Meta-Learning for Stacked Classification

Alexander K. Seewald ALEX@SEEWALD.AT; ALEXSEE@AI.UNIVIE.AC.AT
Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Wien, Austria

Abstract

In this paper we describe new experiments with the ensemble learning method Stacking. The central question in these experiments was whether meta-learning methods can be used to accurately predict various aspects of Stacking's behaviour. The resulting contributions of this paper are twofold: When learning to predict the accuracy of stacked classifiers, we found that the single most important feature is the accuracy of the best base classifier. A simple linear model involving just this feature turns out to be surprisingly accurate. The associated regression line has a gradient larger than one, hinting that, in the limit, Stacking is indeed better than the best included base classifier. When learning to predict significant differences between Stacking and three other ensemble learning methods, we have found simple models, all but one of which are based on single features which can be efficiently computed directly from the dataset. These models can be used to decide in advance which ensemble learning method to use on a given dataset, since neither of them is always the best choice.

1. Introduction

Meta-learning[1] 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). Here we are concerned with meta-learning of ensemble learning schemes or *meta-classification schemes*. Stacking can be considered the best-known such scheme and was introduced in (Wolpert, 1992). We take a more general view of meta-learning and use it to predict two aspects of Stacking's behaviour.

First, we investigated predicting the accuracy of stacked classifiers, to see which factors contribute to its performance. For this, we considered a variety of features, computed either directly from the dataset or from the classifiers which were part of the ensemble. Considering the unsatisfactory state of this field (Pfahringer, Bensusan & Giraud-Carrier, 2000; Bensusan & Kalousis, 2001), we were surprised to find that a single feature is able to predict the accuracy surprisingly well.

Afterwards we aimed to decide via meta-learning the question which meta-classification scheme to apply to the dataset at hand. Trying to predict the best scheme may fail because this does not take into account that more than one best scheme may exist – even all considered classifiers may be statistically indistinguishable on some datasets! Our approach takes this into account by constructing binary problems concerned with finding which one of two meta-classification schemes is significantly better than the other while excluding those examples where both are statistically indistinguishable – since in that case, both answers can be considered correct[2]. Since we focus on Stacking in this paper, we do not consider all possible pairs of two schemes but only those three involving Stacking.

1.1 Introducing Meta-Classification Schemes

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 (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. The best-known such *meta-classification scheme*

[1] The term *Meta-Learning* is also used elsewhere to refer to ensemble learning schemes – so in a way we achieve to relate both aspects of this term in our paper.

[2] Earlier experiments using an additional class for this case were unsuccessful – this may be explained by too much class overlap, see Figure 1.

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

Dataset	cl	Inst	disc	cont	bL	E
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

is the family of *stacking* algorithms (Wolpert, 1992). The idea behind stacking is to use the predictions of the original classifiers as attributes in a new training set that keeps the original class labels. Essentially, Stacking combines output from a set of base classifiers via one meta classifier.

A straightforward extension of this approach is using class probability distributions of base classifiers[3] which convey not only prediction information, but also confidence for all classes. This was found to be superior in (Ting & Witten, 1999) when used with *multi-response linear regression* (MLR) as meta classifier. We used this extension in our experiments.

We will now describe our experimental setup and our two sets of features describing dataset characteristics and base classifier accuracy & diversity. Then we will give results for predicting the accuracy of stacked classifiers, followed by meta-learning of significant differences. Afterwards we give a short overview on related meta-learning research and conclude this paper.

2. Experimental setup

In our experiments, we used twenty-six datasets from the UCI machine learning repository (Blake & Merz, 1998). Details can be found in Table 1. We used Stacking with all of the following seven base classifiers for our experiments, which were chosen in an attempt to maximize diversity. All algorithms were taken from the Waikato Environment for Knowledge Analysis (WEKA[4]), Version 3-1-8.

