Grading Classifiers

Alexander K. Seewald and **Johannes Fürnkranz**
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Wien, Austria
[alexsee,juffi]@ai.univie.ac.at

Abstract

In this paper, we introduce grading, a novel meta-classification scheme. While stacking uses the predictions of the base classifiers as meta-level attributes, we use "graded" predictions (i.e., predictions that have been marked as correct or incorrect) as meta-level classes. For each base classifier, one meta classifier is learned whose task is to predict when the base classifier will err. Hence, just like stacking may be viewed as a generalization of voting, grading may be viewed as a generalization of selection by cross-validation and therefore fills a conceptual gap in the space of meta-classification schemes. Grading may also be interpreted as a technique for turning the error-characterizing technique introduced by Bay and Pazzani (2000) into a powerful learning algorithm by resorting to an ensemble of meta-classifiers. Our experimental evaluation shows that this step results in a performance gain that is quite comparable to that achieved by stacking, while both, grading and stacking outperform their simpler counter-parts voting and selection by cross-validation.

1 Introduction

When faced with the decision "Which algorithm will be most accurate on my classification problem?", the predominant approach is to estimate the accuracy of the candidate algorithms on the problem and select the one that appears to be most accurate. Schaffer (1993) has investigated this approach in a small study with three learning algorithms on five UCI datasets. His conclusions are that on the one hand this procedure is on average better than working with a single learning algorithm, but, on the other hand, the cross-validation procedure often picks the wrong base algorithm on individual problems. This problem is expected to become more severe with an increasing number of classifiers.

As a cross-validation basically computes a prediction for each example in the training set, it was soon realized that this information could be used in more elaborate ways than simply counting the number of correct and incorrect predictions. One such meta-classification scheme is the family of *stacking* algorithms (Wolpert, 1992). The basic idea of stacking is to use the predictions of the original classifiers as attributes in a new training set that keeps the original class labels.

In this paper, we investigate another technique, which we call *grading*. The basic idea is to learn to predict for each of the original learning algorithms whether its prediction for a particular example is correct or not. We therefore train one classifier for each of the original learning algorithms on a training set that consists of the original examples with class labels that encode whether the prediction of this learner was correct on this particular example. Hence—in contrast to stacking—we leave the original examples unchanged, but instead modify the class labels.

The algorithm may also be viewed as an attempt to extend the work of Bay and Pazzani (2000) who propose to use a meta-classification scheme for characterizing model errors. Their suggestion is to learn a comprehensible theory that describes the regions of errors of a given classifier. While the step of constructing the training set for the meta classifier is basically the same as in our approach,[1] their approach is restricted to learning descriptive characterizations but cannot be directly used for improving classifier performance. The reason is that negative feedback—when the meta classifier predicts that the base classifier is wrong—only rules out the class predicted by the base classifier, but does not help to choose among the remaining classes (except, of course, for two-class problems). *Grading* may be viewed an ensemble of meta-classifiers of the type introduced by Bay and Pazzani (2000), which is—as we will see—able to significantly improve the performance of its constituent base classifiers on two-class and multi-class problems. The achieved performance gain is comparable to that of stacking.

We will describe this idea in more detail in section 2 and compare it to cross-validation, stacking, and a voting technique in section 3.

2 Grading

Figure 1(a) shows a hypothetical learning problem with n_e training examples, each of them encoded using n_a attributes x_{ij} and a class label cl_i. In our example, the number of different class labels n_l is 2 (the values t and f) but this is no principal restriction. We are now assuming that we have n_c *base classifiers* C_k, which were evaluated using some cross-validation scheme. As cross-validation ensures that each example is used as a test example exactly once, we have obtained one prediction for each classifier and for each training example (Figure 1(b)).

A straight-forward use of this prediction matrix is to let each classifier vote for a class, and predict the class that receives the most votes. Stacking, as originally described by Wolpert (1992), makes a more elaborate use of the prediction matrix. It adds the original class labels cl_i to it and uses this new data set—shown in Figure 1(d)—for training another classifier. Examples are classified by submitting them to each base classifier (trained on the entire training set) and using the predicted labels as input for the meta classifier learned from the prediction matrix. Its prediction is then used as the final prediction for the example.

By providing the true class labels as the target function, stacking provides its meta-learner with indirect feedback about the correctness of its base classifiers. This feedback can be made more explicit. Figure 1(c) shows an evaluation the base classifiers'

[1] A minor difference is that they use a hold-out set, while we use cross-validation for computing the predictions of the base classifier.

Attributes			Class
x_{11}	$\dots$	x_{1n_a}	t
x_{21}	$\dots$	x_{2n_a}	f
$\dots$	$\dots$	$\dots$	$\dots$
x_{n_e1}	$\dots$	$x_{n_en_a}$	t

(a) training set

C_1	C_2	$\dots$	C_{n_c}
t	t	$\dots$	f
f	t	$\dots$	t
$\dots$	$\dots$	$\dots$	$\dots$
f	f	$\dots$	t

(b) predictions of the classfiers

C_1	C_2	$\dots$	C_{n_c}
$+$	$+$	$\dots$	$-$
$+$	$-$	$\dots$	$-$
$\dots$	$\dots$	$\dots$	$\dots$
$-$	$-$	$\dots$	$+$

(c) "graded" predictions

C_1	C_2	$\dots$	C_{n_c}	Class
t	t	$\dots$	f	t
f	t	$\dots$	t	f
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
f	f	$\dots$	t	t

