Genetic Approach for Optimizing Ensembles of Classifiers

Fco. Javier Ordóñez and **Agapito Ledezma** and **Araceli Sanchis**
Universidad Carlos III de Madrid
Avda. de la Universidad, 30, 28911
Leganés (Madrid). Spain

Abstract

An ensemble of classifiers is a set of classifiers whose predictions are combined in some way to classify new instances. Early research has shown that, in general, an ensemble of classifiers is more accurate than any of the single classifiers in the ensemble. Usually the gains obtained by combining different classifiers are more affected by the chosen classifiers than by the used combination. It is common in the research on this topic to select by hand the right combination of classifiers and the method to combine them, but the approach presented in this work uses genetic algorithms for selecting the classifiers and the combination method to use. Our approach, *GA-Ensemble*, is inspired by a previous work, called *GA-Stacking*. *GA-Stacking* is a method that uses genetic algorithms to find domain-specific *Stacking* configurations. The main goal of this work is to improve the efficiency of *GA-Stacking* and to compare *GA-Ensemble* with current ensemble building techniques. Preliminary results have show that the approach finds ensembles of classifiers whose performance is as good as the best techniques, without having to set up manually the classifiers and the ensemble method.

Introduction

In recent years there has been a growing interest in a particular area of Machine Learning: the combination of classifiers (Dietterich 1997). This approach is known as ensembles of classifiers in the supervised learning area, and the main idea behind is that ensembles are often much more accurate than the individual classifiers that make them up. Usually the members of the ensemble are generated by applying a single learning algorithm (Dietterich 2000) (homogeneous classifiers). On the other hand, there are other research in the area that use different learning algorithms over a dataset to generate the members of the ensemble (Wolpert 1992) (heterogeneous classifiers).

In order to generate a homogeneous ensemble of classifiers there are several methods that can be grouped in: sub-sampling the training examples (e.g. bagging (Breiman 1996) and boosting (Freund & Schapire 1995)); manipulating the input features (Cherkauer 1996); manipulating the output target (e.g. ECOC (Dietterich & Bakiri 1995)); and injecting randomness in the learning algorithm (Kolen & Pollack 1991). Once the classifiers have been generated, they are combined, in most cases by voting or weighted voting.

One example of methods that use different learning algorithms in order to generate a heterogeneous ensemble of classifiers is *Stacking* (Wolpert 1992). *Stacking* uses an algorithm to learn how to combine the outputs of a set of classifiers that have been obtained by different learning algorithms (meta-classifier).

One of the problems of *Stacking* is how to obtain the right combination of base-level classifiers and the meta-classifier, specially in relation to each specific dataset. In a previous work (Ledezma *et al.* 2004), we presented an approach which used Genetic Algorithms (GA) to search for good domain-specific *Stacking* configurations. *GA-Stacking* found several *Stacking* configurations that performed very well given a specific domain, but the time it needed to find a solution grew exponentially depending on the characteristics of the domain. The approach presented in this paper is inspired on *GA-Stacking*, continuing that line of thought, but not limited to find *Stacking* configurations. Some somewhat related approaches have used recently GA's in order to determine optimal ensembles of classifiers in different contexts (Altinçay 2004; de Oliveira *et al.* 2005). In this work we let the genetic algorithm to find a good set of base level classifiers and which method is more suitable to combine them, either using a metaclassifier or combining the base level classifiers by voting. Furthermore, we have extended the previous *GA-Stacking* approach by using a set of classifiers trained *a priori* in order to improve the efficiency of the genetic algorithm. Finally we have compared results with two recent ensembles methods based on *Stacking*.

This paper is organized as follows. Section "*Stacking* and *GA-Stacking*" gives some background on *Stacking* and show our previous work based on genetic algorithms, *GA-Stacking*. Section "*GA-Ensemble*" introduces our genetic approach, *GA-Ensemble*. Sections "Experimental setup" and "Empirical results" describe the experimental setup and the results, respectively. The last section, "Conclusions and future works", draws some conclusions.

