Combining Classifiers by Constructive Induction

João Gama

LIACC, FEP - University of Porto
Rua Campo Alegre, 823
4150 Porto, Portugal
Phone: (+351) 2 678830 Fax: (+351) 2 6003654
Email: jgama@ncc.up.pt
WWW: http://www.up.pt/liacc/ML

Abstract. Using multiple classifiers for increasing learning accuracy is an active research area. In this paper we present a new general method for merging classifiers. The basic idea of Cascade Generalization is to sequentially run the set of classifiers, at each step performing an extension of the original data set by adding new attributes. The new attributes are derived from the probability class distribution given by a base classifier. This constructive step extends the representational language for the high level classifiers, relaxing their bias. Cascade Generalization produces a single but structured model for the data that combines the *model class* representation of the base classifiers. We have performed an empirical evaluation of Cascade composition of three well known classifiers: *Naive Bayes*, *Linear Discriminant*, and *C4.5*. Composite models show an increase of performance, sometimes impressive, when compared with the corresponding single models, with significant statistical confidence levels.

1 Introduction

Given a learning task which algorithm should we use? Previous empirical studies have shown that there is no overall better algorithm. The ability of a chosen algorithm to induce a good generalization depends on how appropriate the class model underlying the algorithm is for the given task. An algorithm class model is the representation language it uses to express a generalization of the examples. The representation language for a standard decision tree is the DNF formalism that splits the instance space by axis-parallel hyper-planes, while the representation language for a linear discriminant function is a set of linear functions that split the instance space by oblique hyper-planes.

In statistics, Henery[12] refers Rescaling as a method used when some classes are over-predicted leading to a bias. Rescaling consists on applying the algorithms in sequence, the output of an algorithm being used as input to another algorithm. The aim would be to use the estimated probabilities $W_i = P(C_i|X)$ derived from a learning algorithm, as input to a second learning algorithm the purpose of which is to produce an unbiased estimate $Q(C_i|W)$ of the conditional probability for class C_i.

Since different learning algorithms employ different knowledge representations and search heuristics, different search spaces are explored and diverse results are obtained. The problem of finding the appropriate bias for a given task is an active research area. We can consider two main lines: on one side methods that select the most appropriate algorithm for the given task, for example Schaffer's selection by Cross-Validation, and on the other side, methods that combine predictions of different algorithms, for example Stacked Generalization [21].

The work that we present here follows the second research line. Instead of looking for methods that fit the data using a single representation language, we present a family of algorithms, under the generic name of Cascade Generalization, whose search space contains models that use different representation languages. Cascade generalization performs an iterative composition of classifiers. At each iteration a classifier is generated. The input space is extended by the addition of new attributes. Those new attributes are obtained in the form of a probability class distribution given, for each example, by the generated base classifier. The language of the final classifier is the language used by the high level generalizer. But it uses terms that are expressions from the language of low level classifiers. In this sense, Cascade Generalization generates a unified theory from the base theories. The experimental work shows that this methodology usually improves the accuracy with significant statistical levels.

The next section of the paper presents the framework of cascade generalization. In section 3, we present an illustrative example. In section 4 we review previous work in the area of multiple models. In section 5, we perform an empirical study using UCI data sets. The last section presents an analysis of the results and concludes the paper.

2 Cascade Generalization

Consider a learning set $D = (\mathbf{x_n}, Y_n)$ $n = 1, ..., N$, where $\mathbf{x_n} = [x_1, ..., x_m]$ is a multidimensional input vector, and Y_n is the output variable. Since the focus of this paper is on classification problems, Y_n takes values from a set of pre defined values, that is $Y_n \in Cl_1, ..., Cl_c$, where c is the number of classes. A classifier $\Im$ is a function that is applied to the training set in order to construct a predictor $\Im(\mathbf{x}, D)$ of y values. This is the traditional framework for classification tasks. Nevertheless, our framework requires that the predictor $\Im(\mathbf{x}, D)$ outputs a vector of conditional probability distribution $[p1, ..., pc]$, where p_i represents the probability that the example $\mathbf{x}$ belongs to class i, this is $P(y = Cl_i|\mathbf{x})$. The class that is assigned to the example $\mathbf{x}$, is that one that maximizes this last expression.