- DecisionTable: a decision table learner.
- IBk: the IBk instance-based learner using K=1 nearest neighbors. K=1 was chosen to offset the K* algorithm with a maximally local learner.
- J48: a Java port of C4.5 Release 8 (Quinlan, 1993)
- KernelDensity: a simple kernel density classifier.
- KStar: the K* instance-based learner (Cleary & Trigg, 1995), using all nearest neighbors and an entropy-based distance function.
- MLR: a multi-class learner based on linear regression, which tries to separate each class from all other classes by linear discrimination (*Multi-response Linear Regression*)
- NaiveBayes: the Naive Bayes classifier using multiple kernel density estimation for numeric attributes.

We used the following four meta-classification schemes.

- 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. We used MLR as the level 1 learner.[5]
- X-Val chooses the best base classifier on each fold by an internal ten-fold CV. This is just the selection by cross-validation we mentioned in the beginning.
- 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 the base classifier's estimate of the class probability distribution for the example. I.e. the mean class distribution of all classifiers is calculated. It is the only scheme which does not use an expensive internal cross-validation.
- Grading is an implementation of the grading algorithm evaluated in (Seewald & Fürnkranz, 2001) which uses IBk ($K = 10$) as meta-classifier. Basically, Grading trains a meta classifier for each base classifier which

[3]Every prediction is replaced by a vector of probabilities, one for each class.

[4]The Java source code of WEKA has been made available at `www.cs.waikato.ac.nz`.

[5]Relatively global and smooth level-1 (=meta) generalizers should perform well (Wolpert, 1992; Ting & Witten, 1999).

tries to predict when its base classifier fails. This decision is based on the dataset's attributes. A weighted voting of the base classifiers' prediction gives the final class prediction. The voting weight is the confidence for a correct prediction of a base classifier, which is estimated by its associated meta classifier.

We used seventeen dataset-related features which uniquely characterize the dataset. These were inspired by the StatLOG project (Brazdil, Gama & Henery, 1994) and reimplemented in WEKA.

- *Inst*, the number of examples.
- *log(Inst)* which is the natural logarithm of *Inst*.
- *Classes*, the number of classes.
- *Attrs*, the number of attributes (excluding the class)
- *PropNomAttrs*, number of nominal attributes as a proportion of *NumAttrs*.
- *PropContAttrs*, number of numeric attributes as a proportion of *NumAttrs*.
- *PropBinAttrs*, number of binary-valued nominal attributes as a proportion of *NumAttrs*.
- *ClassEntropy*, the entropy of the class attribute.
- *AttrEntropy*, the entropy of all attributes.
- *MutualEntropy*, the mutual entropy of class and attributes.
- *EquivAttrs*, the equivalent number of attributes, $\frac{ClassEntropy}{MutualEntropy}$
- *RelEquivAttrs*, $\frac{EquivAttrs}{Attrs}$
- *S/N*, the signal-to-noise ratio.
- *MeanAbsCorr*, the mean absolute pairwise correlation between all different pairs of numeric attributes.
- *MeanAbsSkew*, the mean absolute skew of all numeric attributes.
- *MeanAbsKurtosis*, the mean absolute kurtosis of all numeric attributes.
- *defAcc*, the default accuracy, i.e. the proportion of the most common class.

Additionally, we used the accuracies of our seven base-learners as features. We also calculated standard statistical features of this set of seven accuracies. Furthermore, we used the same statistical features over pairwise base classifier κ-statistics[6], a measure of diversity due to (Dietterich, 2000)

- Seven accuracy values, one for each base classifier (*DT, IBk-K1, J48, KD, KStar, MLR, NB-K*)
- Eight statistical features describing the set of accuracy values (*MinAcc, MaxAcc, MeanAcc, StDevAcc, SkewAcc, SkewAcc*^{2}*, KurtosisAcc, relRangeAcc* $= \frac{MaxAcc-MinAcc}{StDevAcc}$)
- Eight statistical features describing the set of all pairwise κ-statistics between base classifiers (*MinK, MaxK, MeanK, StDevK, SkewK, SkewK*^{2}*, KurtosisK, relRangeK*)
- *relMeanAcc*=$\frac{AvgAcc}{defAcc}$, the ratio of average accuracy to default accuracy.

