Hybrid Algorithms with Instance-Based Classification

Iris Hendrickx and Antal van den Bosch

ILK / Computational Linguistics and AI, Tilburg University,
P.O.Box 90153, NL-5000 LE Tilburg, The Netherlands
{i.h.e.hendrickx, antalb}@uvt.nl

Abstract. In this paper we aim to show that instance-based classification can replace the classifier component of a rule learner and of maximum-entropy modeling, thereby improving the generalization accuracy of both algorithms. We describe hybrid algorithms that combine rule learning models and maximum-entropy modeling with instance-based classification. Experimental results show that both hybrids are able to outperform the parent algorithm. We analyze and compare the overlap in errors and the statistical bias and variance of the hybrids, their parent algorithms, and a plain instance-based learner. We observe that the successful hybrid algorithms have a lower statistical bias component in the error than their parent algorithms; the fewer errors they make are also less systematic.

1 Introduction

A distinguishing characteristic of instance-based learning [1, 2] is that it is non-abstracting local learning method. It does not abstract from the training instances to form a model, but stores them as such in memory. All effort is diverted to the classification phase. To classify a new instance the instance-based learning algorithm searches through memory to find the most similar instances in the local neighborhood of the new instance, and assigns the majority class label of the neighborhood.

Instance-based learning is also referred to as *lazy* learning as opposed to *eager* learning. Eager learning algorithms put significant effort in abstracting from the training instances by creating condensed representations (decision trees, rule sets, probability matrices, hyperplanes, etc.) during the learning phase. The classification phase of an eager learner reduces to a relatively effortless application of the abstracted representation to new instances.

This contrast between instance-based learning (which puts effort in classification) and eager learning (which invests its effort in the learning phase) forms the motivation for constructing the hybrids described in this paper. Earlier work has shown that combining lazy and eager learning techniques can be beneficial to generalization performance [3, 5]. In this paper we describe hybrid algorithms in which we combine effort-intensive eager learning in rule learning and maximum-entropy models with effort-intensive instance-based classification. We take the

J. Gama et al. (Eds.): ECML 2005, LNAI 3720, pp. 158–169, 2005.
© Springer-Verlag Berlin Heidelberg 2005

system as constructed by the eager learner and replace its standard classification component by instance-based classification through the k-nearest neighbor (k-NN) classifier. From the eager learner perspective we hope that replacing their simple classification method with the more sensitive local k-NN classification method could improve generalization performance. Put alternatively, we take the eager learner's model and transplant it into the distance metric of the instance-based learner. The hybrid algorithms use the model as produced by the eager learner to modify the distance calculations central in k-NN.

We construct three hybrid algorithms. The first combines maximum-entropy modeling with k-NN, the second and third hybrids both combine rule learning with k-NN, in two different ways. We investigate the performance of the hybrid algorithms and compare them to the performance of their parent algorithms. We also analyze to which extent the hybrid deviates functionally from its two parent algorithms. To get a deeper insight in the differences and commonalities of the parent algorithms and the hybrids, we analyze their overlap in errors and the statistical bias and variance.

In Section 2 we discuss the different learning algorithms and the construction of the hybrid algorithms. Section 3 and 4 provide a description of the experimental setup and the results, respectively. Section 5 describes the error analysis and bias-variance analysis. We discuss our findings in Section 6.

2 Algorithms

We first describe the three machine learning algorithms involved in this study briefly: instance-based learning, maximum-entropy modeling and rule learning. With instance-based learning we focus on two aspects: the MVDM distance metric and feature weighting, because these play a role in the hybrid algorithms. In the next two subsections we describe the construction of each of the hybrid algorithms and our motivations.

The k-nearest neighbor classification rule [1] is the classifier engine of the instance-based learning algorithm. The rule classifies new instances by searching for the k nearest neighbors to the new instance and extrapolating the majority class label found among the k nearest neighbors to the new instance. The distance between instances can be estimated with different distance metrics. A simple metric for nominal features is the overlap metric (or Manhattan distance, or L1-norm distance) which counts the number of mismatching feature values between two instances. A more sophisticated metric that estimates real-valued distances between pairs of nominal values is the Modified Value Difference Metric (MVDM) introduced in [6].