Stacking and *GA-Stacking*

Stacking is the abbreviation to refer to *Stacked Generalization* (Wolpert 1992). The main idea behind Stacking is

to combine classifiers from different learners such as decision trees, instance-based, bayesian or rule-based learners. *Stacked Generalization* works by deducing the biases of the generalizer(s) with respect to a provided learning set. Each classifier uses different knowledge representation and different learning biases, so the hypothesis space will be explored differently, and different classifiers will be obtained. 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 (also known as level-1 model or meta-classifier) tries to learn how the decisions of the base classifiers (or level-0 models) should be combined to obtain the final classification. 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.

One problem of *Stacking* is how to obtain the right combination of classifiers, especially in relation to each specific dataset. If the number of classifiers and algorithms to use is small, the problem can be solved by a simple method: exhaustive search. But when it becomes unfeasible to use exhaustive search, it is very difficult to find the best classifiers combination. The work presented in this paper is based on a previous approach called *GA-Stacking*, which used a genetic algorithm to find *Stacking* configurations for a specific domain.

Most work on Stacking focuses in the selection of the meta-level data and algorithm to generate the base-level classifiers (Seewald 2002; Dzeroski & Zenko 2004). In the *GA-Stacking* approach a genetic algorithm was used to generate the whole *Stacking* system configuration (level-0 and level-1 classifiers) to perform reasonably well in a specific domain. In other words, each individual in the population of the GA, represented a different *Stacking* system configuration.

In a preliminary *GA-Stacking* work (Ledezma, Aler, & Borrajo 2001), the search carried out by the genetic algorithm was limited to find a *Stacking* configuration with four base-level classifiers and the meta-classifier from a set of six available algorithms. However, in the last approach the search space was extended enlarging the number of possible base-level classifiers to ten and including the learning parameters of the algorithms in the search. This implies a very different and much richer search space.

Results showed that *GA-Stacking* was comparable to the best results reported so far, and it was never significantly worse than the other systems tested. The main advantage of *GA-Stacking* was its flexibility and extensibility, since it could use new learning algorithms as soon as they were created, not being restricted by *a priori* assumptions. It was unnecessary to specify in advance parameters such as the number of base classifiers or the algorithms available to be used. On the other hand, *GA-Stacking* required a longer execution time than the other approaches, because several generations of individuals had to be evaluated to obtain a good solution.

GA-Ensemble

In this section we will describe the approach taken by *GA-Ensemble* to build the ensembles of classifiers. As we detail in a previous section, *GA-Stacking* focuses in generating optimized *Stacking* systems, and although good results were reported, a very long execution time was required. The question is: *how to improve the efficiency without losing accuracy?*. *GA-Ensemble* tries to determine, in a reasonable time, which classifiers and which method to combine them is the best option for a specific domain. To solve this issue, in this work the genetic algorithm makes use of a pool of trained classifiers (Zhou *et al.* 2001); that is, all the algorithms needed by the genetic algorithm are trained *a priori* to avoid training them in successive generations of the genetic algorithm. Before the search starts, and considering we have used cross-validation process to prevent overfitting, the training folds of the data set are identified and the set of algorithms available to be used in the search is trained a *priori* to generate the classifiers. Making unnecessary to regenerate the classifiers in every iteration of the genetic algorithm (as was the case in *GA-Stacking*).

GA-Ensemble has three phases (see Figure 1): classifiers generation, problem encoding and the genetic search. Next, we will detail each one of these phases.

Classifiers generation

The first step of *GA-Ensemble* is to take the domain data and the available learning algorithms as input to generate the pool of classifiers, training the learning algorithms. Once a classifier is included in the pool its learning algorithm will not need to be trained any more along the genetic algorithm execution.

Problem encoding

The solutions encoding task is performed in the second phase. There are different ways to represent the solutions of a problem for a genetic algorithm (e.g. binary codification, decimal, hexadecimal, etc). In order to represent the candidate solutions or individuals in *GA-Ensemble*, it has been decided to use a binary representation since it allows the use of canonical GA's. The canonical GA is the original form proposed by Holland (Holland 1975; 1992) and has a stronger mathematical foundation (Goldberg 1989; Mitchell 1996).

In the *GA-Ensemble* approach, each individual in the population of the genetic algorithm represents a different ensemble. The size of the chromosome, that represents an individual, is given by the number of available classifiers in the pool and an extra gene than indicates the method that is going to be used to combine the classifiers (see Figure 2). For example in this work, in which we have used fifteen classifiers, each individual is sixteen genes long.