The above features were computed both on the full data set and also on predictions estimated via tenfold crossvalidation. For meta-learning of significant differences, we only used the latter set because it consistently offered better estimates during the first task. All statistical differences for meta-learning were computed via a t-Test with α=99%, based on the accuracies generated by ten-times ten-fold crossvalidation.

3. Estimating Stacking's Accuracy

This section is concerned with predicting the accuracy of stacked classifiers. Since we prefer simple models by Ockham's Razor, we first investigated the simplest models possible: based on only a single feature. Thus, we assumed linear relationships between each feature and the accuracy of our stacked classifier and characterized this relation by statistical correlation coefficients and mean absolute errors (MAE). Afterwards, we considered more complex and non-linear models obtained by various regression algorithms from machine learning.

3.1 Linear Models based on Single Features

As a first step, we calculated statistical correlation coefficients and mean absolute errors (MAE) for all our features, always versus the accuracy of the stacked classifiers. The dataset-related features can be found in Table 2, and the base-classifier-related features in Table 3. Correlations and MAEs were determined for all meta-data (**All**) and also via leave-one-out crossvalidation (**CV**). In the former, this estimate was based on the output of one linear regression model computed from all meta-examples[7] In the latter case,

[6] A value of 1.0 stands for identical predictions between two learners while a value of 0.0 represents random correlations. A negative value signifies systematic disagreement between classifiers.

[7] Each meta-example consist of features computed from one original dataset, either directly or indirectly, followed by the ac-

Table 2. This table shows the correlations and mean absolute errors (MAE) for dataset-related features vs. the accuracy. The first column shows the correlation for the full meta-dataset (26 examples), the second column shows the correlation estimated by leave-one-out crossvalidation. Best features are shown in **bold**.

Dataset	**All**		**CV**	
Features	Corr	MAE	Corr	MAE
$Inst$	0.26	0.084	0.03	0.090
$logInst$	0.11	0.087	-0.40	0.095
$Classes$	0.37	0.089	-0.09	0.100
$Attrs$	0.07	0.087	-0.58	0.094
$PropNomAttr$	0.29	0.087	-0.04	0.095
$PropContAttr$	0.29	0.087	-0.04	0.095
$PropBinAttr$	0.20	0.088	-0.30	0.098
$ClassEntropy$	0.16	0.089	-0.54	0.100
$AttrEntropy$	0.26	0.085	-0.05	0.094
$MutualEntropy$	0.49	0.072	**0.42**	**0.077**
$EquivAttrs$	**0.58**	**0.068**	0.40	0.079
$RelEquivAttrs$	0.43	0.075	0.26	0.082
S/N	0.23	0.083	0.00	0.088
$MeanAbsCorr$	0.08	0.087	-0.47	0.093
$MeanAbsSkew$	0.06	0.087	-0.45	0.093
$MeanAbsKurtosis$	0.06	0.088	-0.37	0.095
$defAcc$	0.01	0.087	-0.94	0.095

this estimate was based on the output of twenty-six linear models which were determined using all but one meta-example and evaluated on the remaining meta-example. The latter case is a more reliable indicator of model performance on unseen data, however for meaningful interpretation the possibly diverse models have first to be integrated into a coherent whole.

As can be expected from a high-bias linear model, all base-classifier related features show a graceful degradation from **All** to **CV**. We were surprised to note that this is not always true for the dataset-related features - about half of the features have a negative correlation for **CV** whose absolute value is higher than the positive correlation for **All**. This higher negative correlation can unfortunately not be used to predict stacked accuracy[8] and is always coupled to a large MAE. It seems that a lot of the dataset-related features are not relevant to this task or that a one-dimensional linear model is not appropriate to find a meaningful relation. In any case if we disregard negative correlations, the highest correlation is always coupled with the lowest MAE as we would expect it to be.