MVDM estimates, from training data, the distance between two symbolic feature values v_1 and v_2 as a vector distance between their two class distributions:

$$\delta(v_1, v_2) = \sum_{i=1}^{j} |P(C_i|v_1) - P(C_i|v_2)| \quad (1)$$

where the vector length is determined by j, the number of classes, and $P(C_i|v_1)$ represents the conditional probability of class i co-occurring with value 1.

Other possible metrics include alternative distance metrics for vector distances, such as the Jeffrey divergence metric (a symmetrical version of Kullback-Leibler distance), and the dot product metric or the cosine distance metric for numerical features. In addition, the k-NN algorithm can have several other algorithmic parameters such as the k parameter, feature weighting metrics, individual instance weighting metrics, and distance-weighted class voting among nearest neighbors. Feature weighting is an important parameter as its purpose is to assign higher weights to more important features [7]. A mismatch on a feature with a high weight will enlarge the distance between two instances more than a mismatch on a low weighted feature will. Some examples of feature weighting methods are Information Gain, Gain Ratio and Chi-square. In our experiments we employ the TiMBL software [8][1], which implements all of the mentioned optional distance metrics and weighting metrics.

Maximum-entropy modeling [9] is a statistical learning approach that learns a probability distribution from labeled training data. Maximum-entropy models (MAXENT) only represent what is known from the labeled training instances and assume as little as possible about what is unknown; MAXENT converges to a distribution with maximal entropy. Finding the distribution matrix between values and classes with the maximal entropy is done in an iterative way with algorithms such as L-BFGS [10]. In our experiments we use the maximum-entropy modeling software package *maxent* by Zhang Le [2].

Rule learning produces a set of classification rules based on a labeled training set. The condition part of the rules is, depending on the learner's rule grammar, a test on the presence of certain values in the input, combined with for example boolean operators. Many variants of rule learning exist, varying in the way the rules are induced or in the way the rules are applied. A prominent class of rule learners is those using *sequential covering*. In an iterative process they learn one rule at the time (prioritized by some maximized weighted function that considers coverage, accuracy, and byte length), and remove all examples from the data that are covered by this rule. We adopt RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [11] as the rule learning algorithm in our experiments[3]. Ripper can produce ordered and unordered rule sets. In classification, the first matching rule in a ordered rule set determines the class. For an unordered rule set, the matching rule with the lowest error on the training set determines the class.

2.1 *k*-NN and Maximum-Entropy Modeling

In this section we describe the construction and motivation of the hybrid algorithm that combines k-NN with maximum-entropy modeling.

[1] We ran experiments with TiMBL version 5.1.

[2] URL: `http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html`. We ran experiments with maxent version 20041229.

[3] In our experiments we used RIPPER version 2.5 (patch 1).

The distribution matrix structure of MAXENT is identical to the class distribution matrix structure between feature values and classes used by the MVDM metric of k-NN. We exploit this structural equivalence to construct the hybrid algorithm we will henceforth refer to as MAXENT-H. We employ the method proposed by [12]. After training MAXENT, we replace the class distribution matrix of the MVDM metric with the matrix produced by the maximum-entropy learning algorithm. We refer to this new metric as the MAXENT–MVDM distance metric.

By constructing this hybrid we hypothetically repair a known weakness of the MVDM metric of k-NN: its sensitivity to data sparseness. As the normal MVDM metric uses raw conditional probabilities calculated from frequency counts, two low-frequent feature values that accidently occur with the same class will be regarded as identical by MVDM; when they occur with different classes their distance is estimated as maximal. A re-estimation of probabilities such as produced by the maximum-entropy algorithm may smooth the MVDM metric.

Seen from the perspective of MAXENT, the major difference between MAXENT and the hybrid MAXENT-H is that the latter does not use the maximum-entropy matrix and the exponential maximum-entropy probability function to produce class likelihood estimates, but instead uses the MAXENT–MVDM distance metric to find the k nearest neighbors in the data, and extrapolates the neighbors' majority output class.

