Empirical Study of a Stacking State-space 

Agapito Ledezma, Ricardo Aler and Daniel Borrajo
Universidad Carlos III de Madrid
Avda. de la Universidad, 30
28911 Leganés. Madrid (Spain)
Email: ledezma@inf.uc3m.es, aler@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract

Nowadays, there is no doubt that machine learning techniques can be successfully applied to data mining tasks. Currently, the combination of several classifiers is one of the most active fields within inductive machine learning. Examples of such techniques are boosting, bagging and stacking. From these three techniques, stacking is perhaps the less used one. One of the main reasons for this relates to the difficulty to define and parameterize its components: selecting which combination of base classifiers to use, and which classifier to use as the meta-classifier. One could use for that purpose simple search methods (e.g. hill climbing), or more complex ones (e.g. genetic algorithms). But before search is attempted, it is important to know the properties of the search space itself. In this paper we study exhaustively the space of Stacking systems that can be built by using four base learning systems: C4.5, IB1, Naive Bayes, and PART. The results that have been obtained in this paper will be useful for designing new Stacking-based algorithms and tools.

1 Introduction

Nowadays, there is no doubt that machine learning techniques can be successfully applied to data mining tasks. A particularly successful approach is combine basic classifiers to improve accuracy. The most important basic systems that have been proposed are bagging [4], boosting [9], and stacking [19]. Bagging uses majority vote to combine several classifiers obtained from different subsets of the data set. Boosting sequentially learns several classifiers, each focusing on the data that was misclassified by the previous classifier. All the classifiers are combined by weighted vote. Both bagging and boosting use the same learning algorithm to generate the ensemble of classifiers. Stacking learns how to combine the outputs of a set of classifiers that have been obtained by different learning algorithms. There are also many variants that are becoming increasingly sophisticated, such as LPboosting [2] in the boosting subfield. There are also many variants of the basic stacking algorithm [5, 6, 12, 14, 17].

The main problem of stacking and any AI tool that needs to use it, is how to obtain the right combination of base classifiers and the meta-classifier. If the number of classifiers and algorithms to use is small, this problem can be solved by a simple method in a reasonable time: exhaustive search. For instance, if the goal is to build a stacking system made of three base classifiers and the meta-classifier, and there are four available learning algorithms, then only 16 stacking combinations need to be tested. If more classifiers are needed, then sampling techniques or heuristic search could be used instead of exhaustive search, in the same spirit as the wrapper approaches for attribute selection [11].

However, before search is used as the core of automatic configuration of stacking systems, it is important to know the properties of the state-space of stacking systems. In particular, it would be very useful to know the density of "good" stacking systems in these spaces. We also want to empirically test the following hypothesis. In principle, the stacking meta-classifier can determine which base classifiers to take into account to reach the final decision (the base classifier outputs are the inputs to the meta-classifier), much in the same way as any learning algorithm can determine that some of its attributes are irrelevant (by not using them in the final hypothesis). If this is the case, using $n-1$ base classifiers should make no difference to using n classifiers, as the meta-classifier would learn that one of its n classifiers is irrelevant. We explore this issue in detail in the experimental evaluation section.

In this paper, we carry out an exhaustive study on the state-space of stacking systems with two, three, and four base classifiers that have been chosen from four well-known algorithms: C4.5 [16], IB1 [1], Naive Bayes [10], and PART [8]. We could have chosen many other very useful learning algorithms, such as neural networks. However, each experiment is very time consuming, and we have to

bound the number of classifiers to be used. Also, we have used the WEKA tool [18] that did not incorporate until very recently this type of classifiers. Since the results might be dependent on the set of chosen classifiers, we will explore in the future the effect of introducing other types of learning algorithms. We believe that our results will be useful

This paper is organized as follows. Section 2 gives some background on stacking and explains how to explore the state-space of stacking systems. Section 3 describes the experimental setup and the experimental results, respectively. Finally, Section 4 discusses those results, and Section 5 draws some conclusions.