In the case of base-classifier related features, we have an additional dimension: we can estimate the base classifier accuracies on the full dataset (**AllT, CV T**) or via tenfold crossvalidation (**All, CV**). Since Stacking uses CV internally, we expect **All** and **CV** to be better predictors for stacked accuracy. This is indeed the case – a single feature, *MaxAcc*, already yields excellent results. Even though the internal CV of Stacking is based on less data than the CV used to obtain *MaxAcc*, it is still quite competitive to the best classifier by hindsight!

However, computing a crossvalidation on the original dataset comes with a non-negligible computational cost. On the other hand, to compute the features for **All** and **CV** we generate data which could be used as meta-dataset for Stacking so nothing is lost. What we get in this case is that we do not have to compute the outer cross-validation and we do not need to run the meta classifier ten times to obtain a reasonable estimate of stacked classifier accuracy. So this is still about tenfold more efficient that running Stacking directly. An additional computational cost reduction by the same factor could be obtained by using training set accuracy – which motivates the **AllT** and **CV T** set. As expected, in this case we get less good results for best single feature, *MeanAcc*.

3.2 Combined features

In order to test how we may improve our results by using multiple features, we resorted to using standard machine-learning approaches for regression on our meta-dataset. We created one meta-dataset with accuracy estimation via training set (*MetaTrain*) and one estimated via tenfold CV (*MetaCV*). The dataset-related features were included in both cases.

We initially decided to evaluate linear regression, LWR (*locally weighted regression*), model trees[9], regression trees, KStar and IBk instance based learners at the meta-level, but linear regression and model trees proved superior[10], so we only used these two learners for further experiments.

So far, the best single features were *MaxAcc* for *MetaCV* (Train: c=0.96, MAE=0.021; CV: c=0.96, MAE=0.022) and *MeanAcc* for *MetaTrain* (Train: c=0.84, MAE=0.045; CV: c=0.81, MAE=0.049). We will now turn towards models from multiple features to see if we can improve on these simple models.

For *MetaCV*, the best model was a model tree. Both the training set model and all but one crossvalidated model[11] were linear functions of a single feature, *MaxAcc*. Consequently, the performance on the training set is identical to the single-feature case (c=0.96, MAE=0.021). The leave-one-out CV is very slightly worse with c=0.95, MAE=0.023. In this case, combined features are not able to yield better performance than the best single feature. The best model for the accuracy of the stacked classifier is thus:

curacy of the stacked classifier.

[8]The maximum negative correlation appears in feature *defAcc* (-0.94; **CV**) This correlation is based on twenty-six different models, one per leave-one-out training fold. All data would have to be used to determine the final regression line, but then this result can no longer be validated and seems certainly too optimistic. This can also be seen by the quite high MAE.

[9]M5Prime from WEKA, see (Wang & Witten, 1997)

[10]Both were always top two by highest correlation and lowest MAE with the rest of the field – usually far – behind.

[11]generated via leave-one-out CV on the meta-data

Table 3. This table shows the correlation and mean absolute errors (MAE) on base-classifier related features vs. the accuracy. For the first two columns, all features are based on training set performance of the base classifiers. For the last two columns, a ten-fold crossvalidation was used to determine better but more costly estimates of base classifier performance. The first and third column show the correlations and MAE on the full meta-dataset, the second and fourth column show correlations and MAE estimated via leave-one-out crossvalidation.