(d) training set for stacking

Attributes			Class
x_{11}	$\dots$	x_{1n_a}	$+$
x_{21}	$\dots$	x_{2n_a}	$+$
$\dots$	$\dots$	$\dots$	$\dots$
x_{n_e1}	$\dots$	$x_{n_en_a}$	$-$

$\dots$

Attributes			Class
x_{11}	$\dots$	x_{1n_a}	$-$
x_{21}	$\dots$	x_{2n_a}	$-$
$\dots$	$\dots$	$\dots$	$\dots$
x_{n_e1}	$\dots$	$x_{n_en_a}$	$+$

(e) n_c training sets for grading

Figure 1: Illustration of stacking and grading on a hypothetical situation where n_c classifiers are tried on a two-class problem with n_a attributes and n_e examples. n_l, the number of classes, is two (t and f) in the current example, but multi-class problems are processed in the same way. This illustration is slightly simplified since our implementation of stacking uses the whole class probability distributions instead of predictions. For details refer to the text.

predictions. Each entry c_{ik} in the prediction matrix is compared to the corresponding label cl_i. Correct predictions ($c_{ik} = cl_i$) are "graded" with a $+$, incorrect predictions with a $-$.

Grading makes use of these graded predictions for training a set of meta classifiers that learn to predict when the base classifier is correct. The training set for each of these meta classifiers is constructed using the graded predictions of the corresponding base classifier as new class labels for the original attributes. Thus we have n_c two-class training sets (classes $+$ and $-$), one for each base classifier (Figure 1(e)). We now train n_c level 1 classifiers, everyone of which gets exactly one of these training sets, based on the assumption that different base classifiers make different errors. Thus, every one of these level 1 classifiers tries to predict when its base classifier will err.

Note that the proportion of negatively graded examples in these datasets is simply the error rate of the corresponding base classifier as estimated by the cross-validation

procedure. Hence, while selection by cross-validation (Schaffer, 1993) simply picks the classifier corresponding to the dataset with the fewest examples of class $-$ as the classifier to be used for all test examples, grading tries to make this decision for each example separately by focussing on those base classifiers that are predicted to be correct on this example. In this sense grading may be viewed as a generalization of selection by cross-validation.

At classification time, each base classifier makes a prediction for the current example. The final prediction is derived from the predictions of those base classifiers that are predicted to be correct by the meta-classification schemes. Conflicts (several classifiers with different base-level predictions are predicted to be correct) may be resolved by voting or by making use of the confidence estimates of the base classifiers.

In our implementation, the confidence[2] of the level 1 classifier will be summed per class and afterwards normalized to yield a proper class probability distribution. In the rare case that no base classifier is predicted to be correct, all base classifiers are used with $(1 - \textit{confidence})$ as the new confidence, thus preferring those base classifiers for which the level 1 classifier is more unsure of its decision. The most probable class from the final class distribution is chosen as the final prediction. Ties are cut by choosing among all most probable classes the one which is a priori more probable, i.e. the class which occurs more frequently in the training data.

More formally, let p_{ikl} be the class probability calculated by base classifier k for class l and example i. For simplification, we write P_{ik} to mean $(p_{ik1}, p_{ik2}, \ldots p_{ikn_l})$, i.e., the vector of all class probabilities for example i and classifier k. The prediction of the base classifier k for example i $c_{ik} = \arg\max_l \{p_{ikl}\}$, i.e., the class l with maximum probability p_{ikl}.

Grading then constructs n_c training sets, one for each base classifier k, by adding the graded predictions g_{ik} as the new class information to the original dataset (g_{ik} is 1 if the base classifier k's prediction for example i was correct (graded +) and 0 otherwise). $\textit{prMeta}_{ik}$ is the probability that base classifier k will correctly predict the class of example i as estimated by meta classifier k.

From this information we compute the final probability estimate for class l and example i. In case at least one meta classifier grades its base classifier as $+$ (i.e., $\textit{prMeta}_{ik} > 0.5$), we use the following formula:[3]

$$\textit{prGrading}_{il} = \sum \{\textit{prMeta}_{ik} | c_{ik} = l \wedge \textit{prMeta}_{ik} > 0.5\}$$

Otherwise, if no base classifiers are presumed correct by the meta classifiers, we use all base classifiers in our voting.

$$\textit{prGrading}_{il} = \sum \{1 - \textit{prMeta}_{ik} | c_{ik} = l\}$$

The class with highest probability is then chosen as the final prediction $\textit{predBase}_{ik}$.

[2]We measure confidence with the meta classifiers' estimate of the probability $p(+)$ that the example is classified correctly by the corresponding base classifier. Since there are only two classes in the meta datasets, confidence for class $-$ is $p(-) = 1 - p(+)$.

[3]We have also tried to give a penalty α for cases where the base classifier predicts class l but the meta classifier considers this prediction wrong (i.e., $c_{ik} = l \wedge \textit{prMeta}_{ik} \leq 0.5$). This did not seem to work as well.

Note that Ting and Witten (1999) also recommend the use of probability distributions for stacking. Instead of merely adding the classes that are considered to be most likely by each base classifier, they suggest to add the entire n_l-dimensional class probability vector P_{ik}, yielding a meta dataset with $n_l \times n_c$ attributes (instead of only n_c for conventional stacking).

3 Empirical Evaluation

In this section, we compare our implementation of grading to stacking, voting, and selection by cross-validation. We implemented Grading in Java within the Waikato Environment for Knowledge Analysis (WEKA).[4] All other algorithms at the base and meta-level were already available within WEKA.

