Empirical Evaluation of Optimized Stacking Configurations

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

Abstract

Stacking is one of the most used techniques for combining classifiers and improve prediction accuracy. Early research in stacking showed that selecting the right classifiers, their parameters and the metaclassifiers was the main bottleneck for its use. Most of the research on this topic selects by hand the right combination of classifiers and their parameters. Instead of starting from these initial strong assumptions, our approach uses genetic algorithms to search for good stacking configurations. Since this can lead to overfitting, one of the goals of this paper is to evaluate empirically the overall efficiency of the approach. A second goal is to compare our approach with current best stacking building techniques. The results show that our approach finds stacking configurations that, in the worst case, perform as well as the best techniques, with the advantage of not having to set up manually the structure of the stacking system.

1. Introduction

One of the currently favored lines of research in Machine Learning is the combination of classifiers to improve classification accuracy [9]. This approach is known as *ensembles of classifiers* in the supervised learning area. The main idea behind ensembles, is that they are often much more accurate than the individual classifiers that make them up.

Research in this area focused, generally, in generating the ensemble's members applying a single learning algorithm [10]. In order to generate different classifiers, there are several methods that can be grouped in: sub-sampling the training examples (e.g. bagging [3] and boosting [15]); manipulating the input features [5]; manipulating the output target (e.g. ECOC [11]); and injecting randomness in the learning algorithm [21]. Once the classifiers have been generated, they are combined, in most cases by voting or weighted voting.

Other research in the area use different learning algorithms over a dataset to generate the members of the ensemble. One example of this approach is stacking [31]. Stacking uses an algorithm to learn how to combine the outputs of a set of classifiers that have been obtained by different learning algorithms. Our research focuses on this approach.

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. 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. But, when the search space is big, it is difficult to find the best stacking configuration. In a previous work [22], we presented an approach based on genetic algorithms (GA) [17]. A somewhat related approach uses GAs for ensembles of neural networks, but only to determine the right number of ensemble members and in the context of homogeneous ensembles, not heterogeneous classifiers generation algorithms like Stacking [32].

There are also many variants of the basic stacking algorithm. The methods that have reported the best results are stacking with multi-response model trees (SMRMT) [12] and stacking with multi-response linear regression and a reduced set of attributes (StC) [26]. In this paper, we continue our line of thought, based on not imposing strong assumptions on the final stacking configuration, and letting the GA find a good one freely. In particular, we have extended our GA-stacking approach by enlarging the search space of stacking configurations, taking into account not only the learning algorithms but also their parameters. This is easy to achieve in a genetic approach, by coding the parameters in the GA individual. We then compare results with two recent stacking state-of-the-art approaches. We show that our approach is able to find a stacking configuration that performs comparable and in some case better than those state-of-art stacking systems.

This paper is organized as follows. Section 2 gives some background on stacking and recent stacking variants. Section 3 introduces the stacking configuration approach, GA-stacking. Sections 4 and 5 describe the experimental setup

Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004)
1082 3409/04 $20.00 © 2004 **IEEE**

and the results, respectively. Finally Section 6 draws some conclusions.

2. Stacking Approaches

In this Section, we first describe the stacking framework and then we give a brief description of some recent work on stacking.

2.1. Stacked Generalization

Stacking is the abbreviation to refer to Stacked Generalization [31]. The main idea of stacking is to combine classifiers from different learners such as decision trees, instance-based learners, etc. 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 the meta learner. The meta learner (or level-1 model), tries to acquire, using a learning algorithm, how the decisions of the base classifiers (or level-0 models) should be combined to obtain the final classification. This is achieved by a cross-validation-like process, as described in [31].

2.2. Recent Advances

Recent work on stacking addresses the stacking configuration problem: what algorithm and features are to be used in the meta-level. We give a brief review of these recent work here.

One necessary condition to create a good ensemble of classifiers is that base-level classifiers error predictions are uncorrelated [16]. In [24] a variant of stacking is proposed that uses correspondence analysis in order to detect correlations between the base-level classifiers. Once dependencies have been removed from original meta-level space, a nearest neighbor method (meta-level algorithm) is then applied over the result features space.

In [28], they use probability distributions outputs from level-0 models instead of a simple class prediction as meta-level attributes. By using probability distributions as meta-level data, the authors argue that both, prediction and confidence of the base-level classifiers are used. In order to use this type of meta-level data, the authors proposed to use the *multi-response linear regression* technique (MLR) as the meta-level algorithm.