2 Stacking

Stacking is the abbreviation to refer to Stacked Generalization [19]. The main idea of stacking is to combine classifiers from different learners such as decision trees, instance-based, bayesian or rule-based learners. Since each one uses different knowledge representation and different learning biases, the hypothesis space will be explored differently, and different classifiers will be obtained. Thus, it is expected that their errors will not be correlated, and that the combination of classifiers will perform better than the base classifiers.

Once the classifiers have been generated, they must be combined. Stacking uses the concept of meta learner. The meta learner (or level-1 model) tries to learn how the decisions of the base classifiers (or level-0 models) should be combined to obtain the final classification. More formally, given a data set S, stacking first generates a subset of training sets $S_1, ..., S_T$ and then follows something similar to a cross-validation process: it leaves one of the subsets out (e.g. S_j) to use later. The remaining instances $S - S_j$ are used to generate the level-0 classifiers by applying K different learning algorithms, $k = 1, ..., K$, to obtain K classifiers. After the level-0 models have been generated, the S_j set is used to make the training set for the meta learner (level-1 classifier). Level-1 training data is built from the predictions of the level-0 models over the instances in S_j. Level-1 data has K attributes, whose values are the predictions of each one of the K level-0 classifiers for every instance in S_j, and the target class; i.e. the right class for every particular instance in S_j. Once the level-1 data has been built from all instances in S after the internal cross-validation process, any learning algorithm can be used to generate the level-1 model. To complete the process, the level-0 models are re-generated from the whole data set S (this way, it is expected that classifiers will be slightly more accurate). To classify a new instance, the level-0 models produce a vector of predictions that is the input to the level-1 model, which in turn predicts the class.

There are many ways to apply the general idea of stacked generalization. Ting and Witten [17] use probability outputs from level-0 models instead a of class prediction as inputs to the level-1 model. LeBlanc and Tibshirani [12] analyze the stacked generalization with some regularization (non-negative constraint) to improve the prediction performance on one artificial dataset. Other works on stacked generalization have developed different focus [5, 6, 7]. However, none of them have explored the state-space of classifiers for analyzing the effects of automatically defining the "best" classifiers setup.

One of the main difficulties in applying this technique consists on identifying which learning techniques to use in the 0- and 1-levels. In this paper, the whole state-space of stacking systems with $i = 2$, 3, and 4 base classifiers will be studied. Base classifiers are chosen from a set that contains C4.5, IB1, PART, and Naive Bayes. The 1-level classifier is selected from the same set. Once built, each resulting stacking system is tested with a testing set. In general, if b base classifiers can be chosen from n learning algorithms, the number of stacking systems that can be built is $N = \binom{n}{b} * n$. In this paper, three sets of experiments have been carried out, with $n = 4$, and $b = 2$, $b = 3$, and $b = 4$, resulting in 24, 16, and 4 combinations, respectively. This is the space of stacking systems we are going to explore in this article.

Although the number N of stacking systems in the state-space grows fast, it is interesting to know that the effort to compute all the base classifiers needed to build all the stacking systems is linear. An estimation in terms of the number of examples to be processed follows (this is quite appropriate, as all the learning systems used here are linear in the number of examples processed). The basic algorithm of stacking carries out a training cycle J times. This cycle consists of training all the base classifiers with $\frac{(J-1)}{J}$ of the data, and training the meta-classifier with the remaining $\frac{1}{J}$ of the data. Finally, all the base classifiers are trained again with the whole training set. Even if a base classifier obtained from the same learning algorithm (like C4.5) appears in different stacking systems, it is still the same classifier, and has to be generated just once. In that case, if d is the number of training examples, then the number of examples to be processed to build the base classifiers is $n(\frac{J-1}{J} * J * d + d) = Jnd$. Also, let us suppose that each stacking system is tested using W cross validation. In that case, the number of examples to be processed for training the base classifiers is $WJnd$. This quantity grows linearly in all its parameters.