Most of the commonly used classifiers, such as *Naive Bayes* and *Discriminant*, classify each example in this way. Other classifiers, for example *C4.5*, have a different strategy for classifying an example, but it requires small changes in order to obtain a probability class distribution.

We define a constructive operator $\Phi(D', \Im(\mathbf{x}, D))$. This operator has two input parameters: a data set D' and a classifier $\Im(\mathbf{x}, D)$. The classifier $\Im$ generates

a theory from the training data D. For each example $\mathbf{x} \in D'$, the generated theory outputs a probability class distribution. The operator Φ concatenates both the input vector $\mathbf{x}$ with the output probability class distribution. The output of $\Phi(D', \Im(\mathbf{x}, D))$ is a new data set D". The cardinality of D" is equal to the cardinality of D' (they have the same number of examples). Each example in $\mathbf{x} \in D''$ has an equivalent example in D', but augmented with c new attributes. The new attributes are the elements of the vector of probability class distribution obtained when applying classifier $\Im(\mathbf{x}, D)$ to the example $\mathbf{x}$.

Cascade generalization is a sequential composition of classifiers, that at each generalization level applies the Φ operator. Given a training set L, a test set T, and two classifiers $\Im_1$, and $\Im_2$, Cascade generalization proceeds as follows: Using classifier $\Im_1$, generates the $Level_1$ data:

$$Level_1 train = \Phi(L, \Im_1(\mathbf{x}, L))$$
$$Level_1 test = \Phi(T, \Im_1(\mathbf{x}, L))$$

Classifier $\Im_2$ learns on $Level_1$ training data and classifies the $Level_1$ test data:

$$\Im_2(\mathbf{x}, Level_1 train) \text{ for each } \mathbf{x} \in Level_1 test$$

Those steps perform the basic sequence of a cascade generalization of classifier $\Im_2$ after classifier $\Im_1$. We represent the basic sequence by the symbol ∇.

The previous composition could be shortly represented by:

$$\Im_2 \nabla \Im_1 = \Im_2(\mathbf{x}, \Phi(L, \Im_1(\mathbf{x}', L))) \text{ for each } \mathbf{x}' \in \Phi(T, \Im_1(\mathbf{x}'', L))$$

This is the simplest formulation of *Cascade Generalization*. Some possible extensions include the composition of n classifiers, and the parallel composition of classifiers.

A composition of n classifiers is represented by:

$$\Im_n \nabla \Im_{n-1} \nabla \Im_{n-2} ... \nabla \Im_1$$

In this case, Cascade Generalization generates *n-1* levels of data. The high level theory, is that one given by the $\Im_n$ classifier.

A variant of cascade generalization, which include several algorithms in parallel, could be represented in this formalism:

$$\Im_{n+1} \nabla [\Im_1, ..., \Im_n] = \Im_{n+1}(\mathbf{x}, \Phi(L, [\Im_1(\mathbf{x}', L), ..., \Im_n(\mathbf{x}', L)]))$$
$$\text{for each } \mathbf{x} \in \Phi(T, [\Im_1(\mathbf{x}', L), ..., \Im_n(\mathbf{x}', L)])$$

The algorithms $\Im_1, ..., \Im_n$ run in parallel. The operator $\Phi(L, [\Im_1(\mathbf{x}', L), ..., \Im_n(\mathbf{x}', L)])$ returns a new data set L'. L' contains the same number of examples of L. Each example on L' contains $n * c$ new attributes, where c is the number of classes. Each algorithm in the set $[\Im_1, ..., \Im_n]$ contributes with c new attributes.

3 An Illustrative Example

In this example we will consider the UCI data set *Monks-2* [20]. The *Monk's problems* are an artificial robot domain, well known in the Machine Learning community. The robots are described by six different attributes and classified into one of two classes. We have chosen the *Monks-2 problem* because it is known that this is a difficult task for systems that learn decision trees in an attribute-value logic formalism. The decision rule for the problem is: **"The robot is O.K. if exactly two of the six attributes have their *first* value".** This problem is similar to *parity* problems. It combines different attributes in a way which makes it complicated to describe in DNF or CNF using the given attributes only.