Another variant of stacking [27] creates a meta-level classifier for each level-0 classifier. The learning task for each level-1 classifier is to predict whether the level-0 classifier prediction will be correct. The meta-level data is composed of base-level attributes and the class values are *correct* or *incorrect*. The predictions of those classifiers that are combined by summing up the predicted probability distribution.

In [29] a new variant of stacking is described that uses a new learning method in the meta-level. This method, denoted *meta decision trees* (MDTs), replaces class-value predictions in its leaf nodes by the base level-classifiers. The meta-level data is composed of properties of the probability distributions that reflect the confidence of the base-level classifiers, like entropy and maximum probability, rather than the distributions themselves. Based on these properties, small MDT's are generated.

Based on stacking with MLR (SMLR), [26] proposed to reduce the number of meta-level attributes independently of the number of classes, in order to overcome a weakness of SMLR in domains with more than two classes. This method is called *StackingC* (StC).

Dzeroski and Zenko [13] proposed two new variants of stacking. First, based on stacking with MLR, they propose to extend the set of meta-level features augmented with the probability distribution multiplied by the maximum probability and the entropies of the probability distributions. On the other hand, they propose another extension of SMLR in which they use model tree induction instead of linear regression as the meta-level algorithm. This method is called stacking with multi-response model trees (SMRMT).

Comparing stacking approaches overall, SCANN, MDTs, SMLR and SelectBest (selecting the best classifier with cross-validation) seem to perform at about the same level [13]. Moreover, in their work, they concluded that stacking with multi-response model trees performs better than existing stacking approaches, including StC, and selecting the best classifier from the ensemble by cross-validation.

One of the main conclusions of all this previous work is that there are many contradictory results and there is no consensus on which combination of classifiers is the best one.

The main differences of previous work with respect to our approach are that we do not select "a priori":

- which meta-classifier to use
- the parameters of the meta-classifier
- the number of base classifiers
- which of the available base classifiers to use
- the parameters of these base classifiers

The main advantage of our approach is its flexibility and its no "a-priori" commitments. The system is very extensible. It can benefit from new classifiers, since they can be

easily incorporated into the pool of available classifiers together with their parameters, and coded in the GA-stacking chromosome. Another advantage of our approach is that it is dataset dependent, so we also adapt to dataset biases and features, while the rest of approaches use the same configuration for all datasets.

In order to evaluate our approach, we compare our results with the state-of-the-art, most recent, stacking approaches. In other words, StC and SMRMT.

3. GA-Stacking Approach

In this section we describe the approach taken by GA-stacking to build stacking systems. Given that stacked generalization is composed of a set of classifiers from a different set of learning algorithms, the question is: what algorithm should be used to generate the level-0 and the level-1 models?. As we detail in Section 2.2, most work on stacking focuses in the selection of the meta-level data and algorithm. In the GA-stacking approach, a genetic algorithm is 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 a population in GA, represents a different stacking system configuration.

We have chosen GA's to search for good stacking configurations because they have proven useful in the past for similar tasks as the one currently at hand: large and complex (multimodal) search spaces, not requiring strong assumptions on the function to be optimized. Also, any structure can be evolved in principle, as far as it can be coded as a fixed-length bit-string (at least, in canonical GA's).

In the previous GA-stacking version [22], the search carried out by the genetic algorithm was limited to find a stacking configuration with four base-level classifiers and the meta-level algorithm from a set of six available algorithms. In this paper, we extend the search space enlarging the number of possible base-level classifiers to ten and including the parameters of the algorithms in the search, which generates a very different and much richer search space. Moreover, in this work, we use fifteen possible algorithms to generate the base-level and meta-level classifiers.

The application of genetic algorithms to optimization problems requires that the following are defined:

1. the representation of the individuals (candidate solutions)
2. the fitness function that is used to evaluate the individuals
3. the selection-scheme (e.g., selection proportional to fitness)
4. the genetic operators that will be used to evolve the individuals
5. the parameters of the GA (e.g., population size, generations, probability of crossover, etc.)

In relation to the individual representation, each individual is a chromosome with 66 genes. They contain the codes of the ten algorithms to build the level-0 classifiers (first 60 genes) and the meta-classifier (last 6 genes). Each classifier is coded in six genes in which the first gene represents the classifier's name and the remaining genes represent the classifiers's parameters (see Figure 1). The gene size depends on the coded parameter. For example the gene that represent the classifiers's name has a size of four bits.

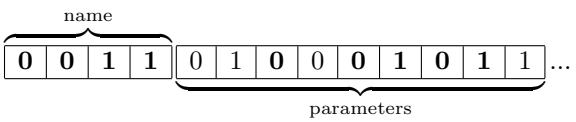


Figure 1. Classifier codification.