When *GA-Ensemble* makes use of a metaclassifier to combine the base-level classifiers, a multi-response model tree (Frank *et al.* 1998) is always used, so the meta-level algorithm is never included in the search space.

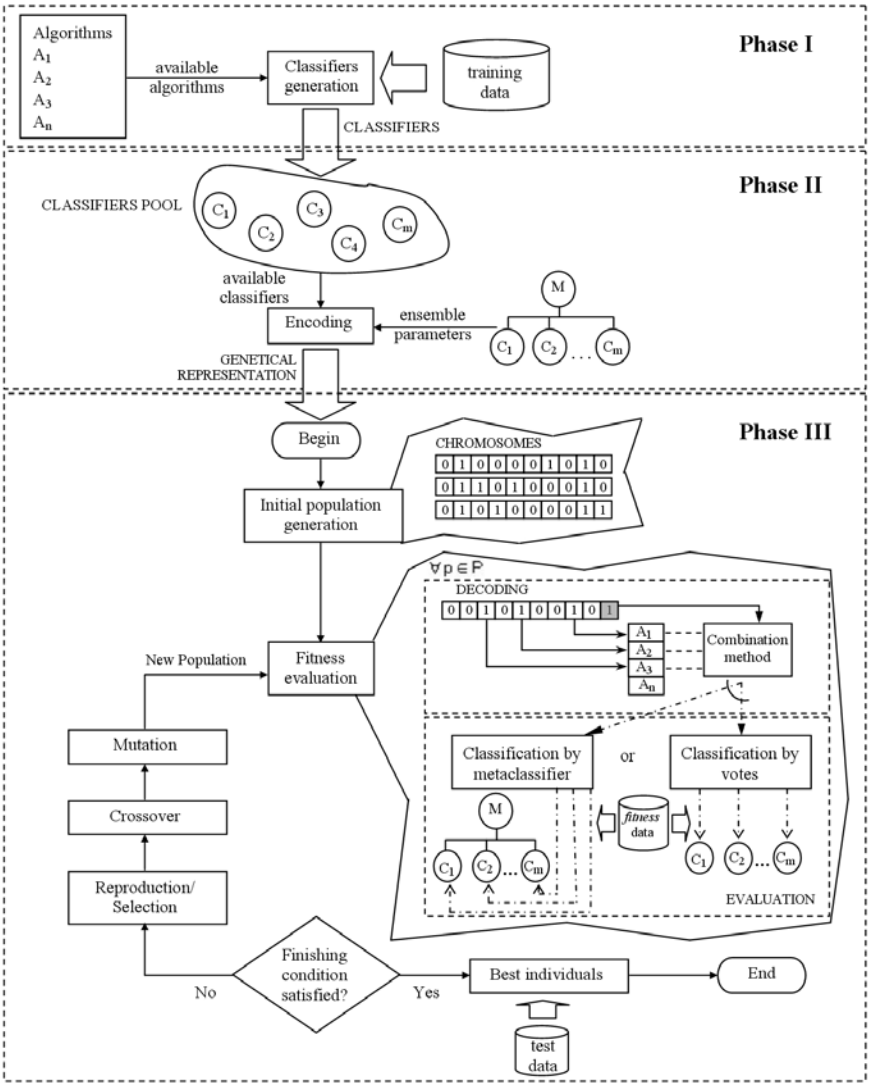

Figure 1: General framework of *GA-Ensemble*.

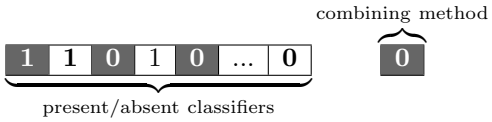

Figure 2: Ensemble codification.

Genetic search

In the last phase a standard genetic algorithm is applied so as to evaluate each individual of the population and obtain the ensemble optimal configuration as output of the system. We have used a genetic algorithm because it is proved that they can handle large and complex (multimodal) search spaces, not requiring strong assumptions on the function to be optimized.

In order to evaluate each individual, we used the classification accuracy of the ensemble encoded in the individual over a defined fold cross-validation as the fitness function. The GA parameters are detailed in section "GA parameters".

Experimental setup

In these preliminary experiments we have used several algorithms implemented in the *Waikato Environment for Knowledge Analysis* - WEKA - (Witten & Frank 2000). This tool includes all the algorithms used to generate the base classifiers and the ensemble generation algorithms.