Using ten fold Cross Validation, the error rate of *C4.5* is 32.9%, and of *Naive Bayes* is 49.5%. The composite model C4.5 after *Naive Bayes*, $C4.5\nabla NaiveBayes$, operates as follows: the $Level_1$ data was generated, using the *Naive Bayes* as the classifier. C4.5 was used for the $Level_1$ data. The composition $C4.5\nabla NaiveBayes$, obtains an error rate of 17.8%, which is substantially lower than the error rates of both *C4.5* and *Naive Bayes*. None of the algorithms in isolation can capture the underlying structure of the data. In this case, Cascade was able to achieve a notable increase of performance. Figure 1 presents one of the trees generated by $C4.5\nabla NaiveBayes$.

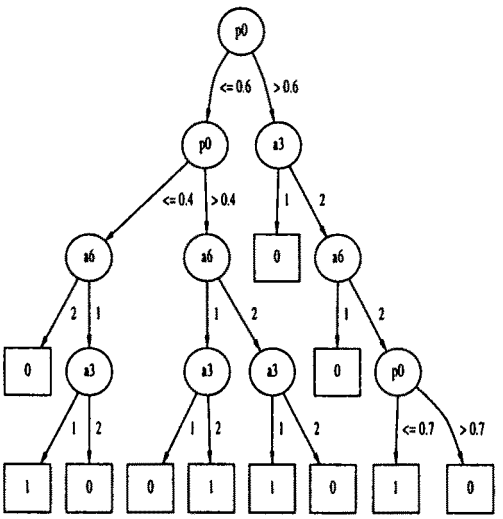

Fig. 1. Tree generated by C4.5 ∇Bayes

The tree contains a mixture of the original attributes (a3, a6) and the new attributes constructed by *Naive Bayes* (*p0*). At the root of the tree, appears the attribute *p0*. This attribute is the conditional probability $p(Class = False|\mathbf{x})$ given by the *Naive Bayes*. The classification rule used by *Naive Bayes* is: choose the $Class_i$ that maximizes $p(Class_i|\mathbf{x})$. The decision tree generated by C4.5 uses the constructed attributes given by Naive Bayes, but redefining different decision surfaces. Because this is a two class problem, the Bayes rule uses p0 with threshold 0.5, while the decision tree chose the threshold at 0.6. Those decision nodes are a kind of function given by the Bayes strategy. For example, the attribute *p0* can be seen as a function that computes $p(Class = False|\mathbf{x})$ using the Bayes theorem. The decision tree performs a sequence of tests based on the conditional probabilities given by the Bayes theorem. In a certain sense, this decision tree combines both representation languages: Bayes and Trees. The constructive step performed by *Cascade*, inserts new axis that incorporates new knowledge provided by the Naive Bayes. It is this new knowl-

edge that allows the significant increase of performance verified with the Decision Tree, despite the limitations of Naive Bayes to fit complex spaces. It is this kind of synergies between classifiers that Cascade Generalization explores.

4 Related Work

We can analyze previous work on the area of multiple models through two dimensions. One dimension is related to the different methods used for combining classifications. The other dimension is related to the methods used for generating different models.

4.1 Combining Classifications

Combining classifications usually occurs at classification time. We can consider two main lines of research. One group includes methods where all base classifiers are consulted in order to classify a query example, the other, methods that characterize the area of expertise of the base classifiers and for a query point only ask the opinion of the experts. Voting is the most common method used to combine classifiers. As pointed in Ali[1], this strategy is motivated by the Bayesian learning theory which stipulates that in order to maximize the predictive accuracy, instead of using just a single learning model, one should ideally use all hypotheses (models) in the hypothesis space. The vote of each hypothesis should be weighted by the posterior probability of that hypothesis given the training data. Several variants of the voting method can be found in machine learning literature: from uniform voting where the opinion of all base classifiers contributes to the final classification with the same strength, to weighted voting, where each base classifier has a weight associated, that could change over the time, and strengthens the classification given by the classifier.

4.2 Generating Different Models

Buntine's Ph.D. thesis [5], refers to at least two different ways of generating multiple classifiers. The first one, involves a single tree that is generated from the training set and then pruned back in different ways. The second method is referred to as Option Trees. These kind of trees are in effect an ensemble of trees. Each decision node contains not only a univariate test, but also stores information about other promising tests. When using an Option Tree as a classifier the different options are consulted and the final classification is given by voting. He shows that, if the goal is to obtain an increase of performance, the second method out performs the first, basically, due to the fact that it produces different syntactic models. Breiman[2] proposes Bagging, that produces replications of the training set by sampling with replacement. Each replication of the training set has the same size as the original data, but some examples don't appear in it, while others may appear more than once. From each replication

of the training set a classifier is generated. All classifiers are used in order to classify each example on the test set, usually using a uniform vote scheme.