In order to evaluate each individual, we used the classification accuracy of the stacking system over a two fold cross-validation as the fitness function. Cross-validation is used to prevent over-fitting. The selection scheme and GA parameters are detailed in section 4.4.

4. Experimental Setup

In this work we have used the algorithms implemented in the Waikato Enviroment for Knowledge Analysis - WEKA - [30]. This includes all the base classifiers and stacking approaches.

4.1. Data Sets

For the experimental test of the stacking system configurations we have used 18 data sets from the well know repository of machine learning databases at UCI [2]. These datasets have been used widely in other stacking comparative work. In order to evaluate the GA-stacking approach in a different set of instances than the GA training set, we divided each data set into two parts: part A used to evaluate and compare stacking approaches and part B used to find the optimal stacking configuration. Table 1 shows the data sets characteristics.

4.2. Learning Algorithms

In order to obtain the optimal stacking configuration, 15 learning algorithms have been used to generate the level-0 classifiers and the meta-classifier:

Dataset	Attributes	Instances	Part A	Part B	Classes
australian	14	690	345	345	2
balance	4	625	312	313	3
breast-w	9	699	349	350	2
car	6	1728	1382	346	4
chess	36	3196	2876	320	2
diabetes	8	768	384	384	2
echo	6	132	66	66	2
german	20	1000	500	500	2
glass	9	214	107	107	6
heart	13	270	135	135	2
hepatitis	19	155	77	78	2
hypo	25	3163	2846	317	2
image	19	2310	1848	462	7
ionosphere	34	351	175	176	2
iris	4	150	75	75	3
soya	35	683	341	342	19
vote	16	435	217	218	2
wine	13	178	89	89	3

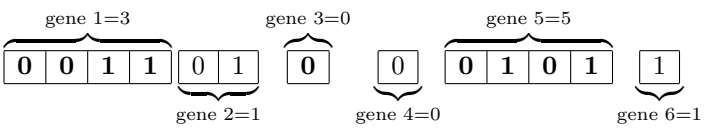
Table 1. Datasets descriptions

- C4.5 [25]. It generates decision trees
- A probabilistic Naive Bayes classifier [19]
- IBk. This is Aha's instance based learning algorithm [1]
- PART [14]. It forms a decision list from pruned partial decision trees generated using the C4.5 heuristic
- Decision Stump [18]. A one level decision tree.
- Decision Table [20]. A simple decision table majority classifier
- Random Forest [4]. It constructs a Random Forest that is formed by combining a large number of un-pruned trees.
- Random Tree. It construct a tree that considers K random features at each node
- MLR or MRMT [12]. The multi-response linear regression or multi-response model tree. The selection of a linear regression or a model tree is a parameter of the classification via the regression method implemented in Weka.
- K* [6]. An instance-based learning algorithm which uses an entropy based distance measure.
- VFI [8]. It is a voting feature interval classifier.
- Conjunctive Rule. It is a single conjunctive rule learner that can predict with numeric and nominal class labels.
- JRip [7]. A propositional rule learner.
- Nnge [23]. Nearest neighbor like algorithm using non-nested generalized exemplars.
- Hyper Pipes [30]. It is classifier that builds a HyperPipe for each category that contains all points of that category.

As stated in the previous section, the GA chromosome also contains the parameters of the classifiers. We have used all the parameters that Weka provides for each one of the latter learning algorithms. For example, Weka's C4.5 (j48) has the following parameters:

- -U: Use unpruned tree
- -C: Confidence. Set confidence threshold for pruning (Default: 0.25)
- -M: Number. Set minimum number of instances per leaf (Default: 2)
- -R: Use reduced error pruning. No subtree raising is performed
- -N: Number. Set number of folds for reduced error pruning. One fold is used as the pruning set (Default: 3)
- -B: Use binary splits for nominal attributes
- -S: Don't perform subtree raising
- -L: Do not clean up after the tree has been built
- -A: If set, Laplace smoothing is used for predicted probabilities

Therefore, we have coded the parameters of each learning technique into a fixed-length gene (see Figure 2).



Gene	Value	Option
1	3	c4.5
2	1	-R
3	0	
4	0	
5	5	-C 0.35
6	1	-A

Figure 2. Codification example.

4.3. Stacking Approaches

We have compared the stacking configurations found by GA-stacking (GA-S) with recent two state-of-art stacking approaches:

- SMRMT: stacking with multi-response model trees as we described in Section 2.2. Since a recent study used a different number of base classifiers [12], we carried out experiments with two different numbers of base-level classifiers, three and six. We use the same three

base-level classifiers that have been used in [12], C4.5, IBk and Naive Bayes. In the second set of experiments we add the K*, Decision Table and MLR. In [12] they used these base-level algorithms plus a simple decision kernel density that is not implemented in Weka version 3.4.