Relevant related work is reported in [4]. They compare instance-based learning (with MVDM metric) with Naive Bayes, and construct a range of intermediate hybrid variants, each more or less similar to Naive Bayes or the instance-based learner with MVDM. One of these variants has a close resemblance to our hybrid MAXENT-H, namely the variant that stores all training instances in memory, and uses the Naive Bayes metric to calculate distances between instances. The results of the reported experiments are quite diverse and inconclusive.

2.2 *k*-NN and Rule Learning

In this section we describe and motivate the construction of two different hybrid algorithms that combine rule learning with k-NN.

We use the rule set as induced by RIPPER to construct the hybrid algorithm, analogous to [13]. Per instance, whether it is in the training data or in the test data, we convert the rules into binary features that represent whether or not the rule fires on the instance. We generate two versions of the hybrid algorithm. In the first version the binary rule-features replace the original features in the instances. In other words, this operation transforms the original feature space into a new one [5]. We convert all training and test instances into this binary format and feed them to the instance-based learner. The hybrid subsequently uses the k-NN classification method to classify new examples. We refer to this hybrid as RULES-R-H, where the middle R denotes *replace*.

From the k-NN perspective, replacing the original features of the instances by rule-features can be considered as a compression and filtering step in which the rule learning algorithm has removed noise and irrelevant information, and grouped interacting feature values together of which k-NN is incapable. From

the perspective of the rule learning algorithm, we do not have the simple classification strategy of taking the class of the rule that fires first, but the local classification method of k-NN.

In the hybrids, rules are presented as active-inactive binary features, and more than one rule can be active for a particular instance. As k-NN can be used with $k > 1$, the nearest neighbors can also contain different active rules that are applied to the new instance. Several rules, instead of only one, may be involved in the classification.

In the second version of the hybrid, RULES-A-H, where A stands for *adding*, the rule-features are added to the original instance features. Thus, this hybrid is a k-NN classifier with extra added features that represent the per-instance firing patterns of the induced rule set. In this case the rule features cannot be considered as a compression and filtering step, but adding these rule-features modifies the distance calculations in k-NN. As explained above feature weighting in k-NN gives a higher weight to more important features. As many of the created rule-features will have a strong predictive power, they are likely to receive high feature weights, making them able to influence the distance calculation.

[3] also proposes a hybrid algorithm that combines rule learning with k-NN called 'RISE'. RISE applies creates a rule set by carefully generalizing instances. It searches for an optimal rule set by repeatedly finding the nearest instances, and generalizing over them. The most important difference between RISE and our approach is that RISE considers rules as generalized instances, while our approach differentiates between rules and instances as we transform rules to create features that are added to the instances or replace the original features in the instances.

3 Experimental Setup

We apply the three parent algorithms and the three hybrid algorithms on 29 data sets from the UCI repository of machine learning databases [14]. We perform 10-fold cross validation (CV) experiments and measure the mean accuracy and standard deviation on the ten folds. We conduct paired t-tests between outcomes of pairs of algorithms to determine the significance of the difference in performance.

k-NN and RULES offer several algorithmic parameters that, individually and in combination, can affect the functioning of the algorithms in unpredictable ways. We use a wrapped-based method to set them automatically for all k-NN and RULES modules involved in our study, including the hybrids. For small datasets it is feasible to run pseudo-exhaustively a large amount of wrapped validation experiments [15], covering all possible combinations of nominal parameter values and sequences of selected values of real-valued parameters. We do this for data sets below 1,000 instances: we perform wrapped internal 10-fold CV experiments nested within the main 10-fold CV experiments. We measure accuracy and select the average-best combination of settings over the internal ten folds.

For larger data sets a complete recombination of algorithmic parameter settings tested on the entire training set becomes infeasible. Rather than running the algorithms with their default settings, we adopt *wrapped progressive sampling*, or WPS [16], a heuristic automatic procedure that, on the basis of validation experiments internal to the training material, searches among algorithmic parameter combinations for a combination likely to yield optimal generalization performance on unseen data.