The Boosting algorithm from Freund and Schapire [9] maintains a weight for each example in the training set that reflects its importance. Adjusting the weights causes the learner to focus on different examples leading to different classifiers. Boosting is an iterative algorithm. At each iteration the weights are adjusted in order to reflect the performance of the corresponding classifier. The weight of the misclassified examples is increased. The final classifier aggregates the learned classifier at each iteration by weighted voting. The weight of each classifier is a function of its accuracy.

Wolpert [21] proposes Stacked Generalization, a technique that uses learning in two levels. A learning algorithm is used to determine how the outputs of the base classifiers should be combined. The original data set constitutes the level zero data. All the base classifiers run at this level. The level one data are the outputs of the base classifiers. Another learning process occurs using as input the level one data and as output the final classification. This is a more sophisticated technique of cross validation that could reduce the error due to the bias.

Brodley[4] presents MCS, a hybrid algorithm that combines in a single tree, nodes that are univariate tests, multivariate tests generated by linear machines, and instance based learners. At each node MCS uses a set of If-Then rules in order to perform a hill-climbing search for the best hypothesis space and search bias for the given partition of the dataset. The set of rules incorporates knowledge from expert domains. Gama[10, 11] presents *Ltree*, also a hybrid algorithm that combines a decision tree with a linear discriminant by means of constructive induction.

Chan and Stolfo[6] presents two schemes for classifier combination: *arbiter* and *combiner*. Both schemes are based on meta learning, where a meta-classifier is generated from a training data, built based on the predictions of the base classifiers. An arbiter is also a classifier and is used in order to arbitrate among predictions generated by the different base classifiers. Later[7], extended this framework using $arbiters/combiners$ in an hierarchical fashion generating $arbiter/combiner$ binary trees.

4.3 Discussion

Reported results relative to Boosting or Bagging are quite impressive. Using 10 iterations (that is generating 10 classifiers) Quinlan[16] reports reductions of the error rate between 10% and 19%. Quinlan argues that these techniques are mainly applicable for unstable classifiers. Both techniques requires that the learning system should not be stable, in order to obtain different classifiers when there are small changes in the training set.

Under an analysis of bias-variance decomposition of the error of a classifier[13], the reduction of the error observed when using Boosting or Bagging is mainly due to the reduction in the *variance*. Ali[1] refers to that *"the number of training examples needed by Boosting increases as a function of the accuracy of the*

learned model. Boosting could not be used to learn many models on the modest training set sizes used in this paper.".

Wolpert[21] says that successful implementations of Stacked Generalization is a *"black art"*, for classification tasks and the conditions under which Stacked works are still unknown. Recently, Ting[18] have shown that successful stacked generalization requires to use output class distributions rather than class predictions. In their experiments, only the MLR algorithm (a linear discriminant) was suitable for level-1 generalizer.

Tumer[19] presents analytical results that showed that the combined error rate depends on the error rate of individual classifiers and the correlation among them. This was confirmed in the empirical study presented in [1].

The main point of Cascade Generalization is its ability to merge different models. As such, we get a single model whose components are terms of the base model's language. The bias restriction imposed by using single model classes is relaxed in the directions given by the base classifiers. Cascade Generalization gives a single structured model for the data, and this is a strong advantage over the methods that combine classifiers by voting. Another advantage of Cascade Generalization is related to the use of probability class distributions. Usual learning algorithms produced by the Machine Learning community uses categories when classifying examples. Combining classifiers by means of categorical classes looses the strength of the classifier in its prediction. The use of probability class distributions allows us to explore that information.