- StC: stacking with a reduced set of meta-level attributes as we described in section 2.2. We carried out experiments with the same three base-level algorithms used in SMMRT and five base-level algorithms as described in [26], excluding the kernel density estimator, mentioned in the previous item.

4.4. GA Parameters

The parameters used for the GA in the experiments are shown in Table 2.

PARAMETER	VALUE
POPULATION SIZE	100
GENERATIONS	50
ELITE RATE	0.1
CULL RATE	0.4
MUTATION RATE	0.1

Table 2. GA parameters

4.5. Evaluating and Comparing Algorithms

In order to evaluate our approach, we applied GA-stacking three times in every domain with dataset B. The best individual obtained was saved. This individual is not a stacking system, but just a configuration of meta-level and base-level classifiers, which has to be trained in order to perform classification tasks. This individual was then compared with the other stacking approaches by repeating 10 times a stratified 10-fold cross-validation with dataset A. Weka's paired t-test was used to test for significance with the other stacking approaches ($\alpha = 0.05$).

5. Empirical Results

5.1. GA-Stacking approaches comparison

In this subsection we compare our current GA-S approach (which looks for the classifier parameters) with our previous GA-S (which only found the classifiers and used their default parameters set in Weka). In both cases the number of possible base classifiers to be considered by the GA is 10. Table 5.1 displays the results of the test in five domains.

Domain	GA-S without parameters	GA-S with parameters
australian	**87.64**	87.38
breast-w	95.71	95.71
car	97.33	**97.55**
diabetes	74.03	**74.36**
german	**75.96**	74.04
ionosphere	88.43	**90.62**

Table 3. Comparison of GA-Stacking with and without using parameters.

Both GA-S were run under the same conditions. Although, Table 5.1 shows some differences, they are not statistically significant (WEKA's t-test was used for this purpose, $\alpha = 0.05$).

5.2. Main Results

Table 4 shows the results obtained for GA-stacking, StC (with 3 and 5 base level classifiers), and SMRMT (with 3 and 6 base level classifiers). In 9 out of 18 domains, GA-stacking gets the best results, although differences with the best other systems are not statistically significant. Only those results marked with * are significantly worse. It is noticeable that GA-stacking is never significantly worse (although this is also true of SMRMT with 6 base classifiers). If we add the differences between GA-stacking and the other systems, GA-stacking gets a relative improvement of 20.46% and 11.31% with StC (3 and 5 BLC) and 16.86% and 6.27% with SMRMT (3 and 6 BLC).

It is also interesting to consider which classifiers the GA-stacking found for the best configuration meta-level. Other researchers claim that MLR [28, 26] or MRMT [12] at the meta-level give good results regardless of the base classifiers. We have found that the meta-level classifier selected by GA-stacking is NaiveBayes in 5 domains, RandomForest (4 domains), MLR (3), IBK (3), PART (2), and Nnge (1). MRMT is never used. This does not mean that the conclusions of these researchers are wrong, but that other good alternatives exist for the meta classifier, depending on the domain, if the configuration is found by search.

6. Conclusions and Future Work

In this paper, we have presented a extended version of GA-stacking, an approach to find good stacking configurations by means of genetic search. GA-stacking not only determines which meta-level, and which (and how many) base classifiers must be present, but also their learning parameters. The main advantage of GA-stacking is its flexi-

Dataset	GA-S	StC		SMRMT	
		3-BLC	5-BLC	3-BLC	6-BLC
australian	87.11	**88.40**	88.02	88.16	87.24
balance-scale	**95.45**	90.30*	90.43*	94.26	94.97
breast	95.27	94.65	95.30	94.27	**95.53**
car	96.98	92.06*	94.35*	95.59*	**97.20**
chess	**99.40**	99.18	99.18	99.19	99.22
diabetes	**74.83**	73.79	**74.83**	73.73	74.05
echo	89.05	**90.22**	86.02	89.88	89.05
german	72.52	75.00	**76.42**	74.34	75.68
glass	**74.99**	67.97	71.38	66.13*	71.87
heart	80.38	80.40	81.29	**81.90**	80.79
hepatitis	**79.75**	79.55	79.57	79.07	77.63
hypho	99.06	**99.30**	99.29	99.29	99.29
images	**98.05**	96.73*	97.61	96.63*	97.74
ionosphere	**90.91**	88.80	89.32	88.63	87.85
iris	94.21	94.07	**95.29**	94.63	94.70
soybean	**93.87**	91.58	91.79	90.67	91.26
vote	95.72	**95.76**	95.94	95.21	95.44
wine	95.61	94.94	95.29	94.72	**96.85**

