overall accuracy boost and overall cost. The predictions of x_i (these predictions are included in meta-data) is to reflect the behavior of base classifiers trained on the whole T. The bigger the k, the more data each base learning algorithm will be trained on, the classifiers they generate should more closely reflect the behavior of the classifiers trained on the entire data set. Hence, the meta-data will more closely show the correlation of the predictions of base classifiers. It may be intuitive to guess that the bigger the k, the more accurate a final classifier system will be. But our result shows that this is not necessarily the case. It should be very obvious that the cost to carry out meta-learning will increase rapidly with the increase of k. We will see in Section 4 how much this cost will increase from k=2 to k=n.

3 Experiments

Three inductive learning algorithms were used in our experiments. We obtained ID3 (Quinlan 1986) and CART (Breiman *et al.* 1984) as part of the IND package (Buntine & Caruana 1991) from NASA Ames Research Center; both algorithms compute decision trees. BAYES is a Bayesian classifier that is based on computing conditional probabilities (frequency distributions), which is described in (Clark & Niblett 1987). BAYES was re-implemented in C.

Two molecular biology sequence analysis data sets, obtained from the UCI Machine Learning Database, were used in our studies. They are of different sizes and difficulties. The secondary protein structure data set (SS) (Qian & Sejnowski 1988), 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 (Qian & Sejnowski 1988)) in the data. The amino acid sequences were split into shorter sequences of length 13 according to a windowing technique used in (Qian & Sejnowski 1988). The sequences were then divided into a disjoint training and test set, according to the distribution described in (Qian & Sejnowski 1988). The training set, T, for this task has 17300 instances and the test set has 4325. The prediction accuracy of most of the learning algorithms today on SS is around 50%, so there is ample room for improvement. The DNA splice junction data set (SJ) (Noordewier, Towell, & Shavlik 1991), 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, T, for this task and the rest, 638 sequences, served as the test set. The prediction accuracy on splice junction is around 90%.

We have applied 5-fold cross validation on combiner and stacked generalization. Different combinations of three base and one meta-learner are conducted on the two data sets and the results are shown in the following tables (1 through 4).

4 Results

There are several ways to look at the results. First, we see if the employment of both combiner and stacked generalization improves prediction accuracy with respect to the underlying 3 base classifiers. As shown in Table 1 , the accuracy of single classifiers (ID3 and CART) on secondary structure is around 57%. The accuracy of both combiner and stacked generalization is about 61%. That is an increase of 4% or 173 (out of 4325) more examples correctly classified. In the DNA splice junction data set (also shown in Table 1), the increase of combiner and stacked generalization (with average accuracy of around 95.5%) over single classifiers (ID3 and CART, average accuracy is around 92%) is 3.5%. These increases are

Single Classifier Accuracy on SS			
	Learner		
Fold	ID3(%)	CART(%)	BAYES(%)
1	56.1	57.6	60.0
2	52.9	57.1	62.5
3	54.1	57.2	62.7
4	52.2	57.1	60.8
5	56.6	56.9	61.4
μ	56.4	57.2	61.5
σ	1.9	0.3	1.2

Single Classifier Accuracy on SJ			
	Learner		
Fold	ID3(%)	CART(%)	BAYES(%)
1	90.1	93.4	95.1
2	91.2	84.8	96.9
3	90.4	94.0	95.3
4	89.7	94.5	95.3
5	88.4	93.9	94.5
μ	90.0	94.1	95.4
σ	1.0	0.6	0.9

Results on SS						
Base learners: ID3, CART & BAYES						
	Meta-learner in Stacked Generalization			Meta-learner in Combiner		
accuracy in percentage(%)						
Fold	ID3(%)	CART(%)	BAYES(%)	ID3(%)	CART(%)	BAYES(%)
1	59.6	60.0	61.4	60.0	60.0	61.1
2	62.3	59.7	61.3	62.6	62.6	61.5
3	62.7	62.7	62.5	62.8	62.7	62.1
4	60.3	60.3	60.0	60.7	60.8	60.8
5	59.7	60.1	59.2	60.9	60.9	60.7
μ	60.9	60.6	60.9	61.4	61.4	61.2
σ	1.5	1.2	1.3	1.2	1.2	0.6