Datasets

For the experimental test of this approach we used 10 data sets from the well-know repository of machine learning databases at UCI (Blake & Merz 1998) (see Table 1). These datasets have been used widely in other comparative works.

Dataset	Attributes	Instances	Classes
australian	14	690	2
balance	5	625	3
car	6	1728	4
chess	36	3196	2
diabetes	8	768	2
glass	9	214	6
hepatitis	19	155	2
hypo	25	3163	2
ionosphere	35	351	2
vehicle	19	846	4

Table 1: Datasets descriptions

Learning algorithms

In order to obtain the optimal combination of classifiers, 15 learning algorithms have been used to generate the base level classifiers, some of them with different parameters:

- A probabilistic *Naive Bayes* classifier (John & Langley 1995)
- PART (Frank & Witten 1998). Its creates decision list from partial pruning decision trees generated using C4.5 heuristic.
- C4.5 (Quinlan 1993). It generates decision trees.
- C4.5 using unpruned tree.
- *Decision Stump* (Iba & Langley 1992). It is an algorithm that generates one level decision tree.
- *Decision Table* (Kohavi 1995). It is a simple classifier that use the mayority class.
- Decision Table using the nearest neighbour instead of global table majority.
- Classification Via Regression method. It classifies using regression methods. The M5 algorithm (Quinlan 1992) is used as the regression method.
- *Random Forest* (Breiman 2001). This algorithm constructs a Random Forest that forms combining a great number of unpruned decision trees.
- *Random Tree* (Witten & Frank 2000). This algorithm constructs a tree considering K random attributes at each node. It does not carry out any pruning.
- VFI (Demiroz & Guvenir 1997). It is an algorithm that generates a classifier that classifies an instance based on features intervals.
- *Conjunctive Rule*. This algorithm generates a simple conjunctive rules classifier.
- *JRip* (Cohen 1995). An algorithm that generates propositional rules.
- *Nnge* (Martin 1995). It is a nearest neighbor algorithm that use non-nested generalized exemplars.
- *HyperPipes* (Witten & Frank 2000). It generates a classifier that constructs a HyperPipe for each category, which contains all the points of that category.

Ensemble approaches

The ensembles of classifiers found by our approach, *GA-Ensemble* (GAE), have been compared with three well-know ensemble methods: Bagging, Boosting and *StackingC* (Seewald 2002) (StC). The base level classifiers of the *StackingC* method have been included in the comparative for a better evaluation.

- StC: *Stacking* with a reduced set of meta-level attributes. We carried out experiments with two different numbers of base level classifiers, three and six. It has been used the same three base-level classifiers used in (Dzeroski & Zenko 2002): C4.5, IBk and Naive Bayes. In the second set of experiments we added the K*, Decision Table and *multi-response model tree* (MMT).
- Bagging: method for generating multiple versions of a predictor and using them to get an aggregated predictor. We used the C4.5 algorithm as base algorithm.
- Boosting: Method for improving the accuracy of any given learning algorithm producing a series of sequential classifiers. The basis for boosting is the C4.5 algorithm.

GA parameters

The parameters used for the genetic algorithm in the preliminary experiments are shown in Table 2.

PARAMETER	VALUE
POPULATION SIZE	50
GENERATIONS	30
ELITE RATE	0.1
CULL RATE	0.05
MUTATION RATE	0.1

Table 2: GA parameters

Evaluating and comparing algorithms

In order to evaluate our approach, we applied the genetic algorithm three times in every search process. The best individual obtained was saved. This individual encoded an ensemble configuration, defining the combining method and base-level classifiers. It was then compared with the other ensemble approaches using a 10-fold cross-validation. Weka's paired t-test was used to test for significance with the other algorithms ($\alpha = 0.05$).