We test five algorithmic parameters of k-NN with a total of 925 parameter combinations, default settings are marked in bold:

- number of nearest neighbors: **1**, 3, 5, 7, 9, 11, 13, 15, 19, 25, 35;
- feature weighting: none, **gain ratio**, information gain, shared variance, chi-square;
- distance metric: **overlap**, MVDM, Jeffrey divergence;
- neighbor weighting: **normal majority voting**, inversed linear weighting, inversed distance weighting (only when $k > 1$)
- frequency threshold for switching from MVDM distance metric to overlap metric: 1, 2;

For the rule learning algorithm RIPPER we test seven algorithmic parameters which leads to a total of 972 parameter combinations to be tested:

- number of extra optimization rounds: 0, 1, **2**;
- order of the classes: starts by making rules for **the most frequent classes**, start with least frequent classes, unordered.
- rule simplification: **0.5**, 1.0, 2.0;
- misclassification cost: 0.5, **1.0**, 2.0;
- minimum number of instances covered by rule: **1**, 2, 5, 10, 20, 50;
- negative tests for nominal valued features: yes, **no**;

We did not optimize the parameters of MAXENT as it was shown in [16] that neither exhaustive wrapping nor WPS increased the generalization accuracy of this algorithm. We train MAXENT with L-BFGS parameter estimation, 100 iterations and a Gaussian prior with mean zero and σ^2 of 1.0.

Different machine learning algorithms have different methods to deal with continuous feature values. In order to rule out differences between algorithms we discretize the continuous features in some of the UCI benchmark tasks in a preprocessing step, using the entropy-based discretization method of [17].

4 Results

In this section we describe the results of all algorithms discussed in Section 2. Table 1 lists the names and number of instances of the 29 data sets, along with the mean accuracies and standard deviations of 10-fold CV experiments with all algorithms. (Note: *cl-h-disease* stands for 'cleveland-heart-disease', and *soybean-l* stands for 'soybean large'.)

We first look at the performance of the three parent machine learning algorithms. Table 2 shows the results of significance tests on the 29 UCI benchmarks

Table 1. Mean accuracy and standard deviation of the 10-fold CV experiments on the 29 UCI tasks for all algorithms. Best performances per task are printed in boldface.