Results on SJ						
Base learners: ID3, CART & BAYES						
	Meta-learner in Stacked Generalization			Meta-learner in Combiner		
accuracy in percentage (%)						
Fold	ID3(%)	CART(%)	BAYES(%)	ID3(%)	CART(%)	BAYES(%)
1	95.6	95.8	94.8	95.0	95.1	94.8
2	96.1	96.4	95.8	96.4	96.9	95.6
3	95.5	95.5	96.2	95.8	95.8	96.4
4	96.7	96.2	96.2	96.6	96.2	96.1
5	95.0	95.6	93.9	95.1	95.6	93.9
μ	95.8	95.9	95.4	95.8	95.9	95.4
σ	0.7	0.4	1.0	0.7	0.7	1.0

Table 1: Summary of prediction accuracy for Secondary Structure and Splice Junction

significant. We also notice that in both SS and SJ datasets, the best single classifier is BAYES. The accuracy of both combiner and stacked generalization is close to that of BAYES. For the two particular data sets under study and the particular combination of learning algorithms, we could pick BAYES instead of using meta-learning to be the best learner. But on the average, combiner and stacked generalization (and others in the literature) are more accurate than any single classifiers (refer results in (Chan & Stolfo 1993a; Wolpert 1992)).

Second, we see which of combiner and stacked generalization has a higher accuracy improvement. The two methods are comparable, but combiner performs a little better. On secondary structure data sets, combiner has an average of 0.5% higher accuracy than stacked generalization , but this is within 1% standard deviation, so their accuracy are essentially the same. On splice junction, their performance are nearly the same.

Third, we examine the correlation between these two methods. We have applied a simple approach to determine: how many examples they both correctly label; how many examples one method can label correctly, but the other cannot; how many examples they both give the same wrong answer; and how many examples that they both give wrong but different answers. Correlation analysis results are shown in Table 2. The results show that the two methods are very close to each other. In most of the cases, they either both give the correct answer or both give the same wrong answer, indicating they both have effectively learned the same knowledge.

Fourth, we display the cost of both methods in Table 3. The difference in their time cost is huge. On the SS dataset, while combiner spent no more than 7 minutes to learn, stacked generalization spent 9 days. On SJ dataset, it took combiner half a minute to learn, but took stacked generalization nearly 6 hours and a half. There are orders of magnitude difference in performance for comparable accuracy gain.

Finally, we look into the meta-data itself. Each piece of meta-data is composed of the prediction of

Results on SS																	
Meta-learner: ID3						Meta-learner: CART						Meta-learner: BAYES					
Fold	SC	CI	IC	SI	DI	Fold	SC	CI	IC	SI	DI	Fold	SC	CI	IC	SI	DI
1	2560	17	35	1699	14	1	2595	0	0	1730	0	1	2402	252	241	1336	94
2	2677	18	30	1594	6	2	2397	185	308	1325	110	2	2452	201	206	1394	72
3	2619	94	95	1493	24	3	2713	0	0	1612	0	3	2407	295	278	1253	92
4	2310	298	314	1280	123	4	2533	77	97	1579	39	4	2309	288	321	1270	137
5	2376	205	256	1395	93	5	2460	141	172	1486	66	5	2277	283	349	1272	144

Results on SJ																	
Meta-learner: ID3						Meta-learner: CART						Meta-learner: BAYES					
Fold	SC	CI	IC	SI	DI	Fold	SC	CI	IC	SI	DI	Fold	SC	CI	IC	SI	DI
1	604	6	2	24	2	1	604	7	3	24	0	1	605	0	0	33	0
2	610	3	5	20	0	2	611	4	7	16	0	2	610	1	0	27	0
3	605	4	6	22	1	3	605	4	6	23	0	3	614	0	1	22	1
4	610	7	6	15	0	4	612	2	2	22	0	4	613	1	0	24	0
5	595	11	12	20	0	5	609	1	1	27	0	5	599	0	0	39	0

Note:

- SC: Same Correct, or both of stacked generalization and combiner predict correctly.
  CI: Correct Incorrect, or stacked generalization correctly predicts but combiner incorrectly labels
  IC: Incorrect Correct, or stacked generalization incorrectly labels, while combiner correctly labels
  SI: Same Incorrect, or stacked generalization and combiner make the same wrong labels
  DI: Different Incorrect, both are incorrect, but their answers are different
- Results are shown in number of predictions on five test sets.