5 Experiments

5.1 The Algorithms

Ali [1] and Tumer[19] among other authors, suggest as a method that allows us to reduce the correlation errors, the use of "radically different types of classifiers". This was the criterion that we have used in order to select the algorithms for the experimental work. We use three classifiers: a *Naive Bayes*, a *Linear Discriminant*, and a *Decision Tree.*

Naive Bayes The Bayes approach in order to classify a new example E, is the use of Bayes theorem in order to compute the probability of each class C_i, given the example. The chosen class is the one that maximizes: $p(C_i|E) = p(C_i)p(E|C_i)/p(E)$. If the attributes are independent, $p(E|Ci)$ can be decomposed into the product $p(v_1|C_i) * ... * p(v_k|C_i)$. Domingos [8] show that this procedure has a surprisingly good performance in a wide variety of domains, including many where there are clear dependencies between attributes. The required probabilities are computed from the training set. In the case of nominal attributes we use counts. Continuous attributes were discretized. The number of bins that we use is a function of the number of different values observed on the training set: $k = min(10; nr.\ different\ values)$. This heuristic was used in [8] and elsewhere with good overall results. Missing values were treated as another possible value for the attribute, both on the training and test data. *Naive*

Bayes uses all the attributes in order to classify a query point. Langley [14] refers that *Naive Bayes* relies on an important assumption: that the variability of the dataset can be summarized by a single probabilistic description, and that these is sufficient to distinguish between classes. From an analysis of *Bias-Variance*, this implies that *Naive Bayes* uses a reduced set of models to fit the data. The result is low variance, but if the data cannot be adequately represented by the set of models, we obtain a large bias.

Linear Discriminant A linear discriminant function is a linear composition of the attributes where the sum of the squared differences between class means is maximal relative to the internal class variance. It is assumed that the attribute vectors for examples of class C_i are independent and follow a certain probability distribution with probability density function f_i. A new point with attribute vector $\mathbf{x}$ is then assigned to that class for which the probability density function $f_i(\mathbf{x})$ is maximal. This means that the points for each class are distributed in a cluster centered at μ_i. The boundary separating two classes is a hyper-plane and it passes through the mid point of the two centers. If there are only two classes one hyper-plane is needed to separate the classes. In the general case of q classes, $q-1$ hyper-planes are needed to separate the classes. By applying the linear discriminant procedure described below, we get $q_{node}-1$ hyper-planes. The equation of each hyper-plane is given by[12]:

$$H_i = \alpha_i + \sum_j \beta_{ij} * x_j \text{ where } \alpha_i = -\tfrac{1}{2}\mu_i^T S^{-1}\mu_i \text{ and } \beta_i = S^{-1}\mu_i$$

We use a Singular Value Decomposition (SVD) in order to compute S^{-1}. SVD is numerically stable and is a tool for detecting sources of collinearity. This last aspect is used as a method for reducing the features used at each linear combination. *Discrim* uses all, or almost all, the attributes in order to classify a query point. Breiman,[3] refers that from an analysis of Bias-Variance, Linear Discriminant is a stable classifier although it can fit a small number of models. It achieves their stability by having a limited set of models to fit the data. The result is low variance, but if the data cannot be adequately represented by the set of models, then the result is a large bias.

Decision Tree We have used *C4.5* (release 8) [17]. This is a well known decision tree generator and widely used by the Machine Learning community. In order to obtain a probability class distribution, we need to modify C4.5. C4.5 stores a distribution of the examples that fall at each leaf. From this distribution and using m-estimates [1] [15] we obtain a probability class distribution at each leaf. A *Decision tree* uses only a subset of the available attributes, in order to classify a query point. Breiman [3] among other researchers, note that Decision Trees are unstable classifiers. Small variations on the training set could cause large changes in the resulting predictors. These classifiers have high variance but they can fit any kind of data: the bias of a decision tree is low.

[1] In all the experiments reported m was set to 0.5.

5.2 The Datasets

We have chosen 17 data sets from the UCI repository. All of them are well known and previously used in other comparative studies. In order to evaluate the proposed methodology we performed a 10 fold Cross Validation (CV) on the chosen datasets. Datasets were permuted once before the CV procedure. All algorithms where used with the default settings. In each iteration of CV, all algorithms were trained on the same training partition of the data. Classifiers were also evaluated on the same test partition of the data. Comparisons between algorithms were performed using *t-paired tests* with significant level set at 95%.

Table 1 presents data sets characteristics and the error rate and standard deviation of each base classifier. Relative to each algorithm, + (-) sign in the first column means that the error rate of this algorithm is significantly better (worse) than C4.5. These results provide an evidence, once more, that no single algorithm is better overall.