3 Experiments and results

¿From the many alternatives for inductive techniques, in this work we have used the algorithms implemented in the Waikato Environment for Knowledge Analysis,

WEKA [18]. This software includes all the learning algorithms that we have used to build the base classifiers and an implementation of Stacked Generalization (stacking) that use probability outputs from level-0 models instead a simple class prediction as inputs to the level-1 model [17]. We selected four learning algorithms to build the stacking system:[1]

- C: C4.5 [16]. We used the version that generates decision trees.
- R: PART [8]. It generates a decision list from pruned partial decision trees generated using the C4.5 heuristic.
- N: A probabilistic Naive Bayesian classifier [10].
- I: IB1. Aha's instance based learning algorithm [1].

For the experimental test of the stacking system configuration we have used five data sets from the well known repository of machine learning databases at UCI [3]. This data sets have different sizes and configurations. Table 1 shows the data sets characteristics. In all the experiments we do ten-fold cross-validation. Thus, all results shown in this paper are the average of the cross-validation process.

The results obtained in the first set of experiments are shown in Table 2. In this set of experiments we used two base classifiers, thus obtaining 24 stacking systems by the combination of the four learning algorithms available. The best results in terms of accuracy are given in bold face. And as shown later in Table 5 most best results in Table 2 have more accuracy than the base classifiers (four out of five datasets).

In the second set of experiments we increased the number of base classifiers from two to three, resulting in 16 stacking systems. Table 3 shows the results obtained from this set of experiments. Also, in four of the five best stacking systems, we found the same configuration of level-0 learning systems, Naive Bayes, IB1 and PART.

In the last set of experiments we used four base classifiers. The results obtained from these experiments are shown in Table 4. Except for one domain, the stacking systems with three base classifiers have better results (Table 5) than the stacking systems with four base classifiers. Table 5 also provides results for the four algorithms used as standalone learning algorithms. C4.5-Bagging and C4.5-boosting results are also given for comparison purposes. The number of classifiers in bagging and boosting systems was set to 10 (boosting and bagging of C4.5 with this settings has shown good results in the literature [15]).

[1]For experimental purposes only default setting for all learning algorithms have been used.

4 Discussion

In order to draw conclusions from the experiments performed in Section 3, Table 6 summarizes the best results obtained by each of the three main groups of classifiers used in this paper: base classifiers, stacking combinations, and bagging/boosting (actually, Table 5 shows that boosting is always better than bagging, so only results for boosting are displayed). Also, the difference between the best and worst results in the table is shown in the fourth column.

The data collected in the previous section points to the following conclusions:

1. As it was expected, the best results are always obtained by the ensemble of classifiers systems (either stacking or boosting) against the single inductive learning techniques. Differences between the three kinds of systems are never large (3.34% is the largest, 1.7% on average).

2. In the stacking state-space there are systems that achieve comparable results to boosting: the average accuracy difference between stacking and boosting gives +0.26% in favor of stacking (the same comparison between stacking and the best base classifier results in +1.51%). Moreover, in those domains where stacking does better than boosting, it does so quite frequently. For instance, in the DNA domain, S3/S2 find 10/10 stacking systems that are better than boosting (the proportions are 0.4167/0.625, respectively). Similarly, in the HEART domain, S4/S3/S2 find 2/9/8 stacking systems (out of 4/24/16, respectively) that are better than boosting. Also, in these two domains, the best base classifier does very well (better than boosting, for instance). Probably, stacking is taking advantage of this fact.

3. S3 seems to obtain the best results most frequently (four out of five). Therefore, merely increasing the number of base classifiers does not always pay off in terms of accuracy. Also, the meta-classifier in S4 was unable to determine that one of its base classifiers was not required (it could have done this by just ignoring the appropriate base classifier). Otherwise, in all domains, there would be a S4 system that would always be at least equal to the best of the S3 systems, and this is not the case.

4. Tables 2, 3, and 4 show that Naive Bayes is the best meta-classifier in all domains and all combinations (except in the S2/MUSK case).

The previous analysis refers mainly to the best results only. Cumulative probability graphs, shown in Figures 1, 2, 3, and 4, and to 5, summarize the behavior of the whole

Table 1. Descriptions of the used datasets.