Classifier	**AllT**		**CV T**		**All**		**CV**	
Features	Corr	MAE	Corr	MAE	Corr	MAE	Corr	MAE
DT	0.77	0.055	0.67	0.062	0.83	0.046	0.79	0.051
$IBk-K1$	0.72	0.072	0.69	0.076	0.95	0.030	0.94	0.032
$J48$	0.79	0.055	0.73	0.061	0.84	0.042	0.81	0.046
KD	0.69	0.076	0.53	0.086	0.95	0.029	0.94	0.031
$KStar$	0.59	0.084	-0.21	0.102	0.93	0.037	0.92	0.040
MLR	0.50	0.072	0.07	0.084	0.32	0.083	-0.11	0.097
$NB-K$	0.81	0.050	0.73	0.056	0.77	0.052	0.68	0.059
$MinAcc$	0.58	0.069	0.27	0.080	0.40	0.078	0.00	0.091
$MaxAcc$	0.71	0.072	0.68	0.088	**0.96**	**0.021**	**0.96**	**0.022**
$MeanAcc$	**0.84**	**0.045**	**0.81**	**0.049**	0.92	0.033	0.91	0.036
$StDevAcc$	0.58	0.065	0.36	0.074	0.26	0.083	0.05	0.088
$SkewAcc$	0.46	0.073	0.31	0.079	0.32	0.083	0.02	0.091
$SkewAcc^2$	0.35	0.081	0.16	0.087	0.20	0.083	-0.10	0.089
$KurtAcc$	0.39	0.077	0.22	0.083	0.29	0.083	0.12	0.088
$relRangeAcc$	0.40	0.075	0.26	0.080	0.20	0.090	-0.20	0.097
$MinK$	0.56	0.065	0.46	0.069	0.40	0.075	0.23	0.081
$MaxK$	0.00	0.087	-0.19	0.096	0.18	0.085	-0.14	0.091
$MeanK$	0.70	0.059	0.61	0.064	0.64	0.060	0.57	0.065
$StDevK$	0.48	0.063	0.36	0.068	0.15	0.086	-0.20	0.100
$SkewK$	0.58	0.071	0.43	0.078	0.66	0.064	0.58	0.070
$SkewK^2$	0.61	0.082	0.32	0.094	0.57	0.075	0.42	0.083
$KurtK$	0.52	0.088	0.09	0.100	0.59	0.070	0.47	0.077
$relRangeK$	0.08	0.090	-0.81	0.099	0.35	0.076	0.21	0.081
$relMeanAcc$	0.28	0.082	0.08	0.087	0.38	0.078	0.27	0.082

$$StAcc = 1.074 * MaxAcc - 0.082$$

For *MetaTrain*, the best model was generated by linear regression. While the performance on the training set is better (c=0.96, MAE=0.023), the leave-one-out CV is worse in both measures than the best single feature (c=0.74, MAE=0.057). This is most probably due to overfitting. Thus we have also found no better model than the best single feature.

However, from preliminary experiments using a smaller feature set we had already seen better results for *MetaTrain*. Therefore we have decided to run a wrapper feature subset selection for both linear regression and model trees and also for *MetaCV* and *MetaTrain* separately – four wrapper subset selections in all.

For feature subset selection from *MetaCV*, the linear model was best, but model trees were only slightly behind. While the training set performance is clearly better (c=0.98, MAE=0.016), the leave-one-out CV shows only a small reduction in MAE from 0.022 to 0.021; the correlation stays the same. This small reduction is MAE is probably not significant, so we still conclude that for *MetaCV*, we have found no model which is significantly better than the best single feature, *MaxAcc*.

For feature subset selection from *MetaTrain*, the linear model was also best, this time by a larger margin. Both the training set performance (c=0.97, MAE = 0.019) and the leave-one-out CV performance (c=0.94, MAE=0.03) are clearly better than the best single feature, *MeanAcc*. Given that, as expected, the correlation is lower and MAE higher for leave-one-out CV – an indication against overfitting – we will now analyze both the model on all data and twenty-six crossvalidation models further to see if we are able to integrate them into a single, coherent model of stacked classifier accuracy.

In each leave-one-out fold, the linear model used all nine selected features. So we determined mean and standard deviation of the multiplicative weights of all features which can be found in Table 4. As can be seen, the weights of the model from all data (**Weight by train**) are all easily within the confidence intervals of the average weights from the twenty-six crossvalidated models – another indication that the model is reasonable.