3.1 Datasets and Experimental Setup

For an empirical evaluation we chose twenty-six datasets from the UCI Machine Learning Repository (Blake & Merz, 1998) which are listed in Table 1. The datasets were selected arbitrarily before the start of the experiments.

All reported accuracy estimates are the average of ten ten-fold stratified cross validations, except those in Table 6 where only one ten-fold stratified cross validation was computed to evaluate the a priori motivated choice of IBk as level 1 classifier.

3.2 Base Classifiers

We evaluated each meta-classification scheme using the same six base learners, which were chosen to cover a variety of different biases in an attempt to maximize diversity.

The algorithms are:

- DecisionTable: a decision table learner
- J48: a Java port of C4.5 Release 8 (Quinlan, 1993)
- NaiveBayes: the Naive Bayes classifier using kernel density estimation for numerical attributes
- KernelDensity: a simple kernel density classifier
- MultiLinearRegression: a multi-class learner based on linear regression, which tries to separate each class from all other classes by linear regression (*multi-response linear regression*)
- KStar: the K* instance-based learner (Cleary & Trigg, 1995)

All algorithms are implemented in WEKA Release 3.1.8. They all return a class probability distribution, i.e., they do not predict a single class, but give probability estimates for each possible class.

[4]The Java source code of WEKA has been made available at `www.cs.waikato.ac.nz`, see also (Witten & Frank, 2000)

Table 1: The used datasets with number of classes and examples, discrete and continuous attributes, baseline accuracy (%) and a priori entropy in bits per example (Kononenko & Bratko, 1991).

Dataset	classes	examples	discrete	continuous	baseline	entropy
audiology	24	226	69	0	25.22	3.51
autos	7	205	10	16	32.68	2.29
balance-scale	3	625	0	4	45.76	1.32
breast-cancer	2	286	10	0	70.28	0.88
breast-w	2	699	0	9	65.52	0.93
colic	2	368	16	7	63.04	0.95
credit-a	2	690	9	6	55.51	0.99
credit-g	2	1000	13	7	70.00	0.88
diabetes	2	768	0	8	65.10	0.93
glass	7	214	0	9	35.51	2.19
heart-c	5	303	7	6	54.46	1.01
heart-h	5	294	7	6	63.95	0.96
heart-statlog	2	270	0	13	55.56	0.99
hepatitis	2	155	13	6	79.35	0.74
ionosphere	2	351	0	34	64.10	0.94
iris	3	150	0	4	33.33	1.58
labor	2	57	8	8	64.91	0.94
lymph	4	148	15	3	54.73	1.24
primary-t.	22	339	17	0	24.78	3.68
segment	7	2310	0	19	14.29	2.81
sonar	2	208	0	60	53.37	1.00
soybean	19	683	35	0	13.47	3.84
vehicle	4	846	0	18	25.41	2.00
vote	2	435	16	0	61.38	0.96
vowel	11	990	3	10	9.09	3.46
zoo	7	101	16	2	40.59	2.41

Table 2: Accuracy (%) ± standard deviation for DecisionTable, J48, KernelDensity, KStar, MultiLinearRegression, and NaiveBayes. These six learners were used as base learners for all four meta-classification schemes.

Dataset	DecisionTable	J48	KernelDensity	KStar	MultiLinearRegression	NaiveBayes
audiology	75.31±2.00	76.95±1.05	77.35±0.58	80.00±0.62	79.07±0.98	72.21±0.65
autos	78.78±1.61	82.05±1.15	76.59±0.92	72.54±1.44	65.41±1.77	61.76±1.46
balance-scale	75.28±1.35	77.70±0.59	88.58±0.45	89.02±0.41	86.62±0.58	91.54±0.32
breast-cancer	71.92±1.72	73.88±1.41	72.62±0.87	74.27±0.60	71.15±1.01	73.18±0.66
breast-w	94.55±0.45	94.48±0.62	95.34±0.18	95.28±0.34	95.79±0.14	97.47±0.12
colic	82.36±1.30	85.43±0.43	79.32±0.61	75.92±0.43	81.88±0.94	78.75±0.36
credit-a	84.71±0.70	85.41±0.71	81.72±0.70	78.99±0.67	85.54±0.27	81.26±0.31
credit-g	71.83±1.01	71.18±1.40	69.83±0.54	70.09±0.67	75.77±0.43	74.58±0.41
diabetes	74.32±1.18	73.74±0.79	71.41±0.51	70.29±0.43	76.97±0.46	75.31±0.28
glass	70.33±1.06	67.71±2.14	70.05±0.97	75.42±1.50	56.59±2.02	50.79±1.17
heart-c	78.84±1.22	76.63±1.69	75.94±0.95	75.64±1.28	84.59±0.75	84.29±0.39
heart-h	79.76±0.90	79.66±1.18	78.71±0.77	77.72±0.74	86.39±0.42	84.97±0.39
heart-statlog	82.52±1.80	78.67±1.49	76.56±0.94	76.81±0.77	83.78±0.89	84.63±0.66
hepatitis	80.19±2.00	79.10±1.53	80.19±1.22	80.39±1.10	83.74±1.35	84.52±0.68
ionosphere	89.74±1.03	89.74±0.74	89.00±0.31	84.02±0.70	86.55±0.48	91.77±0.47
iris	92.80±0.93	94.67±0.70	95.20±0.53	94.67±0.00	84.27±0.78	95.93±0.38
labor	84.91±1.89	80.18±3.31	85.44±2.20	92.28±1.23	87.54±1.74	93.68±1.23
lymph	74.59±1.50	75.47±1.91	81.96±1.28	85.00±1.34	84.93±1.72	83.11±1.23
primary-t.	40.41±1.09	41.83±1.25	39.76±0.50	38.23±0.96	46.58±0.55	49.47±0.94
segment	92.20±0.55	96.90±0.28	97.29±0.12	97.11±0.16	83.23±0.09	85.81±0.15
sonar	71.73±2.02	73.32±1.90	85.87±0.56	84.71±1.13	72.45±1.09	71.97±1.11
soybean	86.56±0.96	92.47±0.54	91.11±0.21	87.88±0.34	93.69±0.13	92.90±0.21
vehicle	64.93±0.97	72.74±1.28	69.17±0.64	70.04±0.60	74.23±0.58	61.04±0.60
vote	94.53±0.69	96.46±0.27	92.60±0.24	93.31±0.23	95.63±0.00	90.16±0.18
vowel	72.09±1.12	80.17±1.04	99.11±0.15	98.77±0.28	42.92±0.68	70.58±1.15
zoo	89.70±0.96	92.28±0.42	96.04±0.00	96.04±0.00	92.57±1.88	95.05±0.00
Average	79.04	80.34	81.41	81.32	79.15	79.87