DATASET	ATTRIBUTES	ATTRIBUTES TYPE	INSTANCES	CLASSES
HEART	13	NUMERIC-NOMINAL	303	2
SONAR	60	NUMERIC	208	2
MUSK	166	NUMERIC	476	2
IONOSPHERE	34	NUMERIC	351	2
DNA	60	NOMINAL	3190	3

Table 2. Accuracies rates of stacking systems with two base classifiers (S2).

MC	BC	DNA	SONAR	HEART	MUSK	IONOSPHERE
C	C-R	93.25	79.04	77.67	81.46	90.88
	I-C	93.83	79.52	78.00	86.67	90.03
	I-R	92.82	80.00	77.33	85.00	89.17
	N-C	95.40	76.19	83.33	82.08	90.88
	N-I	94.80	78.57	83.00	86.25	90.03
	N-R	95.21	76.19	82.33	81.46	89.74
I	C-R	91.91	75.23	73.00	79.58	90.60
	I-C	92.94	77.61	71.33	85.21	91.17
	I-R	91.81	75.23	76.00	85.00	86.89
	N-C	94.58	66.66	74.67	76.25	88.03
	N-I	93.89	73.33	78.67	81.88	86.89
	N-R	94.20	70.00	76.67	74.58	85.19
N	C-R	93.35	79.04	79.67	81.46	91.17
	I-C	94.05	**80.47**	80.67	85.00	90.31
	I-R	93.26	79.52	78.33	84.79	**91.74**
	N-C	**95.81**	78.09	82.67	82.29	90.88
	N-I	95.59	79.52	83.00	86.25	88.89
	N-R	95.81	75.23	**83.67**	81.67	91.17
R	C-R	93.95	79.04	78.00	81.46	90.88
	I-C	93.29	79.04	78.00	**86.46**	90.03
	I-R	92.72	77.61	77.00	85.00	90.03
	N-C	95.72	77.61	83.33	82.08	90.03
	N-I	94.58	76.66	83.33	86.25	89.74
	N-R	95.18	76.66	82.33	81.46	90.03

Table 3. Accuracy rates of stacking systems with three base classifiers (S3).

MC	BC	DNA	SONAR	HEART	MUSK	IONOSPHERE
C	C-I-R	94.24	80.00	77.00	85.83	91.17
	C-N-R	95.37	77.14	82.67	82.08	92.02
	N-I-C	95.65	79.04	83.00	86.88	90.60
	N-I-R	95.37	79.04	83.33	86.46	89.74
I	C-I-R	92.13	76.19	72.33	85.83	91.17
	C-N-R	94.87	72.85	74.67	77.92	88.60
	N-I-C	94.49	75.71	75.00	84.58	89.17
	N-I-R	94.43	74.76	77.33	85.63	86.04
N	C-I-R	94.20	80.95	81.33	87.29	90.31
	C-N-R	95.97	79.52	82.33	82.71	92.02
	N-I-C	**96.19**	81.42	84.00	88.33	90.60
	N-I-R	95.94	**81.90**	**84.33**	**88.54**	**92.31**
R	C-I-R	93.89	76.66	77.00	85.63	85.47
	C-N-R	95.75	77.61	82.67	82.08	92.31
	N-I-C	95.81	77.14	81.67	86.67	90.60
	N-I-R	95.37	70.47	84.33	86.46	91.74

Table 4. Accuracy rates of stacking systems with four base classifiers (S4).

MC	BC	DNA	SONAR	HEART	MUSK	IONOSPHERE
C	C-I-R-N	95.13	80.00	83.00	86.67	90.60
I	C-I-R-N	95.24	77.14	73.00	85.83	88.89
N	C-I-R-N	**96.06**	**80.48**	**83.33**	**87.29**	**92.88**
R	C-I-R-N	95.87	73.33	81.00	86.67	90.31

stacking state-space in each of the 5 domains. These graphics give the probability (y-axis) of obtaining a stacking system with a testing accuracy equal or better than some value (in the x-axis). The accuracies for the best base classifier (BC), boosting, and bagging are displayed as vertical lines. The most remarkable regularities found in these figures are:

- Even though S3 usually finds the best stacking system in four of the five domains, S4 has a better chance of finding good enough stacking configurations: S4's cumulative curve is mostly over S3's in all domains, Actually, this ordering is also true for S3 and S2 (S3 $>$ S2).
- S4 configurations display a smaller variation than both S2 and S3: the worse S4's stacking configuration is usually much better than the worse S3's and S2's systems, except in the HEART domain. This result has to be qualified, because, in S4, there are only 4 configurations, corresponding to the four different meta-classifiers (the base classifiers are always the same). Perhaps results would vary more if there were more

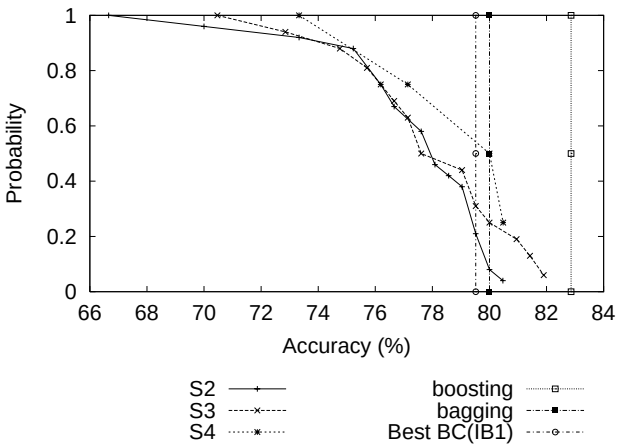

Figure 1. Cumulative probability in the SONAR domain.

Table 5. Accuracy rates of base classifiers and the best stacking systems.

DOMAIN	C4.5	IB1	NBAYES	PART
SONAR	78.57	79.52	67.14	76.67
HEART	80.33	78.67	84.00	77.67
DNA	94.14	76.44	95.37	93.29
MUSK	81.88	86.25	73.54	81.67
IONOSPHERE	90.31	86.89	83.19	91.17
	BEST S2	BEST S3	BEST S4	BAGGING/BOOSTING (WITH C4.5)
SONAR	80.47	81.90	80.48	80.00/**82.86**
HEART	83.67	**84.33**	83.33	77.00/82.00
DNA	95.81	**96.19**	96.06	94.49/94.58
MUSK	86.46	88.54	87.29	88.75/**89.38**
IONOSPHERE	91.74	92.31	92.88	90.88/**93.73**

Table 6. Best results from the three following groups: base classifiers, stacking combinations, and boosting/bagging with C4.5.

DOMAIN	BEST BASE CLASSIFIER	BEST STACKING	BOOSTING (C4.5)	DIFFERENCE BEST/WORSE
SONAR	IB1 (79.52)	S3 (81.90)	(**82.86**)	3.34
HEART	BAYES (84.00)	S3 (**84.33**)	(82.00)	2.33
DNA	BAYES (95.37)	S3 (**96.19**)	(94.58)	1.61
MUSK	IB1 (86.25)	S3 (88.54)	(**89.38**)	3.13
IONOSPHERE	PART (91.17)	S4 (92.88)	(**93.73**)	2.56

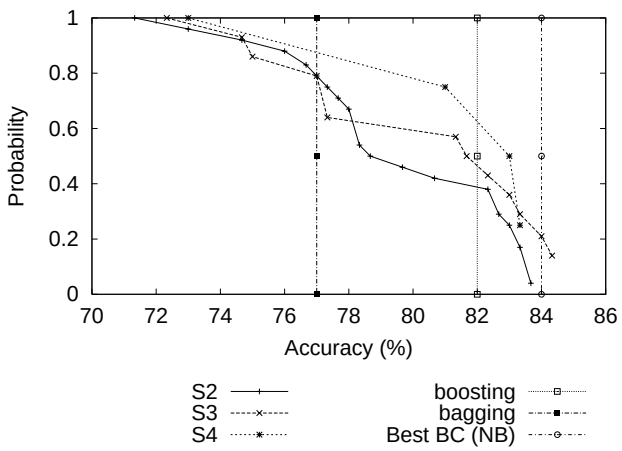

Figure 2. Cumulative probability in the HEART domain.

base classifiers to choose from.