Dataset	Class	Examples	Types		Bayes		Discrim	C4.5
Australian	2	690	7 N 6 Cont		13.8 ±3.5		14.1 ±6	15.3 ±6.3
Balance	3	625	4 Cont	-	28.8 ±6.3	+	13.3 ±4.4	22.3 ±5.3
Breast	2	699	9 Cont		2.4 ±1.9	+	4.1 ±6	6.1 ±6.1
Diabetes	2	768	8 Cont		25.7 ±5.5	+	22.7 ±5	24.8 ±6.6
German	2	1000	24 Cont		27.7 ±4.4	+	24.0 ±6.2	29.1 ±3.7
Glass	6	213	9 Cont	-	41.8 ±12	-	41.3 ±11	32.3 ±9.6
Heart	2	270	6 N 7 Cont	+	16.7 ±5		16.7 ±3.6	19.9 ±7.2
Ionosphere	2	351	33 Cont		9.1 ±6.3	-	13.4 ±5.4	9.1 ±5.8
Iris	3	150	4 Cont		6.0 ±4.9		2.0 ±3.2	4.7 ±4.5
Monks-1	2	432	6 Nom	-	25.0 ±3.9	-	33.3 ±11.3	2.3 ±4.4
Monks-2	2	432	6 Nom	-	49.6 ±9.0		34.0 ±5.9	32.9 ±5.9
Monks-3	2	432	6 Nom	-	2.8 ±2.4	-	22.5 ±8.7	0.0 ±0.0
Satimage	6	6435	36 Cont	-	18.8 ±1.5	-	16.1 ±1.5	13.9 ±1.3
Segment	7	2310	18 Cont	-	9.5 ±2.1	-	8.3 ±2.5	3.3 ±1.3
Vehicle	4	846	18 Cont	-	41.4 ±3.9	+	22.2 ±5.1	28.8 ±3.9
Waveform	3	2581	21 Cont	+	18.8 ±1.5	+	15.3 ±2	24.0 ±2.2
Wine	3	178	13 Cont		2.8 ±4		1.7 ±3.8	6.7 ±8.2
Average of Error rates					20.1		17.9	16.2

Table 1. Data Characteristics and Results of Base Classifiers

5.3 Cascade Generalization

We have run all the possible two level combinations of base classifiers. Table 2(a) presents the results of using C4.5 at the top level. Each column corresponds to a Cascade Generalization combination. For each combination, the significance of *t test* is presented comparing the composite model with the individual components, in the same order that they appear on the header.

The trend on these results is a clear improvement over the base classifiers. We never observe an error rate degradation of a composite model in relation to the individual components. Using C4.5 as the high level classifier the performance is improved with a significant statistical level of 95%, 22 times over one of the components, and it degraded 5 times. Using *Naive Bayes* at the top, there

Dataset	C4.5∇Bayes			C4.5∇C4.5			C4.5∇Disc			C4∇Dis∇Bay		Stacked Gen
Australian			14.3 ±3.1	15.2 ±6.2			14.8 ±6.1			13.6 ±6		14.3 ±5
Balance	+	+	6.1 ±2.8	22.1 ±5.2	+	+	5.4 ±2.0			6.6 ±2	+	12.5 ±5
Breast(W)			2.8 ±1.7	5.6 ±4.8	+		4.1 ±6.0			2.7 ±2		2.4 ±2
Diabetes			25.2 ±6.9	24.4 ±6.9			24.2 ±5.8			24.8 ±9		22.4 ±6
German	+	+	24.9 ±4.4	29.1 ±3.7			26.2 ±6.0			28.4 ±4	-	25.0 ±5
Glass			38.1 ±9.6	32.3 ±12.0			36.1 ±10.9			37.6 ±12		34.3 ±12
Heart		-	21.1 ±4.9	20.0 ±7.2			17.8 ±5.5			17.0 ±5		16.3 ±8
Iono		-	13.1 ±6.6	8.9 ±5.8	-		13.1 ±5.1			10.6 ±6		10.6 ±6
Iris			6.7 ±5.4	4.7 ±4.5			3.4 ±3.4			3.3 ±4		4.0 ±3
Monks-1		+	1.6 ±3.1	2.3 ±4.4		+	2.3 ±4.4			1.4 ±3		2.3 ±4
Monks-2	+	+	17.9 ±9.9	32.9 ±5.9			32.9 ±5.9			16.7 ±9	+	32.9 ±6
Monks-3		+	0.4 ±0.9	0.0 ±0.0		+	0.0 ±0.0			0.2 ±1	+	2.1 ±2
Satimage	+	+	13.0 ±1.4	13.7 ±1.3	+	+	12.4 ±1.5			13.0 ±1		13.5 ±1
Segment		+	4.0 ±0.8	3.2 ±1.4		+	3.4 ±1.5			3.0 ±1		3.4 ±1
Vehicle		+	27.4 ±5.9	28.2 ±4.3		-	28.2 ±4.3			22.1 ±3	+	29.2 ±4
Waveform	+		17.2 ±2.3	- 24.4 ±2.0	+		16.6 ±1.4			16.9 ±2		16.5 ±2
Wine			3.9 ±4.6	6.7 ±8.2			2.2 ±3.9			3.4 ±4		2.8 ±4
Mean	13.9			16.1			14.3			13.0		14.4