Table 2: Summary of correlation analysis between combiner and stacked generalization for Secondary Structure and Splice Junction

C_i (trained from y_i) on x_i (see Section 2), which we call meta component data (refer to Table 4). In our experiment, the meta-data is composed of predictions by classifiers (generated by ID3, CART and BAYES) on unseen x_i. The predictions by these classifiers are the meta-component data. The accuracy of meta-component data is measured by comparing the correct labels of x_i against the predictions by these classifiers. The closer the accuracy of meta-component data is to the accuracy of a single classifier, the more accurate it simulates the behavior of the base classifier. The accuracy of the meta-component data of stacked generalization should be closer to the accuracy of single classifiers, because in predicting a data element, the classifier is trained from the remaining n-1 elements. By comparing Table 4 with the single classifier accuracy in Table 1 ,we can see that the accuracy of component data of stacked generalization is closer to the accuracy of a single classifier than that of combiner, which means it closely mimics the behavior of single classifiers. The accuracy of the stacked generalization's meta-component and the accuracy of single classifier differs by no more than 1%, but the accuracy of the combiner's meta-component and the accuracy of single classifier differs by as much as 5%. However, this seems not to boost the accuracy in stacked generalization more than in the combiner. This is an interesting finding.

5 Discussion

As pointed out in section 2, it may be intuitive to guess that the accuracy boost of stacked-generalization will be more than that of combiner. But the results do not support this. There may be 3 possible explanations for this, all requiring further study:

1. One argument is that the correlation of meta component data actually decides the overall accuracy boost. The accuracy of meta component data to reflect the behavior of the base classifiers may not be the decisive factor. We performed correlation analysis at the meta-classifier level; the same analysis can be performed at the base-classifier level.
2. Another suggestion is that the complexity (or the difficulty to learn) of the meta data may also be important. The meta data of stacked generalization may actually reflect the relationship of the predictions of the base classifiers, but the relationship is very subtle and very difficult for the meta learner to learn effectively, so the overall accuracy is not increased as much as we would hope. Specialized learning algorithms may be developed for the sole purpose of learning how to combine classifiers.
3. Conflicts in the meta data may introduce the problem. As an example, look at Figure 4. In the first example, there is a data element that ID3 labels 1, CART labels 2, BAYES labels 0, but the actual answer is 1. In the second case, ID3, CART

Results on SS			
	Combiner		
Fold	ID3	CART	BAYES
1	336.9	340.4	336.8
2	340.7	344.4	340.7
3	346.1	349.6	346.3
4	340.3	344.5	340.3
5	351.0	354.7	351.1
μ	343.0	346.7	343.0

Results for SJ						
	Stacked Generalization			Combiner		
Fold	ID3	CART	BAYES	ID3	CART	BAYES
1	23720.4	+0.50	-0.16	30.6	30.93	29.9
2	23865.7	+0.50	-0.16	29.8	30.3	30.0
3	23442.9	+0.50	-0.16	31.6	31.2	30.1
4	23784.3	+0.50	-0.16	30.5	30.2	30.0
5	23805.9	+0.50	-0.16	29.9	30.7	29.8
μ	23723.8	23724.3	23723.7	30.5	30.7	30.0

Note:

- The timing cost of stacked generalization on SS data set is huge, we did not have exclusive use of the machines to measure it. The method we used to estimate its cost is to measure the time used to generate 1000 meta data. The estimate we give below is the result made by multiplying the actual time for 1000 items by 17.3 (17300/1000=17.3) plus the time to learn the 3 base classifiers and meta classifier. Full Stacking Timing cost on SS= 771760 seconds (nearly 9 days)
- For the SJ data set, We couldn't measure the time cost of stacked generalization for different meta-learner (ID3, CART and BAYES) from start, that would require a lot of computer resources. We accurately measured the time cost of stacked generalization with ID3 as meta-learner from start, estimated the case for CART as meta-learner by adjusting the difference ID3 versus CART to learn the meta-classifier. The difference was that it took 0.50 seconds more for CART to learn the meta-data than ID3. We estimated the the time cost of using BAYES as meta-learner with the same approach.
- Also, only CPU time of learning is counted.