Table 4. Accuracy of StC and SMRMT with different number of base-level classifiers (BLC) and GA-stacking

bility and extensibility. It can use new learning algorithms as soon as they are invented and it is not restricted by "a priori" assumptions. Empirical results in domains currently used in this field show that GA-stacking is comparable to the best results reported so far, and it is never significantly worse than the other systems tested (with the advantage that paremeters such as the number of base classifiers, or the algorithms available to be used as base, need not be specified in advance). With respect to accuracy, if we add the relative improvements over the other systems across all the domains, positive differences are always obtained, quite large in some cases. Therefore, if accuracy is very important for a given task, we believe GA-stacking should be used. On the other hand, GA-stacking requires a longer execution time than the other approaches, because several generations of individuals must be evaluated in order to obtain a good individual. For most domains this is not crucial, given that most classification tasks do not require to work in a short real time.

On the other hand, our research has shown that giving more freedom to the search for a stacking configuration does not always pay off in terms of accuracy. For instance, letting GA-stacking to search for the parameters of the base classifiers (as well as the classifiers themselves) do not improve accuracy significantly (but results are not worse either).

But accuracy is not always the only aspect used to evaluate stacking configurations, although it is usually the only aspect considered in stacking research. Configuration size, on-line classification speed, etc, can also be the relevant qualities in some domains. In the future, we plan to investigate the flexibility of GA-stacking so that these qualities are considered. For instance, we intend to add selective pressure towards accurate, but also simple (few base classifiers) stacking configurations. Also, we would like to add information to the chromosome about the meta-level data to be used, so that the stacking configuration can use the most appropriate representation for each domain.

References

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

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

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

[4] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] K. Cherkauer. 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*, pages 15–21, 1996.

[6] John G. Cleary and Leonard E. Trigg. K*: an instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine Learning*, pages 108–114, 1995.

[7] William W. Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.

[8] Gulsen Demiroz and H. Altay Guvenir. Classification by voting feature intervals. In *Proceedings of the 9th European Conference on Machine Learning*, pages 85–92, 1997.

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

[10] Thomas G. Dietterich. Ensemble methods in machine learning. In Josef Kittler and Fabio Roli, editors, *Multiple Classifiers Systems: first international workshop; proceedings /MCS 2000*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15, Cagliari, Italy, June 2000. Springer.

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

[12] Saso Dzeroski and Bernard Zenko. Stacking with multi-response model trees. In Josef Kittler Fabio Roli, editor, *Proceedings of Multiple Classifier Systems, Third International Workshop, MCS 2002*, Lecture Notes in Computer Science, Cagliari, Italy, 2002. Springer.

[13] Saso Dzeroski and Bernard Zenko. Is combining classifiers better than selecting the best one? *Machine Learning*, (54):255–273, 2004.

[14] 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.

[15] 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.

[16] L. Hansen and P. Salamon. Neural network emsembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[17] John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

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

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

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

[21] John F. Kolen and Jordan B. Pollack. Back propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems*, pages 860–867, 1991.

[22] Agapito Ledezma, Ricardo Aler, and Daniel Borrajo. *Data Mining: a Heuristic Approach*, chapter Heuristic Search Based Stacking of Classifiers. Idea Group Publishing, 2001.

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

[24] Christopher J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1-2):33–58, 1999.

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

[26] Alexander K. Seewald. How to make stacking better and faster while also taking care of an unknown weakness. In Achim G. Hoffmann Claude Sammut, editor, *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002)*, Sidney, Australia, July 2002. Morgan Kaufmann.

[27] Alexander K. Seewald and Johannes Fürnkranz. An evaluation of grading classifiers. In Frank Hoffmann, David J. Hand, Niall M. Adams, Douglas H. Fisher, and Gabriela Guimarães, editors, *Advances in Intelligent Data Analysis, 4th International Conference, IDA 2001, Proceedings*, Lecture Notes in Computer Science, pages 115–124, 2001.

[28] Kai Ming Ting and Ian H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10:271–289, 1999.

[29] Ljupco Todorovski and Saso Dzeroski. Combining multiple models with meta decision trees. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 54–64, 2000.

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

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

[32] J. Wu Z.-H. Zhou and W. Tang. Ensembling neural networks: Many could be better than al. *Artificial Intelligence*, 137(1-2), 2002.