task	# inst.	*k*-NN	MAXENT	RULES	MAXENT-H	RULES-R-H	RULES-A-H
abalone	4177	24.6 ± 2.8	23.6 ± 1.7	18.1 ± 1.7	22.8 ± 2.2	18.0 ± 1.7	**25.0** ± 2.5
audiology	226	80.5 ± 6.3	80.9 ± 5.0	76.5 ± 7.7	81.3 ± 5.8	61.0 ± 9.4	**81.8** ± 5.2
bridges	104	54.7 ± 10.6	**61.6** ± 9.1	53.8 ± 14.3	55.7 ± 13.1	52.9 ± 17.2	55.7 ± 13.1
car	1728	96.5 ± 1.3	90.9 ± 2.2	97.6 ± 1.1	96.5 ± 1.5	94.0 ± 4.0	**98.4** ± 0.9
cl-h-disease	303	55.7 ± 5.4	55.1 ± 5.0	**58.4** ± 5.9	**54.8** ± 5.8	**58.4** ± 5.9	**58.4** ± 5.2
connect4	67557	77.7 ± 1.8	75.7 ± 0.5	76.3 ± 1.7	78.1 ± 1.9	75.2 ± 1.3	**78.6** ± 2.5
ecoli	336	**79.5** ± 4.9	76.5 ± 7.8	69.7 ± 10.9	78.6 ± 2.8	72.6 ± 12.1	78.0 ± 6.3
flag	194	66.9 ± 11.4	**69.8** ± 13.4	61.8 ± 8.8	68.9 ± 14.9	61.8 ± 7.8	65.8 ± 10.9
glass	214	67.7 ± 8.4	**70.1** ± 11.6	60.7 ± 6.5	61.5 ± 9.9	60.8 ± 8.0	66.9 ± 10.1
kr-vs-kp	3196	96.8 ± 1.2	96.8 ± 0.6	**99.2** ± 0.5	99.1 ± 0.4	**99.2** ± 0.5	**99.2** ± 0.5
letter	20000	95.6 ± 0.5	85.0 ± 0.7	73.8 ± 1.5	**95.9** ± 0.7	74.6 ± 1.4	95.6 ± 0.4
lung-cancer	32	33.3 ± 12.9	39.2 ± 24.7	31.7 ± 24.1	**43.3** ± 13.3	25.0 ± 12.9	34.2 ± 16.0
monks1	432	**100.0** ± 0.0	75.0 ± 4.0	99.3 ± 2.0	93.7 ± 10.5	**100.0** ± 0.0	**100.0** ± 0.0
monks2	432	94.0 ± 11.8	65.1 ± 5.5	72.0 ± 8.1	96.3 ± 8.3	75.5 ± 12.2	**97.0** ± 4.2
monks3	432	**97.2** ± 2.5	**97.2** ± 2.5	**97.2** ± 2.5	**97.2** ± 2.5	96.5 ± 2.6	**97.2** ± 2.5
mushroom	8124	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	98.7 ± 2.6	**100.0** ± 0.0
nursery	12960	**99.4** ± 0.5	92.4 ± 0.4	97.7 ± 0.8	97.9 ± 1.1	97.9 ± 0.7	99.2 ± 0.4
optdigits	5620	**98.0** ± 0.7	95.8 ± 0.8	89.3 ± 1.0	97.3 ± 0.6	89.9 ± 1.0	97.9 ± 0.6
pendigits	10992	**93.4** ± 0.9	86.0 ± 1.2	82.6 ± 1.7	92.0 ± 1.4	81.7 ± 3.5	92.5 ± 1.5
promoters	106	87.0 ± 7.1	92.5 ± 9.0	79.4 ± 7.9	**93.5** ± 8.2	80.3 ± 8.7	83.5 ± 11.2
segment	2310	95.7 ± 0.9	92.1 ± 2.8	90.5 ± 3.6	95.9 ± 1.1	90.6 ± 3.6	**95.8** ± 1.3
solar-flare	1389	94.2 ± 2.2	**94.7** ± 1.8	94.6 ± 1.5	94.2 ± 1.8	94.6 ± 1.5	94.2 ± 1.9
soybean-l	683	92.8 ± 4.2	92.2 ± 2.8	91.1 ± 3.0	**93.1** ± 3.2	91.8 ± 3.2	92.8 ± 3.5
splice	3190	95.3 ± 1.0	94.6 ± 0.8	94.1 ± 1.6	94.8 ± 1.5	94.2 ± 1.1	**95.8** ± 0.7
tictactoe	958	95.8 ± 3.8	98.3 ± 0.7	99.7 ± 0.7	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0
vehicle	846	**67.6** ± 4.5	63.5 ± 5.2	55.7 ± 5.0	66.3 ± 5.4	56.0 ± 5.2	64.2 ± 5.7
votes	435	95.2 ± 2.4	**96.5** ± 2.1	94.2 ± 1.6	95.4 ± 2.3	94.2 ± 1.6	94.9 ± 1.3
wine	178	96.1 ± 2.6	94.9 ± 5.3	93.2 ± 6.5	**96.6** ± 2.7	92.7 ± 7.0	95.5 ± 4.9
yeast	1484	53.3 ± 2.9	49.3 ± 3.9	42.2 ± 3.2	**55.4** ± 3.0	40.2 ± 2.1	52.0 ± 4.7

Table 2. Comparison of the three parent algorithms through summary counts of won/tied/lost outcomes of paired *t*-tests on the mean accuracy and standard deviation of the 10-fold CV experiments on the 29 UCI tasks.

	k-NN	MAXENT	RULES
k-NN		10/17/2	14/12/3
MAXENT	2/17/10		10/13/6
RULES	3/12/14	6/13/10	

tasks of each of the algorithms compared to the other. Each cell shows the number of times the algorithm in the row won/tied/lost as compared to the algorithm in the column. The counts in the table are based on paired *t*-tests at $p < 0.05$ on the pairwise accuracies obtained in 10-fold CV experiments. Overall, the results indicate that *k*-NN performs better than the other two parent algorithms. MAXENT tends to perform better than RULES on 10 data sets, and RULES outperforms MAXENT on 6 data sets.