As expected, the best single feature *MeanAcc* has the highest weight. *MLR* and *DT*, two base classifier features, also obtain high negative weights. The default accuracy *defAcc* seems to be less relevant. Generally, features derived

Table 4. This table shows the feature weights of the best model on *MetaTrain*, both of the training set model and the average weights of the twenty-six cross-validated models. *Const* specifies the constant term of the linear model.

Feature	Weight by CV	Weight by train
$MeanAcc$	3.9850 ± 0.1766	3.9700
MLR	-0.9122 ± 0.0503	-0.9080
DT	-0.6955 ± 0.0685	-0.6940
$defAcc$	0.1186 ± 0.0114	0.1180
$relRangeAcc$	0.0804 ± 0.0067	0.0796
$PropNomAttr$	0.0472 ± 0.0050	0.0473
$relRangeK$	0.0039 ± 0.0054	0.0042
$EquivAttrs$	-0.0027 ± 0.0002	-0.0027
$Inst$	0.0001 ± 0.0000	0.0001
$Const$	-1.7312 ± 0.0728	-1.7300

from classifiers seem to be more relevant in the context of predicting accuracy than those derived directly from the datasets, which was also found in (Bensusan & Kalousis, 2001). Space restrictions prevent us from showing the regression formula, but it can be easily constructed via Table 4.

4. Meta-Learning of Significant Differences

This section is concerned with predicting significant differences between Stacking and three other meta-classification schemes. For each of Stacking vs. Voting, Stacking vs. Grading and Stacking vs. X-Val, we generated a separate meta-dataset consisting of all dataset-dependent and classifier-dependent features[12] followed by a binary class variable, being 1 if Stacking is significantly better than the other scheme and 0 otherwise. In case there is no significant difference, we removed the respective example from the meta-dataset, under the premise that in this case we can consider both variants to be equivalent and thus judge either answer to be correct.

On these meta-datasets, we evaluated a number of standard machine learning algorithms available in WEKA[13] via leave-one-out crossvalidation. We only discuss the best models which in most cases seem to be rather simple and based on single attributes only, hinting that they may be robust.

4.1 Stacking vs. Voting

For Stacking vs. Voting, there are twelve datasets without significant differences. After removing them from our meta-dataset, we have fourteen instances, seven with class=1, seven with class=0. The baseline accuracy is thus 50%. Here, IBk is the best meta-learner with an accuracy of 92.86% and a single error for *vote*. A cross-validation using only seven folds shows the exact same result.

When removing the base-classifier dependent features, IBk is still the best classifier with an additional error on *labor*, the smallest dataset. In this case MLR which is also a global learner is equally good. So we may tentatively conclude that for this meta-dataset, there seems to be no single feature which can predict the significant differences as good as a combination of all features. Figure 1 shows a projection of our meta-dataset into two dimensions. The single $*$ in the $\circ$ territory corresponds to dataset *vote*.

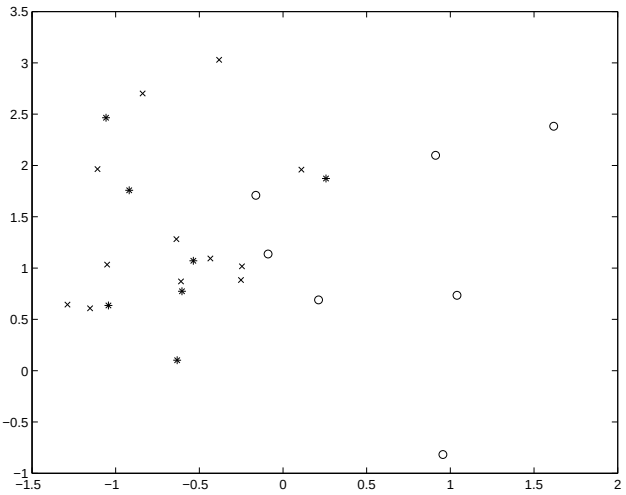

Figure 1. This shows a projection of the Stacking vs. Voting meta-dataset onto two dimensions, via principal components analysis (53.1% of variance explained). $*$ stands for class=1 (where Stacking is better) and $\circ$ for class=0. Those datasets which were no part of the meta-dataset (no significant differences between Stacking and Voting) are also shown, as $\times$.