Table 2. (a) Results of Cascade Generalization. (b)Comparison with Stacked

are 21 cases against 9. Using *Discrim* at the top, there are 22 cases against 7. In same cases, there is a significant increase of performance when compared to all the components. For example, the composition *C4.5* ∇*Naive Bayes* improves, with statistical significance, both components on 4 datasets, *C4.5* ∇*Discrim* and *Naive Bayes* ∇*Discrim* on 2 datasets, and *Discrim* ∇ *C4.5* on 1 dataset. The most promising combinations are C4.5∇*Discrim* and C4.5∇*Naive Bayes*. The new attributes built by *Discrim* or *Naive Bayes* set relations between attributes, that are outside the scope of DNF algorithms like *C4.5*. Those new attributes systematically appears at the root of the composite models. A particular successful application of *Cascade* is on *Balance* dataset.

In another experiment, we have compared C4∇Discrim∇Bayes against *Stacked Generalization*, which was reimplemented following the method of Ting[18]. In this scheme *Discrim* is the $level_1$ algorithm. C4.5 and Bayes are the $level_0$ algorithms. The attributes of the $level_1$ data are the conditional probabilities $P(C_i|\mathbf{x})$, given by the $level_0$ classifiers. The $level_1$ data is built using a (internal) 5 fold stratified cross validation. On those datasets, C4∇Discrim∇Bayes performs significantly better on 4 datasets and worst on one. *Cascade* competes well with Stacked method, with the advantage that it doesn't use the *internal cross validation.*

5.4 How Far from the Best?

Error rates are not comparable between datasets. Although the *t paired tests* procedure is commonly used for determining whether two means are statistically different, this procedure only permits to compare two algorithms. We are interested in comparisons which involve several algorithms. As such, for each dataset we identify the classifier with lowest error rate. Call it E_{low}. Denote the error rate of $algorithm_i$ on the given dataset as E_{alg_i}. Now we compute the *Error margin* as

C4∇Dis	C4∇Bay	*SG*	C4∇C4	C4	Bay∇C4	Dis∇Bay	Disc	Bay∇Dis	Dis∇C4	Dis∇Dis	Bay∇Bay	Bayes
1.8	1.9	2.5	4.8	4.9	5.5	5.9	7.0	7.1	7.1	7.5	8.3	8.9

Table 3. Average of Distances to Best

the standard deviation of a Bernoulli distribution: $E_m = \sqrt{E_{low} * (1 - E_{low})/N}$ where N is the number of examples in the test set. For each algorithm in comparison, we compute the distance to the best algorithm in terms of E_m. That is, low value of $Distance_i$, means that the $algorithm_i$ has an error rate similar to the best algorithm, whilst high value means that the performance of $algorithm_i$ is far from that of the best algorithm. The goal of this analysis is to compare algorithm performance across datasets. E_m is a criterion that can give insights about the difficulty of the problem. Table 3 summarizes the averages of distances of all models.

6 Conclusions

This work presents a new methodology for classifier combination. The basic idea of Cascade Generalization consists on a reformulation of the input space by means of insertion of new attributes. The new attributes are obtained by applying a base classifier. The number of new attributes is equal to the number of classes, and for each example, they are computed as the conditional probability of the example belonging to $class_i$ given by the base classifier. The new attributes are terms, or functions, in the representational language of the base classifier. This constructive step acts as a way of extending the description language bias of the high level classifiers.