Comparisons (again in won/tied/lost counts) between the hybrid algorithms and their three parent algorithms are displayed in Table 3. We observe that MAXENT-H performs quite equally to *k*-NN, while outperforming MAXENT on 12

Table 3. Comparison of the three hybrids with their parent algorithms, through summary counts of won/tied/lost outcomes of paired t-tests on the mean accuracy and standard deviation of the 10-fold CV experiments on 29 UCI tasks.

	k-NN	MAXENT
MAXENT-H	4/21/4	12/17/0
	k-NN	RULES
RULES-R-H	2/14/13	3/23/3
RULES-A-H	4/24/1	16/13/0

data sets. RULES-R-H performs worse than k-NN: it wins only on two data sets and has a significantly lower accuracy in 13 cases. RULES-R-H performs very similarly to RULES. The second rule learning hybrid RULES-A-H has a significantly higher accuracy than k-NN on 4 data sets, and on 16 tasks compared to RULES.

5 Analysis

We are not solely interested in whether each hybrid has a better overall generalization performance than one or both of the parent algorithms. We also investigate to which extent the hybrid deviates functionally from its two parent algorithms. In this section we take a closer look at the degree of overlap in the errors made by the hybrids compared to their parents. Additionally, we measure the statistical bias and variance of the algorithms.

5.1 Complementary Error Rate Analysis

The complementary error rate between two algorithms A and B, $Comp(A, B)$, measures the percentage of mistakes that A makes which are not made by algorithm B [18]:

$$Comp(A, B) = \left(1 - \frac{\#\text{ of common errors}}{\#\text{ of errors of A only}}\right) * 100 \tag{2}$$

The relative magnitude of the complementary error rate between two algorithms can be seen as an indication of their functional similarity with respect to classification behavior. The lower the complimentary error rate between a pair of algorithms is, the more they are functionally similar.

We calculate the complementary rates between each hybrid compared to its two parent algorithms where the hybrid is A in $Comp(A, B)$. Table 4 (second column) lists macro averages over the 29 data sets. Almost all pairs of algorithms have complementary rates of more than 30%, meaning that at least one-third of the misclassified instances by the hybrid is classified correctly by the parent

Table 4. Complementary rates and overlapping errors between hybrids and their parent algorithms, macro-averaged over the 29 data sets

Two algorithms	Complementary rate	Error overlap
MAXENT-H – k-NN	32.9	60.0
MAXENT-H – MAXENT	34.9	54.2
RULES-A-H – k-NN	32.3	60.9
RULES-A-H – RULES	28.7	56.2
RULES-R-H – k-NN	54.8	32.0
RULES-R-H – RULES	21.3	74.3

classifier. The exception to this observation is the pair RULES-R-H – RULES which produces the low rate of 21.3%, indicating that their functional classification behavior is relatively similar.

Besides calculating whether the classifiers misclassify the same instances, we also count whether they make the same errors. We investigate the errors that are made by the hybrid algorithms and we calculate the percentage of times that the parent algorithms assign the same incorrect label. Table 4 (third column) displays the macro average of overlap in error labels on the 29 data sets. We see that the hybrids MAXENT-H and RULES-A-H have approximately 5% more overlap in error with k-NN than with the eager parent algorithm. The hybrid RULES-R-H makes the same errors as RULES to a very high extent (74%), while having little overlap (32%) with k-NN.

5.2 Bias–Variance Analysis

The expected average error of a classifier can be decomposed in three components: *statistical bias*, *variance* and *noise*. The statistical bias of an algorithm reflects the systematic error of the algorithms whereas the term variance expresses the variability in error over a set of different training sets. Noise presents the errors in the data.