Table 3: Time cost of combiner and stacked generalization on Secondary Structure and Splice Junction (seconds)

and BAYES give the same predictions as in the first example, but this time the true answer is 2. This means that the mapping from attribute vector to label is not one to one, or not a function mapping. This will be very hard for any learner to learn any correct answer, even a human may not be able to do it correctly. This kind of example represents conflict in training data. Decision tree algorithms, like ID3 and CART are very sensitive to conflicts, BAYES is better equiped to shield against it. We do need to see how much conflict there are in the meta data of both combiner and stacked generalization. One approach that can reduce (but may not eliminate) the number of conflicts is to introduce more base learners, with the hope that all the base classifiers will not make the same predictions on different data. This method is not applicable if the number of base classifiers cannot be increased. Another approach is to include the attribute vector of the data itself in the meta data. Different data will have different attribute vectors[2], so there will be no conflict in the meta-data. This approach is called *class-attribute-combiner*. A detailed discussion about it can be found in (Chan & Stolfo 1993a). One advantage of this method is that it will not require more base classifiers. However, this approach will on the one hand add more to the complexity of meta data, on the other hand, it may reduce the effect of meta-learning by these extra attributes. We will see which method works best in future studies.

[2] We assume that the original training data is conflict-free.

Correct Class	ID3	CART	BAYES
1	1	2	0
2	1	2	0

Figure 4: Conflict in Meta-data

In section 2, we introduced the idea of k-fold meta-learning. The result shows that the accuracy boost of k=2 (combiner) and k=n (stacked generalization) are similar. So is k arbitrary or is there a k that will lead to significant maximal acurracy gain? How much is this gain? Do we pay for what we get? If the accuracy boost with different k is almost equivalent, k=2 or combiner is obviously the best choice, accurate and cheap. Breiman (Breiman 1992) did some experiments in evaluating the performance of different k values in regression estimators and found $k = 10$ achieves comparable accuracy as n-fold. More extensive and systematic experiments on real data sets would yield better understanding of how the different values of k behave.

The correlation analysis between combiner and stacked generalization shows that they have acquired very similar knowledge. We can study those examples that both of them generalization give the same false answer and perhaps develop a new composition rule that can learn them well.

6 Concluding Remarks

The finding of the experimental comparison between combiner and stacked generalization shows that both

Results on SS						
	Base learner in Stacked Generalization			Base learner in Combiner		
Fold	ID3(%)	CART(%)	BAYES(%)	ID3(%)	CART(%)	BAYES(%)
1	54.0	56.9	62.6	49.3	55.3	61.4
2	55.3	56.8	62.3	49.8	55.5	60.2
3	53.7	57.5	61.9	50.0	56.7	60.2
4	55.9	57.5	62.2	52.4	55.5	61.2
5	57.3	57.3	62.2	53.2	55.5	61.0
μ	55.2	57.9	62.2	50.9	55.7	60.8
σ	1.5	0.5	0.3	1.74	0.6	0.6

Results on SJ						
	Base learner in Stacked Generalization			Base learner in Combiner		
Fold	ID3(%)	CART(%)	BAYES(%)	ID3(%)	CART(%)	BAYES(%)
1	90.3	94.2	95.5	88.6	94.4	95.2
2	90.8	94.3	95.2	86.2	91.3	94.4
3	91.5	94.3	95.7	87.5	93.5	94.8
4	90.8	94.1	95.4	88.4	94.1	95.7
5	91.7	94.4	95.8	88.7	94.4	95.4
μ	91.0	94.3	95.5	87.9	93.5	95.1
σ	0.6	0.1	0.2	1.05	1.3	0.5

Table 4: Summary of accuracy of meta component data for Secondary Structure and Splice Junction(%)

methods obviously boost the accuracy with respect to that of the base classifiers. Their accuracy boost are at roughly the same level. The cost of combiner (where k=2) is moderately small, but the cost of stacked generalization (where k=n) is huge that may prevent it from being applied in realistic databases except for small ones. In consideration of both accuracy and cost, combiner is an obvious better choice.

Acknowledgments

This work has been partially supported by grants from NSF (IRI-94-13847 and CDA-90-24735), NYSSTF, and Citicorp. We thank David Wolpert for many insightful discussions about stacked generalization and meta-learning. The comments from the reviewers helped us to improve this paper.

References

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

Breiman, L. 1992. Stacked regressions. Technical Report 367, Department of Statistics, University of California, Berkeley.

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

Chan, P., and Stolfo, S. 1993a. Experiments on multistrategy learning by meta-learning. In *Proc. Second Intl. Conf. Info. Know. Manag.*, 314–323.

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

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

Clark, P., and Niblett, T. 1987. The cn2 induction algorithm. *Machine Learning* 3:261–285.

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

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

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

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