- The probability that a stacking system is better than the best base classifier in the domain is usually not very large. Table 7 summarizes the relevant values. S2 maximum probability is 0.2083. S3 maximum is 0.5625, and this is the only value over 0.5. S4 is slightly better, because it is able to find frequently ($>=$50%) a stacking system better than any of the base classifiers at least in 3 of the 5 domains. If we consider S2 $\cup$ S3 $\cup$ S4 as the whole state-space, the maximum probability is 0.3409, which is not very large.
- It is also interesting to know whether a stacking configuration is able to get results equal or better than its best base classifier. Table 8 displays the probability that a stacking system is as good (or better) than the best base classifier in the domain, when that classifier is used by the stacking configuration. It is surprising that this probability is often quite low. For instance, when S2 $\cup$ S3 $\cup$ S4 is considered, 42.86% of stacking configurations that make use of the best domain base classifier will be less accurate than it, and this is the best case (dna domain). This reinforces the idea that it

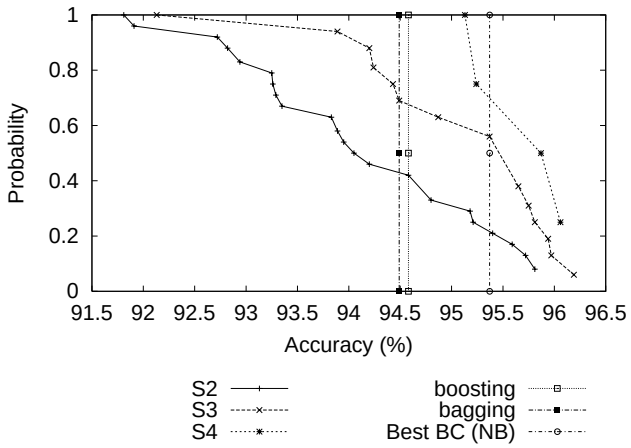

Figure 3. Cumulative probability in the DNA domain.

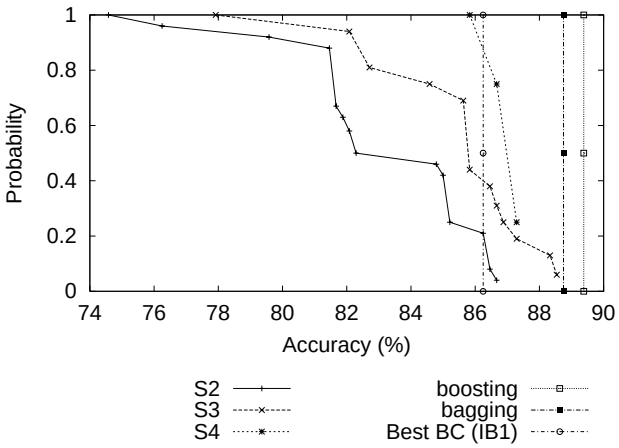

Figure 4. Cumulative probability in the MUSK domain.

Table 7. Probability of finding a stacking system equal or better than any of the base classifiers.

	S2	S3	S4	S2 ∪ S3 ∪ S4
Sonar	0.2083	0.3125	0.50	0.2727
Heart	0.0000	0.1875	0.00	0.0682
dna	0.2083	0.5625	0.50	0.3636
Musk	0.2083	0.4375	0.75	0.3409
Ionosphere	0.1667	0.4375	0.25	0.2727

is very important to be able to obtain the right combination of classifiers in the stacking mix.

Table 8. Probability that a stacking configuration that uses the best domain base classifier is better than it.

	S2	S3	S4	S2 ∪ S3 ∪ S4
dna	0.41667	0.7500	0.50	0.5714
Sonar	0.41667	0.4167	0.50	0.4286
Heart	0.00000	0.2500	0.00	0.1071
Musk	0.41667	0.5833	0.75	0.5357
Ionosphere	0.16670	0.5833	0.25	0.3571

5 Conclusions

The aim of this paper was to systematically study the state-space of hetereogeneous stacking systems. Here, we have studied empirically the state-space of stacking systems with 2, 3, and 4 base classifiers, that can be built using C4.5, PART, Naive Bayes, and IB1. As this state-space is not too large, it can be studied exhaustively. The most important and better confirmed conclusions of this paper are:

- The stacking state-space contains systems which are comparable to boosting. This is important, because even though the computational effort of searching for the best stacking configuration is larger than for boosting, the state-space defined in this paper is small enough to be explored in a reasonable time. Also, only a few base classifiers are needed to get comparable results to boosting.
- However, the density of good stacking systems is not always high. For instance, it is usually more likely than not that a stacking configuration randomly chosen will perform worse than the best base classifier for a domain. What is worse, even if the best domain base classifier is used by the stacking configuration, there is a significant probability that the configuration will do worse than it.
- Therefore, if larger state-spaces are to be searched (because we want to use more base classifiers, for instance), heuristics will be needed to do so efficiently. For instance, our systematic study suggests that Naive Bayes seems to be the most appropriate meta-classifier. Also, simple heuristic methods like hill-climbing, simulated annealing, or genetic algorithms could be used. We have used genetic algorithms with good results in [13].
- We have also found out that merely increasing the number of base classifiers does not always pay off in terms of accuracy. The best configurations obtained by S3 (three base classifiers) are better than those of S2, but also better than the best ones from S4.

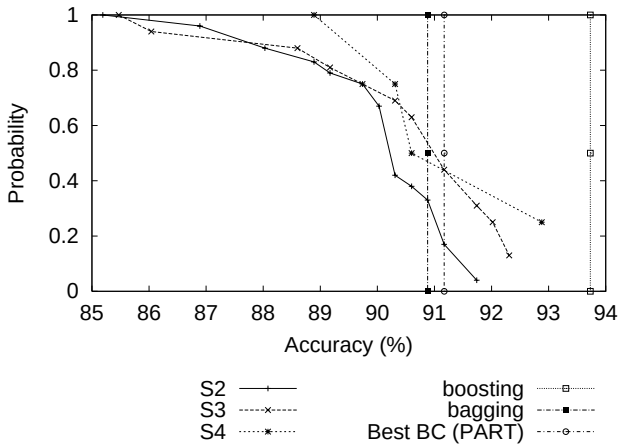

Figure 5. Cumulative probability in the IONOSPHERE **domain.**

- Therefore, it seems that the meta-classifier is not always able to find which base classifier outputs are irrelevant, so it seems a good idea to try to determine in advance which base classifiers are more appropriate (or to carry out some wrapper-style search [11]).

References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, jan 1991.

[2] K. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 65–72. Morgan Kaufmann, 2000.

[3] C. Blake and C. Merz. Uci repository of machine learning databases. databases http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.

[4] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] P. Chan and S. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In M. Kaufmann, editor, *Proceedings of Twelfth International Conference on Machine Learning*, pages 90–98, 1995.

[6] D. Fan, P. Chan, and S. Stolfo. A comparative evaluation of combiner and stacked generalization. In *Proccedings of AAAI-96 Workshop on Integrating Multiple Learning Models*, pages 40–46, 1996.

[7] D. Fan, S. Stolfo, and P. Chan. Using conflicts among multiple base classifiers to measure the performance of stacking. In *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work*, pages 10–17, 1999.

[8] E. Frank and I. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 144–151. Morgan Kaufmann, 1998.

[9] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In Springer-Verlag, editor, *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, 1995.

[10] G. John and P. Langley. Estimating continuous distribution in bayesian classifiers. In M. Kaufmann, editor, *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, 1995.

[11] R. Kohavi and G. John. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, 1995.

[12] M. LeBlanc and R. Tibshirani. Combining estimates in regression and classification. In *Technical Report 9318*. Deparment of Statistic, Univesity of Toronto, 1993.

[13] A. Ledezma, R. Aler, and D. Borrajo. *Data Mining: a Heuristic Approach*, chapter Heuristic Search Based Stacking of Classifiers. Idea Group Publishing, 2001. Accepted for publication.

[14] C. J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36:33, 1999.

[15] J. Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725–730. AAAI Press / MIT Press, 1996.

[16] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[17] K. Ting and I. Witten. Stacked generalization: when does it work? In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1997.

[18] I. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, 2000.

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