In this section we analyze whether the hybrid algorithms have a different statistical bias and variance than their two parent algorithms. We employ the method of [19]: we perform sampling experiments, measure the average error rate and calculate the decomposition into bias and variance components.[4] We select 16 data sets from our original set of 29 that have more than 500 instances.

[19] use the following formula to decompose the expected zero-one loss $E(C)$ of discrete classifiers into bias and variance, given a fixed target and averaged over a sampling of training sets:

[4] We did not optimize the algorithmic parameters, as small training samples do not allow any reliable cross-validated wrapping.

$$E(C) = \sum_x p(x)(\sigma_x^2 + \text{bias}_x^2 + \text{variance}_x) \quad (3)$$

$$\text{bias}_x^2 \equiv 1/2 \sum_{y \epsilon Y} [p(Y_T = y|x) - p(Y_H = y|x)]^2 \quad (4)$$

$$\text{variance}_x \equiv 1/2(1 - \sum_{y \epsilon Y} P(Y_H = y|x)^2) \quad (5)$$

where x represents test example x, and σ_x^2 represents the noise in the data set ([19] argue to estimate noise to be zero as it is hard to calculate in practice); bias^2 (4) is estimated as the squared difference between the true target class and the predicted class, averaged over the training samples. (We refer to bias^2 as 'bias'.) Variance (5) is estimated as the variability over the different training sets. $p(Y_T = y|x)$ is the estimation that test example x is classified as y by the learning algorithm, averaged over the training set samples. $p(Y_F = y|x)$ is the probability that test example x has the true target label y, averaged over the training set samples. Both components are summed over all classes $y \epsilon Y$.

The purpose of these analyses is to investigate whether the proportion between bias and variance differs for the hybrids and their parent algorithms. In order to get a better view on the balance between bias and variance we scaled all error rates to 100%. Table 5 shows the macro averaged bias over the 16 data sets (the variance always being 100 − bias^2%). MAXENT has the highest bias; RULES-R-H has the lowest. When we compare each hybrid algorithm to its two parent algorithms, we see that all hybrids have a lower bias than k-NN, and also lower than the other parent algorithm. The three hybrids make less systematic errors than both of their parent algorithms.

Table 5. The scaled bias component in the error of all algorithms, macro averaged over the 16 data sets

Algorithm	Bias	Algorithm	Bias
k-NN	57.29	MAXENT-H	56.92
MAXENT	65.56	RULES-R-H	51.63
RULES	55.50	RULES-A-H	54.99

6 Discussion

Our experiments have brought forward evidence in two cases that instance-based classification can replace other classification procedures successfully. We constructed hybrids in which the learning component consisted either of rule learning or of maximum-entropy modeling, and in which the classification was performed with the k-NN classification rule. When comparing the hybrid algorithms to their parent algorithms, we observed that MAXENT-H and RULES-A-H both outperform the eager parent algorithm often, and are never significantly

outperformed by them. At the same time these two hybrids perform almost identically to k-NN. When investigating more deeply to which extent the functional behavior of the hybrid algorithms differs from their parent algorithms by error analysis, we see that the hybrids MAXENT-H and RULES-A-H both misclassify different instances than both their parent algorithms in at least 30% of the cases, while RULES-R-H functions quite similarly to RULES. The two successful hybrids, MAXENT-H and RULES-A-H, have a slight higher overlap with k-NN of approximately 5% compared to the error overlap with either RULES or MAXENT.

An intriguing observation is that the bias of the three hybrids is lower than that of their parent algorithms. Combining the observed performance differences (Table 3) and the bias components of all algorithms (Table 5), we can assume that the performance gains of MAXENT-H and RULES-A-H over MAXENT and RULES, respectively, are due to a decrease in the number of systematic errors the hybrids generate. Given that the relative bias components of the two hybrids are also lower than that of k-NN, at virtually no loss of performance, we conclude that these two hybrids, MAXENT-H and RULES-A-H, represent a "best of both worlds" situation, since their different-source components cause them to avoid systematic errors their parent algorithms make.