Table 2 shows the performance of the base algorithms on the datasets. All of them have their respective strengths and weaknesses and perform well on some datasets and badly on others. Judging by the average accuracy (a somewhat problematic measure, see below), KernelDensity and KStar seem to have the competitive edge.

3.3 Meta-Classification Schemes

On these base algorithms, we tested the following four meta-classification schemes:

- Grading is our implementation of the grading algorithms described in section 2. It uses the instance-based classifier IBk with ten nearest neighbors as the meta-level classifier. This choice is discussed in more detail in section 3.5. As described in the previous section, our implementation made use of the class probability distributions returned by the base classifiers. IBk estimates the class probabilities with a Laplace-estimate of the proportion of the neighbors in each class. These estimates are then normalized to yield a proper probability distribution.[5]

- X-Val chooses the best base classifier on each fold by an internal ten-fold cross-validation.

- Stacking is the stacking algorithm as implemented in WEKA, which follows (Ting & Witten, 1999). It constructs the meta dataset by adding the entire predicted class probability distribution instead of only the most likely class (cf. end of section 2) Following Ting and Witten (1999), we also used MultiLinearRegression as the level 1 learner.[6]

- Voting is a straight-forward adaptation of voting for distribution classifiers. Instead of giving its entire vote to the class it considers to be most likely, each classifier is allowed to split its vote according to its estimate of the class probability distribution for the example. Voting adds up the distributions of all six base learners, renormalizes the sum, and chooses the class with highest probability afterwards. It differs from the other three meta-classification schemes in that it does not make use of the class predictions computed by the internal cross-validation. It is mainly included as a benchmark of the performance that could be obtained without resorting to the expensive cross-validation of every other algorithm.

As noted above, all algorithms made use of the class probability distributions returned by the base classifiers (not only of the predictions themselves). This choice was partly motivated by the existing implementation of Stacking within WEKA, and partly because of the experimental evidence that shows that the use of class probability distributions gives a slightly better performance in combining classifiers with stacking (Ting

[5]The exact formula for the confidence is $p(+) = \frac{p+\frac{1}{n_e}}{k+\frac{2}{n_e}}$, where n_e is the size of the training set and p of k neighbors are being graded as $+$.

[6]With stacking, relatively global and smooth level-1 generalizers should perform well (Wolpert, 1992; Ting & Witten, 1999).

Table 3: Accuracy (%) $\pm$ standard deviation for the meta-classification schemes Grading, X-Val, Stacking, and Voting.

Dataset	Grading	X-Val	Stacking	Voting
audiology	83.36±0.84	77.61±1.80	76.02±1.68	84.56±0.85
autos	80.93±0.99	80.83±1.42	82.20±1.68	83.51±1.23
balance-scale	89.89±0.28	91.54±0.32	89.50±0.47	86.16±0.51
breast-cancer	73.99±0.79	71.64±1.65	72.06±1.17	74.86±0.65
breast-w	96.70±0.38	97.47±0.12	97.41±0.13	96.82±0.30
colic	84.38±0.60	84.48±0.86	84.78±0.74	85.08±0.52
credit-a	86.01±0.36	84.87±0.83	86.09±0.86	86.04±0.48
credit-g	75.64±0.53	75.48±0.32	76.17±0.60	75.23±0.55
diabetes	75.53±0.64	76.86±0.51	76.32±0.41	76.25±0.99
glass	74.35±1.24	74.44±1.74	76.45±1.06	75.70±1.76
heart-c	82.74±1.31	84.09±0.74	84.26±0.54	81.55±1.42
heart-h	83.64±0.47	85.78±0.48	85.14±0.88	83.16±0.65
heart-statlog	84.22±1.08	83.56±1.49	84.04±0.73	83.30±1.39
hepatitis	83.42±1.52	83.03±1.69	83.29±1.62	82.77±0.81
ionosphere	91.85±0.49	91.34±0.66	92.82±0.44	92.42±0.54
iris	95.13±0.45	95.20±0.76	94.93±1.14	94.93±0.84
labor	93.68±1.23	90.35±3.33	91.58±1.61	93.86±1.24
lymph	83.45±1.28	81.69±1.76	80.20±0.72	84.05±1.53
primary-t.	49.47±1.26	49.23±0.95	42.63±1.45	46.02±0.64
segment	98.03±0.15	97.05±0.22	98.08±0.11	98.14±0.19
sonar	85.05±0.77	85.05±0.95	85.58±1.01	84.23±1.17
soybean	93.91±0.33	93.69±0.13	92.90±0.60	93.84±0.33
vehicle	74.46±0.74	73.90±0.57	79.89±0.40	72.91±0.52
vote	95.93±0.22	95.95±0.36	96.32±0.42	95.33±0.29
vowel	98.74±0.33	99.06±0.14	99.00±0.10	98.80±0.36
zoo	96.44±0.69	95.05±1.04	93.96±1.09	97.23±0.78
Average	85.04	84.59	84.68	84.88

& Witten, 1999) and ensemble methods (Bauer & Kohavi, 1999). We did some preliminary studies which seemed to indicate that this is also the case for Grading, but we did not yet attempt a thorough empirical verification of this matter.