4.2 Stacking vs. Grading

For Stacking vs. Grading, there are again twelve datasets on which there are no significant differences. After removing them from our meta-dataset, we have fourteen instances whose classes are again equally distributed. Thus the baseline accuracy is also 50%. Here, J48 is the best choice with 92.86% accuracy and only a single error on the smallest dataset, *labor*. The training set model is based on a single attribute, *propNomAttr*. In all fourteen folds but two there is the same model, which also appears as the training set model.

$$\begin{array}{lcl} propNomAttr <= 0.684211 & : & class = 1 \\ \text{otherwise} & : & class = 0 \end{array}$$

In the two other folds, the same attribute appears in the same formula with 0.65 and 0.695652 resp. as value on the right side. It seems that the proportion of nominal attributes plays a role on the performance between Stacking and Grading: in case there are less than about $\frac{2}{3}$ of nominal

[12] Because of the much better results in predicting stacked classifier accuracy, we only considered those classifier features estimated via cross-validation.

[13] All base learners plus 1R and DecisionStump

attributes, Stacking works significantly better than Grading.

4.3 Stacking vs. XVal

For Stacking vs. X-Val, seventeen examples offer no significant differences. Only nine examples remain for our experiments, the baseline accuracy is already 66.7%. Interestingly in this case the best model is from DecisionStump which learns a single J48 node, obtaining 88.9% accuracy, corresponding to a single error on dataset *balance-scale*. It seems J48 is prone to overfitting on this meta-dataset. The training set model is based on *meanAbsSkew*. The models from the nine folds are more diverse: seven times, the following model appears:

$$\begin{array}{lrcl} meanAbsSkew & <= & 0.31 & : \; class = 0 \\ meanAbsSkew & > & 0.31 & : \; class = 1 \\ meanAbsSkew & & missing & : \; class = 0 \end{array}$$

Once the same model appears with value 0.53 instead of 0.31. Once a model based on $numClasses <= 13 : class = 1$ appears. The same overall accuracy is also obtained in a six-fold cross-validation. Still, this is probably those of our models which is least trustworthy.

5. Related Research

Up to now there is no research aiming to either predict the accuracy of meta-classification schemes or to predict which meta-classification scheme to use for a given dataset. In this paper we have investigated both tasks and found them to work quite well. Other Research is usually concerned with simple, at most boosted or bagged, classifiers. We give some representative examples.

Using regression to predict the performance of basic learning algorithms was first investigated in (Gama & Brazdil, 1995), continuing the framework of StatLOG(Brazdil, Gama & Henery, 1994). They report poor results in terms of normalized mean squared error.

(Bensusan & Kalousis, 2001) have recently investigated meta-learning in a similar setting, using features extended from STATLOG, histograms of numeric attributes and landmarking using seven learners, four of which are also used by us as base classifiers. They found that mean absolute errors are usually significantly lower than a default strategy of predicting test dataset error as the mean error of training dataset error. However, a ranking based on these models performs only once significantly better than the default ranking based on average accuracy over all datasets. Only mean absolute errors are given for the regression experiments. Some regression rules from Cubist are shown and interpreted.

(Brazdil, Gama & Henery, 1994) have investigated meta-level learning to predict the best classifier for a given dataset. They use an ad-hoc specified confidence interval around the best accuracy to define *applicable* [14] and *inapplicable* classifiers for each dataset. Our approach uses the statistical t-Test which seems to be more appropriate. While their approach has to integrate possibly conflicting rules concerning applicability, making the evaluation quite complex, our approach can predict significant differences directly. Furthermore, the focus on using only decision trees and derived rules may have lead to suboptimal results as we had to use a variety of machine learning techniques to get optimal results. They also considered only one-against-all comparisons between candidate classifiers instead of the pairwise comparisons best-against-others we investigated.