The hybrid RULES-R-H shows a different behavior than the other two hybrids. RULES-R-H performs worse than k-NN and equals the performance of the rule learning algorithm. Also, complementary rates and overlap in error show that RULES-R-H has a quite similar functional classification behavior to that of the rule learning algorithm. Our expectation was that the hybrid would differ from RULES in classification behavior, as more than one binary rule-feature can be active in the feature representation of the hybrid and larger k values allow several nearest instances with different active bits to be involved. In our experiments on the 29 data sets, the average number of active binary rule-features in the training folds was 1.3 bits on average 67.5 rule-features per instance. However, in 70% of the experiments with RULES-R-H the automatic algorithmic parameter selection has chosen the k value to be 1, meaning that the potential benefit of k-NN classification is not fully explored. The hybrid RULES-A-H uses $k = 1$ in only 23% of the experiments, thereby profiting from the k-NN classification method.

In future work we plan to compare the hybrids to external classifier combination schemes. As our hybrids, classifier combination schemes benefit from combining partly complementary classifier biases; our method has the intrinsic advantage that the resulting classifier is one integrated model rather than two.

Acknowledgments

This research is funded by the Netherlands Organization for Scientific Research (NWO). The authors wish to thank Zhang Le for sharing information on the internals of his MAXENT implementation.

References

1. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. Institute of Electrical and Electronics Engineers Transactions on Information Theory **13** (1967) 21–27
2. Aha, D.W., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning **6** (1991) 37–66
3. Domingos, P.: Unifying instance-based and rule-based induction. Machine Learning **24** (1996) 141–168
4. Ting, K., Cameron-Jones, R.: Exploring a framework for instance based learning and naive bayesian classifiers. In: Proceedings of the Seventh Australian Joint Conference on Artificial Intelligence. (1994) 100–107
5. Sebag, M., Schoenauer, M.: A rule-based similarity measure. In: Topics in case-based reasoning. Springer Verlag (1994) 119–130
6. Cost, S., Salzberg, S.: A weighted nearest neighbour algorithm for learning with symbolic features. Machine Learning **10** (1993) 57–78
7. Wettschereck, D., Aha, D.W., Mohri, T.: A review and comparative evaluation of feature-weighting methods for a class of lazy learning algorithms. Artificial Intelligence Review, special issue on Lazy Learning **11** (1997) 273–314
8. Daelemans, W., Zavrel, J., Van der Sloot, K., Van den Bosch, A.: TiMBL: Tilburg Memory Based Learner, version 5.1, reference manual. Technical Report ILK-0402, ILK, Tilburg University (2004)
9. Guiasu, S., Shenitzer, A.: The principle of maximum entropy. The Mathematical Intelligencer **7** (1985)
10. Nocedal, J.: Updating quasi-Newton matrices with limited storage. Mathematics of Computation **35** (1980) 773–782
11. Cohen, W.: Fast effective rule induction. In: Proceedings 12th International Conference on Machine Learning. (1995) 115–123
12. Hendrickx, I., Van den Bosch, A.: Maximum-entropy parameter estimation for the k-nn modified value-difference kernel. In: Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence. (2004) 19–26
13. Van den Bosch, A.: Feature transformation through rule induction, a case study with the k-nn classifier. In: Proceedings on the workshop on advances in Inductive rule learning at the ECML/PKDD 2004. (2004) 1–15
14. Blake, C., Merz, C.: UCI repository of machine learning databases (1998) `http://www.ics.uci.edu/ mlearn/MLRepository.html`.
15. Kohavi, R., John, G.: Wrappers for feature subset selection. Artificial Intelligence Journal **97** (1997) 273–324
16. Van den Bosch, A.: Wrapped progressive sampling search for optimizing learning algorithm parameters. In: Proceedings of the 16th Belgian-Dutch Conference on Artificial Intelligence. (2004) 219–226
17. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence. (1993) 1022–1027
18. Brill, E., Wu, J.: Classifier combination for improved lexical disambiguation. In: Proceedings of the COLING-ACL'1998. (1998) 191–195
19. Kohavi, R., Wolpert, D.: Bias plus variance decomposition for zero-one loss functions. In: Proceedings of the thirteenth International Conference on Machine Learning. (1996) 275–283