3.4 Comparing the Meta-Classification Schemes

Table 3 shows the accuracies for all meta-classification schemes and their standard deviations w.r.t. each dataset. Not surprisingly, no individual algorithm is a clear winner over all datasets; each algorithm wins on some datasets and loses on others. On average, Grading seems to be slightly more accurate than Stacking (85.04% vs. 84.68%).

Table 4: Significant wins/losses for the four meta-classification schemes against themselves and against all meta learning algorithms. In each column, the first number shows the significant wins for the algorithm in the column, and the second number for the algorithm in the row. $\sum$ meta and $\sum$ base denote wins/losses vs. all meta and all base learning schemes respectively.

	Grading	X-Val	Stacking	Voting
Grading	—	5/6	7/7	2/4
X-Val	6/5	—	6/3	9/7
Stacking	7/7	3/6	—	7/8
Voting	4/2	7/9	8/7	—
$\sum$ meta	17/14	15/21	21/17	18/19
DecisionTable	24/0	22/0	24/0	26/0
J48	20/2	17/3	18/1	20/1
KernelDensity	21/1	18/2	20/2	23/2
KStar	19/0	17/4	17/4	19/1
MultiLinearRegression	17/3	12/2	13/7	14/5
NaiveBayes	13/5	14/1	14/5	17/7
$\sum$ base	114/11	100/12	106/19	119/16

Somewhat surprising is the performance of Voting: although it does not use the expensive predictions obtained from the internal cross-validation, it seems to perform no worse than the other algorithms which do use this information. Of course, a comparison of algorithms with their average accuracy over a selection of datasets has to be interpreted very cautiously because of the different base-line accuracies and variances on the different problems. In the following, we take a closer look at the performance differences.

Table 4 shows significant wins/losses of the meta-classification schemes versus themselves and versus the base classifiers. Significant differences were calculated by a t-test with 99% significance level. Positive and negative differences between classifiers were counted as wins and losses respectively. Among the meta-classification schemes (upper part), Stacking and Grading are best with Grading slightly behind ($wins - losses = 4$ vs. 3) while X-Val and Voting lag far behind (more losses than wins). Although this evaluation shows that—contrary to the average performance discussed above—Voting does not get close to Grading and Stacking, it nevertheless outperforms X-Val. Hence, adding up the predicted class probabilities of different classifiers seems to be a better decision procedure than selecting the best algorithm by cross-validation, which is consistent with other results on ensembles of classifiers (Dietterich, 2000). The lower half of Table 4 shows that all meta-classification schemes are almost always an improvement over the six base learners. Measured in terms of the differences between wins and losses, Voting and Grading are both on first place ($wins - losses = 103$. Interestingly, Stacking's performance seems to be the worst: it has less wins than

Table 5: Performance of Grading as significantly positive (+) and negative (−) differences in accuracy vs. all base learners and the other meta-classification schemes. Differences were evaluated using a t-test with 99% significance niveau.

Dataset	Inst.	DecisionTable	J48	KernelDensity	KStar	MultiLinearRegression	NaiveBayes	X-Val	Stacking	Voting
labor	57	+	+	+		+		+	+	
zoo	101	+	+			+	+	+	+	
lymph	148	+	+						+	
iris	150	+			+	+	−			
hepatitis	155	+	+	+	+					
autos	205	+		+	+	+	+			−
sonar	208	+	+			+	+			
glass	214	+	+	+		+	+		−	
audiology	226	+	+	+	+	+	+	+	+	−
heart-statlog	270		+	+	+					
breast-cancer	286	+		+		+		+	+	
heart-h	294	+	+	+	+	−	−	−	−	
heart-c	303	+	+	+	+	−	−		−	
primary-t.	339	+	+	+	+	+			+	+
ionosphere	351	+	+	+	+	+			−	
colic	368	+	−	+	+	+	+			
vote	435	+	−	+	+	+	+			+
balance-scale	625	+	+	+	+	+	−	−		+
soybean	683	+	+	+	+		+		+	
credit-a	690	+		+	+	+	+	+		
breast-w	699	+	+	+	+	+	−	−	−	
diabetes	768		+	+	+	−		−	−	
vehicle	846	+	+	+	+		+		−	+
vowel	990	+	+	−		+	+	−		
credit-g	1000	+	+	+	+		+			
segment	2310	+	+	+	+	+	+	+		