(Pfahringer, Bensusan & Giraud-Carrier, 2000) investigated meta-learning using landmarkers, which are fast and simple learning algorithms used to characterize the dataset. The features which we derived from base classifiers can be considered similar. They report improved results by landmarking which we also observed for predicting the accuracy of stacked classifiers. However, for predicting significant differences we found dataset-related features to be more appropriate. They report work on removing ties from the meta-dataset and show that in some cases this can hurt meta-learning performance.

[14] within confidence interval $\equiv$ *applicable*

6. Conclusion

In this paper we have investigated the use of machine learning techniques in the context of meta-learning both to predict stacked classifier accuracy and significant differences between Stacking and three other meta-classification schemes. We believe these tasks to be complementary, since predicting the accuracy of meta-classification schemes may potentially also be used to determine significant differences.

We used both dataset-related and base-classifier related features in our tasks. In the context of predicting classifier accuracy, we found that classifier-related features, mostly those derived from accuracy, are better suited to this task, as have others (Bensusan & Kalousis, 2001; Pfahringer, Bensusan & Giraud-Carrier, 2000). A single feature, the accuracy of the best component classifier in the ensemble, is able to predict the accuracy of the stacked classifier quite well. Details can be found in Section 3.2.

However, we have found that for determining significant differences between schemes, features derived directly from the dataset may be better suited – in two of our three meta-learning problems concerned with predicting significant differences a model based on a single dataset-related feature was superior. In the third case an instance-based learner using all features was best, but using just dataset-related features lead to only one additional error, the same learner remained the best one. Additionally, our meta-learning experiments were constructed to predict significant differences only where they appear. While the removal of *ties* from the meta-dataset was previously mentioned in work by (Brazdil, Gama & Henery, 1994), using a less ad-

hoc and more appropriate statistical test to determine those ties seems to have been overlooked previously. Details can be found in Section 4.

At last we have found that there is no single best meta-classifier for predicting significant differences – a variety of machine learning algorithms had to be evaluated for optimal results. This hints that pairwise learning problems have different properties, which may explain why meta-learning is usually so hard. In contrast, meta-learning for meta-classification schemes seems to be a much easier problem.

Acknowledgements

This research is supported by the Austrian *Fonds zur Förderung der Wissenschaftlichen Forschung (FWF)* under grant no. P12645-INF. This research was also partially supported by the ESPRIT LTR project METAL (26.357). The Austrian Research Institute for Artifi cial Intelligence is supported by the Austrian Federal Ministry of Education, Science and Culture. We would like to thank Johannes Fürnkranz and Gerhard Widmer for valuable comments on a draft version of this paper.

References

Bensusan, H., Kalouis, A.: Estimating the Predictive Accuracy of a Classifier. In Proceedings of the twelveth European Conference on Machine Learning, Freiburg, Germany. Springer Verlag.

Blake, C. L., Merz, C. J: UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html` (1998). 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. *Proceedings of the 7th European Conference on Machine Learning (ECML-94)* (83–102). Catania, Italy: Springer-Verlag.

Cleary, J. G., Trigg, L. E: K*: An instance-based learner using an entropic distance measure. Proc. 12th International Conference on Machine Learning (1995) 108–114, Lake Tahoe, CA.

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

Gama, J., Brazdil, P: Characterization of classification algorithms. In Proceedings of the 7th Portugese Conference in AI, EPIA 95, 83–102, 1995.

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

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

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

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

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

Seewald A.K., Fürnkranz J.: An Evaluation of Grading Classifiers, in Hoffmann F. et al. (eds.), Advances in Intelligent Data Analysis, Proc. 4th International Conference, IDA 2001, Springer, 115–124. Also available as Technical Report (extended version), Austrian Research Institute for Artificial Intelligence, Vienna, TR-2001-01, 2001.

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

Wang, Y., Witten, I. H: Induction of model trees for predicting continuous classes. Proc. of poster papers of European Conference on Machine Learning, University of Economics, Faculty of Informatics and Statistics, Prague.

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