Empirical results

Table 3 shows the results obtained for *GA-Ensemble*, Bagging, Boosting and StC (with 3 and 6 base level classifiers). In 4 of 10 domains, *GA-Ensemble* gets the best results, although differences with the best other systems are not statistically significant. If we add the differences between *GA-Ensemble* and the other systems, *GA-Ensemble* gets a relative improvement (RAI) of 30.29% with Boosting, 33,99% with Bagging and 23.53% and 3.65% with StC (3 and 6 BLC). With regard to the base level classifiers *GA-Ensemble* gets a relative improvement of 52,7% with IBk, 51,61% with

Domain	GAE	Boosting	Bagging	IBk	K*	MMT	DT	C4.5	NaiveBayes	StC(3)	StC(6)
australian	**86.66**	85.94	86.37	82.31	79.56	**86.66**	84.05	84.78	77.82	85.36	86.52
balance	**96.31**	78.07	81.59	85.58	88.79	88.15	74.87	77.91	90.39	89.59	89.11
car	**97.68**	95.89	92.59	92.99	87.32	96.64	93.57	91.72	85.58	92.88	96.81
chess	99.56	**99.68**	99.40	96.18	96.87	99.24	97.55	99.40	88.10	99.40	99.40
diabetes	76.81	72.77	76.04	70.83	70.30	**76.82**	73.44	75.65	75.91	76.43	75.90
glass	72.90	73.33	71.51	69.13	75.67	71.06	70.47	61.23	49.54	68.70	**76.16**
hepatitis	84.50	80.54	79.95	81.29	81.91	85.12	79.12	76.58	85.24	84.54	**84.58**
hypo	99.05	99.05	99.11	96.90	97.85	**99.17**	98.95	**99.17**	97.81	**99.17**	99.11
ionosphere	**93.42**	92.30	90.85	87.15	84.60	89.45	90.57	90.57	81.76	92.86	92
vehicle	77.91	76.94	73.40	69.74	70.32	79.33	65.49	72.57	45.38	72.34	**81.56**
RAI	—	30.29	33.99	52.7	51.61	13.16	56.72	55.22	107.27	23.53	3.65

Table 3: Accuracy of StC with different number of base-level classifiers (3 and 6), the base-level classifiers, Boosting, Bagging and *GA-Ensemble*

K*, 13,16% with MMT, 56,72% with DecisionTable and 55,22% with C4.5.

It is also interesting to consider the execution time, since one of the objectives of this work was to reduce the execution time of *GA-Stacking*. *GA-Ensemble* has required a much shorter time than *GA-Stacking* to finish the execution of the genetic algorithm. Considering that *GA-Ensemble* has to generate the base-level classifiers once per each execution of the genetic algorithm and *GA-Stacking* had to train all classifiers to calculate the fitness function of each individual of each generation, the improvement in time is obvious.

Using a function to express this improvement, the time required by *GA-Stacking* would be:

$T_{GA-Stacking} = ((Ta_{BC} * Na_{CI}) + Ta_{MC}) * N_i * N_g$

where Ta_{BC} is the time average to train a base classifier, Na_{CI} is the number average of classifiers which are encoded in an individual, Ta_{MC} is the time average to train a meta-classifier, N_i is the number of individuals in the population and N_g the number of generations of the genetic algorithm.

Whereas the function to express the time required by GA-Ensemble would be:

$T_{GA-Ensemble} = (Ta_{BC} * Tn_{BC}) + ((Ta_{MC} * AI_m) * N_i * N_g)$

where Tn_{BC} is the total number of base classifiers and AI_m is the average of individuals in the population using a meta-classifier as ensemble method.

Conclusions and future works

In this paper, we have presented a preliminary version of *GA-Ensemble*, an approach to find good ensembles configurations for a specific domain by means of genetic search. This preliminary version of *GA-Ensemble* is based on a previous genetic approach, *GA-Stacking*, which was able to generate, given a domain, a good *Stacking* configuration. *GA-Ensemble* not only determines which base classifiers must be present, but also the combining method of these classifiers. One of advantages of *GA-Ensemble* inherited from *GA-Stacking* is its flexibility and extensibility, since it can use new learning algorithms as soon as they are invented being not restricted by *a priori* assumptions. But its main advantage may be its improvement in time, requiring a much shorter time than our previous approach, *GA-Stacking*. Empirical results in domains currently used in this field show that *GA-Ensemble* is comparable to the best results reported so far, and it is never significantly worse than the other systems tested (with the advantage that parameters such as the number of base classifiers need not be specified in advance). With respect to accuracy, if we add the relative improvements over the other systems across all the domains tested, positive differences are always obtained, and quite large in some cases. On the other hand, although *GA-Ensemble* has improved the execution time with respect to *GA-Stacking*, it may still require a long time depending on the used domain; but for most domains this is not crucial, given that usually classifiers do not require to be constructed in real time.