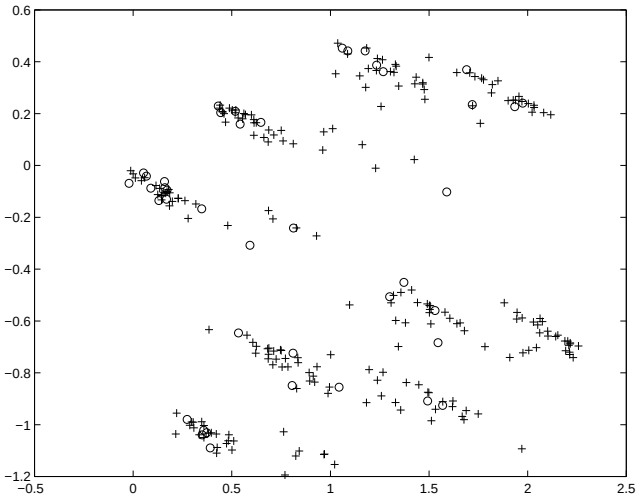

Figure 2: This is a projection of *heart-statlog* into two dimensions using principal components analysis. This projection accounts only for 44.8% of dataset variance so it may be rather skewed and distorted. Examples o are incorrectly classified and + are correctly classified by NaiveBayes during internal cross validation.

Voting and Grading, and the most losses of all meta-classification schemes. Clearly, meta-classification schemes are an improvement over base classifiers in any case.

A more detailed look on Grading is provided by Table 5, which shows its significant differences against all base and meta classifiers. As can be seen, Grading is not worse than any base classifiers on seventeen datasets and better than all base classifiers on two datasets, among them the largest (*segment*). Nine times, Grading is worse than at least one classifier, two times it is worse than two, and never worse than three or more. Our meta-classification scheme can thus be considered an improvement over the base learners in the majority of cases.

In head-to-head comparisons to the alternative meta-classification schemes, Grading is on par with Stacking (7/7), and has a slight advantage over X-Val ($+1$ *win*) and over Voting ($+2$ *wins*). It seems to be better than Stacking and X-Val on smaller datasets, while it seems to outperform Voting on larger datasets.

In summary, the performance of Grading is comparable to that of Stacking. Although Grading has a higher average performance over all datasets (Table 3) and has more significant wins against base classifiers (Table 4), its performance in a head-to-head comparison is equal to that of stacking (Table 5).

3.5 Evaluation of Level-1 Classifiers for Grading

Originally, our choice of IBk as a level 1 classifier was motivated by two reasons:

- For a base learner that performs well, we can expect that the error clusters are quite small. We thought that this problem can most easily be approximated by an instance-based learning algorithms which does not have to explicitly represent the complicated decision boundaries.

- We wanted to use a meta-learner that has a different bias than the base learners, so that it may uncover different types of regularities.

The first argument is illustrated in figure 2 which shows a 2D-projection of the correct and incorrect predictions of NaiveBayes on the *heart-statlog* dataset. We think that this problem might also be an explanation, why C5 was not able to produce comprehensible characterizations of model error in the related work of Bay and Pazzani (2000) (see also section 4).

In a second series of experiments, we wanted to test whether our intuitions were correct. Table 6 shows the results of the six base learners used as level 1 learners. For these experiments, we used only a one-time 10-fold cross-validation on all datasets (as opposed to the average of 10 cross-validations shown in the previous tables).

It turns out that IBk with ten nearest neighbors[7] is in fact one of the best level 1 learners among the seven tested. The only algorithm that appears to be slightly better is NaiveBayes.

However, all relative performances are within 1% of that of IBk, i.e., all algorithms perform about equal. This came as a little surprise to us, because we had expected that there would be more differences in this wide range of algorithms. Some of the algorithms learn local models (IBk), while others always consider all examples for deriving a prediction (KStar). Likewise, some of the algorithms always use all of the features (NaiveBayes), while others try to focus on important ones (J48). Apparently, the particular type of meta-learning problem encountered in grading, has some properties that make it uniformly hard for a wide variety of learning algorithms. We plan further investigations of this matter in the future.

4 Related Work

As already laid out in the introduction, grading is very similar to the approach introduced by Bay and Pazzani (2000). They also train a classifier to learn whether a classification is reliable or not. However, they did not use this approach for decision making, but instead aimed at providing insight about the domain regions in which a learner is not able to discriminate well. Among the two level-1 learners they analyzed, they found that C5 does not produce sufficiently understandable concept descriptions, while their own algorithm STUCCO was a little better. Grading, on the other hand, learns one such classifier for each base classifier, and uses their output for making a final prediction.

This idea of training a meta-classifier to utilize the predictions of multiple base classifiers is, of course, not new and can be found in several variations in the literature.

[7] After choosing IBk as the level 1 classifier (and prior to the evaluation described in this section), we had experimentally determined the number of neighbors ($k = 10$) using a one-time 10-fold cross-validation on all datasets.

Table 6: Grading with different level 1 classifiers. The first column shows the accuracy of IBk, all other columns show accuracy ratios ($\frac{Acc(\mathsf{Meta}_i)}{Acc(\mathsf{IBk})}$).