There are two main points that differentiate Cascade Generalization from other previous methods on multiple models. The first one is related with its ability in merging different models. We get a single model whose components are terms of the base model's language. The bias restrictions imposed by using single model classes are relaxed in the directions given by the base classifiers. This aspect is explored by combinations like C4.5∇*Discrim* or C4.5∇*Naive Bayes*. The new attributes built by *Discrim* or *Naive Bayes* set relations between attributes, that are outside the scope of DNF algorithms like *C4.5*. Those new attributes systematically appears at the root of the composite models.

Cascade Generalization gives a single structured model for the data, and in this way is more adapted to capture insights about problem structure. The second point is related to the use of probability class distributions. The use of probability class distributions allows us to exploit the information about the strength of the classifier. This is very useful information, especially when combining predictions of classifiers. We have shown that this methodology can improve the accuracy of the base classifiers, preserving the ability to provide a single and structured model for the data.

Acknowledgments: Gratitude is expressed to the support given by the FEDER and PRAXIS XXI projects and the Plurianual support attributed to LIACC. Also to P.Brazdil and the anonymous reviewers for useful comments.

References

1. Ali, K. and Pazzani, M. (1996) "Error reduction through Learning Multiple Descriptions", in *Machine Learning, Vol. 24, No. 1* Kluwer Academic Publishers
2. Breiman,L. (1996) "Bagging predictors", in *Machine Learning, 24* Kluwer Academic Publishers
3. Breiman,L. (1996) "Bias, Variance, and Arcing Classifiers", Technical Report 460, Statistics Department, University of California
4. Brodley, C. (1995) "Recursive Automatic Bias Selection for Classifier Construction", in *Machine Learning, 20*, 1995, Kluwer Academic Publishers
5. Buntine, W. (1990) "A theory of Learning Classification Rules", Phd Thesis, University of Sydney
6. Chan P. and Stolfo S., (1995) "A Comparative Evaluation of Voting and Meta-learning on Partitioned Data", in *Machine Learning Proc of 12th International Conference*, Ed. L.Saitta
7. Chan P. and Stolfo S. (1995) "Learning Arbiter and Combiner Trees from Partitioned Data for Scaling Machine Learning", KDD 95
8. Domingos P. and Pazzani M. (1996) "Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier", in *Machine Learning Proc. of 12th International Conference*, Ed. L.Saitta
9. Freund, Y. and Schapire, R (1996) "Experiments with a new boosting algorithm", in *Machine Learning Proc of 13th International Conference*, Ed. L. Saitta
10. Gama, J, (1997) "Probabilistic Linear Tree", in *Machine Learning Proc. of the 14th International Conference* Ed. D.Fisher
11. Gama,J. (1997) "Oblique Linear Tree", in *Advances in Intelligent Data Analysis - Reasoning about Data'*, Ed. X.Liu, P.Cohen, M.Berthold, Springer Verlag LNCS
12. Henery R. (1997) "Combining Classification Procedures" in *Machine Learning and Statistics. The Interface.* Ed. Nakhaeizadeh, C. Taylor, John Wiley & Sons, Inc.
13. Kohavi, R and Wolpert, D. (1996) "Bias plus Variance Decomposition for zero-one loss function", in *Machine Learning Proc of 13th International Conference*, Ed. Lorenza Saitta
14. Langley P. (1993) "Induction of recursive Bayesian Classifiers", in *Machine Learning: ECML-93* Ed. P.Brazdil, LNAI n667, Springer Verlag
15. Mitchell T. (1997) *Machine Learning*, MacGraw-Hill Companies, Inc.
16. Quinlan R., (1996) "Bagging, Boosting and C4.5", *Procs. 13th American Association for Artificial Intelligence*, AAAI Press
17. Quinlan, R. (1993) *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, Inc.
18. Ting K.M. and Witten I.H. (1997) "Stacked Generalization: when does it work ?", in *Procs. International Joint Conference on Artificial Intelligence*
19. Tumer K. and Ghosh J. (1995) "Classifier combining: analytical results and implications", in *Proceedings of Workshop in Induction of Multiple Learning Models*
20. Thrun S., et all, (1991) *The Monk's problems: A performance Comparison of different Learning Algorithms*, CMU-CS-91-197
21. Wolpert D. (1992) "Stacked Generalization", *Neural Networks Vol.5*, Pergamon Press