But accuracy is not always the only aspect used to evaluate ensembles of classifiers, although it is usually the only issue considered relevant. Configuration size, classification speed, etc, can also be the significant qualities in some domains. Further work in this preliminary algorithm can include the research of the flexibility of *GA-Ensemble*. For instance, adding pressure to reduce the number of base-classifiers. Also, it would be a good enhancement to add information to the chromosome about the meta-level classifier to be used, as in *GA-Stacking*.

References

Altinçay, H. 2004. Optimal resampling and classifier prototype selection in classifier ensembles using genetic algorithms. *Pattern Analysis and Applications* 7(1):285–295.

Blake, C., and Merz, C. 1998. UCI repository of machine learning databases. databases http://www.ics.uci.edu/∼mlearn/MLRepository.html.

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

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

Cherkauer, K. 1996. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models*, 15–21.

Cohen, W. W. 1995. Fast effective rule induction. In *Ma-*

chine Learning: Proceedings of the Twelfth International Conference.

de Oliveira, L. E. S.; Morita, M. E.; Sabourin, R.; and Bortolozzi, F. 2005. Multi-objective genetic algorithms to create ensemble of classifiers. In *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005 Proceedings*, number 3410 in Lecture Notes in Computer Science.

Demiroz, G., and Guvenir, H. A. 1997. Classification by voting feature intervals. In *Proceedings of the 9th European Conference on Machine Learning*, 85–92.

Dietterich, T. G., and Bakiri, G. 1995. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2:263–286.

Dietterich, T. G. 1997. Machine-learning research:four current directions. *AI Magazine* 18(4):97–136.

Dietterich, T. G. 2000. Ensemble methods in machine learning. In Kittler, J., and Roli, F., eds., *Multiple Classifiers Systems: first international workshop; proceedings /MCS 2000*, volume 1857 of *Lecture Notes in Computer Science*, 1–15. Cagliari, Italy: Springer.

Dzeroski, S., and Zenko, B. 2002. Stacking with multi-response model trees. In Fabio Roli, J. K., ed., *Proceedings of Multiple Classifier Systems, Third International Workshop, MCS 2002*, Lecture Notes in Computer Science. Cagliari, Italy: Springer.

Dzeroski, S., and Zenko, B. 2004. Is combining classifiers better than selecting the best one? *Machine Learning* 54(3):255–273.

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

Frank, E.; Wang, Y.; Inglis, S.; Holmes, G.; and Witten, I. 1998. Using model trees for classification. *Machine Learning* 2(32):63–76.

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

Goldberg, D. E. 1989. *Genetic Algorithms in search, optimization, and machine learning*. Addison-Wesley.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press.

Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems*. MIT Press, 2ł edition.

Iba, W., and Langley, P. 1992. Induction of one-level decision trees. In *Proceedings of the Ninth International Conference on Machine Learning*, 233–240. Morgan Kaufmann.

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

Kohavi, R. 1995. The power of decision tables. In *Proceedings of the Eighth European Conference on Machine Learning*.

Kolen, J. F., and Pollack, J. B. 1991. Back propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems*, 860–867.

Ledezma, A.; Aler, R.; and Borrajo, D. 2001. *Data Mining: a Heuristic Approach*. Idea Group Publishing. chapter Heuristic Search Based Stacking of Classifiers, 54–67.

Ledezma, A.; Aler, R.; Sanchis, A.; and Borrajo, D. 2004. Empirical evaluation of optimized stacking configurations. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)*.

Martin, B. 1995. Instance-based learning : Nearest neighbor with generalization. Master's thesis, University of Waikato.

Mitchell, M. 1996. *An Introduction to Genetic Algorithms*. MIT Press.

Quinlan, J. 1992. Learning with continuous classes. In *Proceedings of the fifth Australian Joint Conference on Artificial Intelligence*, 343–348. Singapore: World Scientific.

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

Seewald, A. K. 2002. How to make stacking better and faster while also taking care of an unknown weakness. In Claude Sammut, A. G. H., ed., *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002)*. Sidney, Australia: Morgan Kaufmann.

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

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

Zhou, Z.-H.; Wu, J.-X.; Jiang, Y.; and Chen, S.-F. 2001. Genetic algorithm based selective neural network ensemble. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, volume 2.