Dataset	IBk	DecisionTable	J48	KernelDensity	KStar	MultiLinearRegression	NaiveBayes
audiology	84.51	0.9895	0.9895	0.9791	1.0105	0.9895	1.0052
autos	81.95	1.0060	1.0060	0.9643	1.0000	0.9643	1.0060
balance-scale	89.76	1.0000	0.9982	1.0000	1.0018	1.0000	1.0000
breast-cancer	74.48	1.0000	0.9859	1.0047	1.0094	1.0094	1.0000
breast-w	96.57	1.0000	1.0015	0.9970	1.0000	0.9985	1.0015
colic	84.78	0.9936	1.0032	0.9968	1.0032	0.9840	0.9968
credit-a	86.09	0.9916	0.9933	0.9949	0.9966	0.9848	0.9916
credit-g	75.90	0.9934	0.9895	1.0000	0.9829	0.9789	0.9881
diabetes	76.30	1.0085	1.0051	1.0051	1.0085	1.0034	1.0017
glass	73.36	1.0191	1.0000	1.0255	1.0255	1.0382	1.0127
heart-c	85.48	0.9730	0.9691	0.9730	0.9807	0.9653	0.9846
heart-h	83.33	1.0122	1.0122	0.9755	0.9837	1.0041	1.0041
heart-statlog	83.70	0.9956	0.9823	0.9602	0.9735	0.9823	0.9867
hepatitis	81.94	1.0079	1.0236	1.0315	1.0157	1.0157	1.0472
ionosphere	91.17	1.0063	1.0063	1.0000	1.0000	1.0063	1.0094
iris	95.33	1.0000	1.0000	1.0070	1.0000	1.0070	1.0000
labor	94.74	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
lymph	84.46	1.0160	1.0240	0.9840	1.0000	1.0240	1.0080
primary-tumor	49.26	0.9641	0.9222	0.8922	0.9641	0.9641	1.0000
segment	98.10	1.0000	0.9987	0.9969	0.9951	0.9969	0.9965
sonar	86.06	0.9721	0.9441	0.9888	0.9777	0.9665	0.9832
soybean	94.00	0.9969	1.0000	1.0000	0.9751	0.9984	1.0031
vehicle	74.47	1.0254	1.0111	0.9889	0.9810	1.0333	1.0254
vote	96.09	0.9976	1.0000	0.9976	0.9976	0.9952	0.9952
vowel	98.79	0.9969	1.0010	0.9969	0.9980	0.9939	0.9980
zoo	97.03	1.0000	1.0000	1.0102	0.9898	0.9898	1.0102
Avg	85.29	0.9987	0.9949	0.9912	0.9950	0.9959	1.0021
±	10.81	0.0137	0.0217	0.0258	0.0143	0.0196	0.0130

The best-known technique is stacking (Wolpert, 1992), which is why we selected it as a benchmark for our algorithm. In our experiments, we closely followed the recommendations made by Ting and Witten (1999).

The main differences between grading and stacking (or combiners) are that the former does not change the independent variables—by replacing the original attributes with class predictions or class probabilities (or adding them to it)—but instead modifies the class values. Furthermore we create several sets of meta-data, one for each base classifier, and consequently learn several level-1 classifiers. Our predictions are currently combined via a form of voting using prediction confidence estimated from class distribution, but more elaborate schemes could be used here, e.g. by a second meta-classifier which combines the prediction confidences of the first meta-classifier, yielding a three level stacked architecture.

Chan and Stolfo (1995) propose the use of *arbiters* and *combiners*. A combiner is more or less identical to stacking. Chan and Stolfo (1995) also investigate a related form, which they call an *attribute-combiner*. In this architecture, the original attributes are not replaced with the class predictions, but instead they are added to them. As Schaffer (1994) shows in his paper about bi-level stacking, this may result in worse performance.

Cascading (Gama & Brazdil, 2000) is another related variant where the classifiers are applied in sequence and there is no level 1 classifier. Each base classifier, when applied to the data, adds his class probability distribution to the data and returns this augmented dataset, which are to be used by the next base classifier. Thus, the order in which the classifiers are executed becomes important. Cascading does not use an internal cross-validation like most other meta-classification schemes considered here, except voting, and is thus more efficient by an order of magnitude. In grading and stacking the classifier order is not important, thereby reducing the degrees of freedom for classifier selection and thus offering less chances for overfitting. Furthermore, cascading increases the dimensionality of the dataset with each step whereas grading leaves the dimensionality essentially unchanged.

The main differences between grading and stacking (or combiners, cascading) are that it does not change the independent variables—by replacing the original attributes with class predictions or class probabilities (or adding them to it)—but instead modifies the class values. Furthermore we create several sets of meta-data, one for each base classifier, and consequently learn several level-1 classifiers. Our predictions are currently combined via a form of voting using prediction confidence estimated from class distribution, but more elaborate schemes could be used here, e.g. by a second meta-classifier which combines the prediction confidences of the first meta-classifier, yielding a three level stacked architecture.

An arbiter (Chan & Stolfo, 1995) is a separate, single classifier, which is trained on a subset of the original data. This subset consists of examples on which the base classifiers disagree. They also investigate *arbiter trees*, in which arbiters that specialize in arbiting between pairs of classifiers are organized in a binary decision tree. Arbiters are quite similar in spirit to grading. The main difference is that arbiters use information about the disagreement of classifiers for selecting a training set, while grading uses disagreement with the target function (estimated by a cross-validation) to produce a new training set.

Quite related to grading is also the work by Ting (1997), who proposed to use the predictions of base classifiers for learning a function that maps the algorithms' internal confidence measure (e.g., instance typicality for nearest neighbor classifiers or a posteriori probabilities for Naive Bayes classifiers) to an estimate of its accuracy on the output. The main differences to grading is that we use the original feature vectors as inputs, and select the best prediction based on the class probabilities returned by those meta classifiers that predict that their corresponding base classifiers are correct on the example.

A third approach with a similar goal, *meta decision trees* (Todorovski & Džeroski, 2000), aims at directly predicting which classifier is best to classify an individual example. To this end, it uses information about statistical properties of the predicted class distribution as attributes and predicts the right algorithm from this information. The approach is not modular (in the sense that any algorithm could be used at the meta-level) but implemented as a modification to the decision tree learner C4.5. Grading differs from meta decision trees in that respect, and by the fact that we use the original attributes in the datasets for learning an ensemble of classifiers which learn the errors of each base classifier.

There are many approaches that combine multiple models without resorting to an elaborate meta-classification scheme. Best known are ensemble methods such as bagging and boosting, which rely on learning a set of diverse base classifiers (typically by using different subsamples of the training set), whose predictions are then combined by simple voting (Dietterich, 2000). Another group of techniques, *meta-learning*, focusses on predicting the right algorithm for a particular problem based on characteristics of the dataset (Brazdil, Gama, & Henery, 1994) or based on the performance of other, simpler learning algorithms (Pfahringer, Bensusan, & Giraud-Carrier, 2000). Finally, another common decision procedure (in particular with larger datasets) is to take a subsample of the entire dataset and try each algorithm on this sample. This approach was analyzed by Petrak (2000).

5 Conclusions

We have examined a new meta-classification scheme, grading. Essentially, the idea behind grading is to train a new classifier that predicts which base classifiers is correct on a given example. To that end, the original data set is transformed into a two-class dataset with new class labels that encode whether the the base classifier was able to correctly predict this example in an internal cross-validation or not. This approach may be viewed as a direct generalization of selection by cross-validation, which would always select the base classifier that corresponds to the meta dataset with the highest default accuracy.

The experimental evaluation showed that grading is slightly better than stacking according to some performance measures (like the average performance on a selection of UCI datasets or an indirect comparison to the base classifiers), but in a head-to-head comparison the differences are not statistically significant. Both algorithms, stacking and grading, perform better than voting and selection by cross-validation. Furthermore,

grading seems to be biased towards good performance on small datasets. This is a somewhat surprising result and should be studied in the future.

Our results also show that the method seems to be quite insensitive to the choice of a meta-learning algorithm. This is quite surprising, as we investigated a variety of algorithms with very different biases. We are yet unsure why this is the case. A possible reason may lie in the fact that for reasonable performances of the base learning algorithms, the two-class meta data set consists of far more correct than incorrectly predicted examples. Hence the learner should be able to deal with imbalanced training sets, which none of the algorithms we tested specializes in. We have not yet investigated the influence of this issue upon the performance of the learning system.

Some optimizations of the implementation such as weighted distances variable numbers of nearest neighbors, or (maybe most importantly) the use of more complex techniques for combining the meta classifiers have not yet been thoroughly examined.

We remain hopeful that our approach may in time become complementary to stacking. Our current work concentrates on the definition of a common framework for meta-classification schemes, which allows a thorough experimental and theoretical comparison of the different approaches that have been proposed in the literature.

Acknowledgements

This research is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF and the ESPRIT long-term project METAL (26.357). The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture. We want to thank Johann Petrak and Gerhard Widmer for valuable discussions and comments.

References

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*, 105–169.

Bay, S. D., & Pazzani, M. J. (2000). Characterizing model errors and differences. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*. Morgan Kaufmann.

Blake, C. L., & Merz, C. J. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`. Department of Information and Computer Science, University of California at Irvine, Irvine CA.

Brazdil, P. B., Gama, J., & Henery, B. (1994). Characterizing the applicability of classification algorithms using meta-level learning. In Bergadano, F., & Raedt, L. D. (Eds.), *Proceedings of the 7th European Conference on Machine Learning (ECML-94)*, pp. 83–102 Catania, Italy. Springer-Verlag.

Chan, P. K., & Stolfo, S. J. (1995). A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the 12th International Conference on Machine Learning (ICML-95)*, pp. 90–98. Morgan Kaufmann.

Cleary, J. G., & Trigg, L. E. (1995). K*: An instance-based learner using an entropic distance measure. In Prieditis, A., & Russell, S. (Eds.), *Proceedings of the 12th International Conference on Machine Learning*, pp. 108–114 Lake Tahoe, CA.

Dietterich, T. G. (2000). Ensemble methods in machine learning. In Kittler, J., & Roli, F. (Eds.), *First International Workshop on Multiple Classifier Systems*, pp. 1–15. Springer-Verlag.

Gama, J., & Brazdil, P. (2000). Cascade generalization. *Machine Learning*, *41*(3), 315–343.

Kononenko, I., & Bratko, I. (1991). Information-based evaluation criterion for classifier's performance. *Machine Learning*, *6*, 67–80.

Petrak, J. (2000). Fast subsampling performance estimates for classification algorithm selection. In Keller, J., & Giraud-Carrier, C. (Eds.), *Proceedings of the ECML-00 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pp. 3–14 Barcelona, Spain.

Pfahringer, B., Bensusan, H., & Giraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)* Stanford, CA.

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

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning*, *13*(1), 135–143.

Schaffer, C. (1994). Cross-validation, stacking and bi-level stacking: Meta-methods for classification learning. In Cheeseman, P., & Oldford, R. W. (Eds.), *Selecting Models from Data: Artificial Intelligence and Statistics IV*, pp. 51–59. Springer-Verlag.

Ting, K. M. (1997). Decision combination based on the characterisation of predictive accuracy. *Intelligent Data Analysis*, *1*, 181–206.

Ting, K. M., & Witten, I. H. (1999). Issues in stacked generalization. *Journal of Artificial Intelligence Research*, *10*, 271–289.

Todorovski, L., & Džeroski, S. (2000). Combining multiple models with meta decision trees. In Zighed, D. A., Komorowski, J., & ZŻytkow, J. (Eds.), *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD-2000)*, pp. 54–64 Lyon, France. Springer-Verlag.

Witten, I. H., & Frank, E. (2000). *Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–260.