Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning, D. Leake and E. Plaza, eds., Lecture Notes in Aritificial Intelligence, Springer, 455-466, 1997.

Examining Locally Varying Weights for Nearest Neighbor Algorithms

Nicholas Howe and Claire Cardie

Department of Computer Science, Cornell University, Ithaca NY 14850.
E-mail: nihowe@cs.cornell.edu; cardie@cs.cornell.edu

Abstract. Previous work on feature weighting for case-based learning algorithms has tended to use either global weights or weights that vary over extremely local regions of the case space. This paper examines the use of coarsely local weighting schemes, where feature weights are allowed to vary but are identical for groups or clusters of cases. We present a new technique, called class distribution weighting (CDW), that allows weights to vary at the class level. We further extend CDW into a family of related techniques that exhibit varying degrees of locality, from global to local. The class distribution techniques are then applied to a set of eleven concept learning tasks. We find that one or more of the CDW variants significantly improves classification accuracy for nine of the eleven tasks. In addition, we find that the relative importance of classes, features, and feature values in a particular domain determines which variant is most successful.

1 Introduction

The k-nearest-neighbor (k-NN) algorithm is among the oldest of classification schemes for case-based learning. It lies at the heart of many case-based (or instance-based) learning algorithms. (See, for example, Aha et al. (1991)). Given a test case described as pairs of features and values, k-NN finds previously seen cases with the most similar feature values and uses them to predict the class of the new instance. The algorithm works well for some tasks, depending on the type and complexity of the concept to be learned. Researchers have proposed numerous variations of k-NN in an effort to improve its effectiveness on more difficult tasks. In particular, many proposed schemes employ *feature weighting:* the contribution of a feature in calculating the nearest neighbors is scaled according to the importance of the feature. The collection of weights, one per feature, forms a *weight vector*. See Wettschereck, et al. (1997) for a useful survey of feature weighting methods.

Many feature weighting methods apply weights globally: they use a single weight vector that remains constant throughout testing. However, some domains contain features that vary in importance across the instance space (Aha and Goldstone, 1992). *Local weighting* schemes, where feature weights can vary from instance to instance or feature value to feature value, may perform better for such applications. For example, Aha and Goldstone (1992) use a combination of local and global weights for each training instance, and Hastie and Tibshirani (1994) use weights produced individually for each test instance. Stanfill and Waltz's (1986) value difference metric (VDM) takes a slightly different approach by weighting features according to the particular feature values of the test case and individual training cases. Potentially, local weighting schemes can take into account any combination of global, test-case, and training-case data. The locality of particular weighting algorithms can be visualised on a continuum, from global methods that compute a single weight vector for all cases to extremely local methods that compute a different weight vector for each pair of test and training cases. This paper uses the term "local weighting" to describe any scheme in which the computed weights may vary depending on the classes of the cases being compared, their feature values, or other variables. The number of locally varying parameters the metric depends on determines the "degree of locality."

In spite of the existing work on individualized local weighting, less attention has been paid to local weighting on a coarser scale. While using individualized feature weights for each training instance or each test instance is a powerful approach, it may not be the best for all tasks. For example, using individual weights is unnecessary if the important features are the same across larger groups of instances. Statistical properties of larger homogeneous groups may simplify the task of computing appropriate weights.

This paper presents a coarsely local feature weighting scheme, *class distribution weighting* (CDW), that allows weights to vary at the class level. Although classes are certainly not always homogeneous, it is plausible that for many domains the defining features of a class are the same for most or all of the instances belonging to it. Instead of a single global weight vector, CDW computes a different weight vector for each class in the set of training cases using statistical properties of that subset of the data. Furthermore, the CDW scheme can be easily modified to generate a family of related feature weighting methods. Thus we can choose to apply a single set of global weights or to allow finer scale localization that accounts for the importance of particular feature value combinations. Although the algorithms considered here apply only to features with discrete (i.e., symbolic) attribute values, they can potentially be generalized for continuous (i.e., numeric) attributes.

In this paper, we apply CDW and its variants to a collection of classification tasks, and present evidence that the optimal amount of locality for feature weighting varies between the different tasks. For nine of the eleven data sets used, at least one of the CDW weighting schemes significantly improved classification accuracy. (In the other two, none of the fluctuation in results was statistically significant.) With $k = 1$, the most local technique tested produced the most accurate results for seven of the nine tasks for which results varied significantly, but showed significantly lower accuracies for the remaining two tasks. Given the variability, we conclude that it is advantageous to have a family of related techniques (like the CDW family): the best method for each task can be selected via cross-validation on the training cases, or can be based in many cases on relatively simple properties of the task (e.g., the presence of irrelevant features).

The remainder of this paper describes the algorithms and their performance. Section 2 describes CDW and its variants. Section 3 analyzes the results of applying the different algorithms to a set of classification tasks, and discusses why certain algorithms perform better than others in specific tasks. Finally, Sect. 4 discusses related work, including other coarsely local weighting algorithms, and Sect. 5 concludes with possible further extensions of CDW.

2 Class Distribution Weighting Algorithms

The class distribution weighting (CDW) algorithm and its variants start from the premise that the features that are important to match on are those that tend to have different values associated with different classes. An ideal feature would take on a unique set of values for each class. If it existed, that feature would provide all class information readily. In most applications, of course, ideal features are not available, but we can measure the degree to which each feature approximates the ideal. We then weight each feature proportionally to this measurement of the amount of information it provides.

We measure the usefulness of a feature for classification by comparing the distributions of the feature values across various subsets of the training cases. CDW computes a different set of weights for each class in the training set. The weights for a particular class on a given feature are based on a comparison between the distribution of feature values for the cases in that class and the distribution of values for cases in all other classes. If the distributions are highly similar, the feature is considered not useful for distinguishing that class from others, and it is assigned a low weight. If the distributions are highly dissimilar, the feature is considered useful, and it is assigned a high weight.

During classification, we use a variation of the standard weighted k-NN algorithm for symbolic features. Given a test case τ, the proximity to each training case t_k is calculated by

$$D(t_k, \tau) = \sum_{i=1}^{m} \delta_{f_i}(t_k, \tau) W_{f_i \mathrm{Cl}(t_k)} \tag{1}$$

where $W_{f_i \mathrm{Cl}(t_k)}$ refers to the weight of feature f_i as computed for the class of t_k, and

$$\delta_{f_i}(t_k, \tau) = \begin{cases} 1 \text{ if } \mathrm{Val}(f_i, t_k) = \mathrm{Val}(f_i, \tau) \\ 0 \text{ otherwise} \end{cases} \quad . \tag{2}$$

Training instances with higher scores are considered closer to the test instance.

For each class C_j there exists a separate vector $\langle W_{f_1 C_j}, ..., W_{f_m C_j} \rangle$ of weights for each feature. The weights $W_{f_i C_j}$ are calculated as follows. Let $T = \{t_1, ..., t_n\}$ be the set of training examples, and let $F = \{f_1, ..., f_m\}$ be the set of features describing cases in T. Suppose that feature f_i takes on the r_i different values $v_{f_i 1}, ..., v_{f_i r_i}$ across the entire training set. We then define the distribution of feature values for feature f_i over an arbitrary subset $S = \{t_{q_1}, ..., t_{q_s}\} \subseteq T$ as a vector $\Psi(f_i, S) = \langle a_1, ..., a_{r_i} \rangle$ of length r_i such that

$$a_h(f_i, S) = \frac{1}{|S|} \sum_{k=1}^{|S|} \delta_{hik} \tag{3}$$

where

$$\delta_{hik} = \begin{cases} 1 \text{ if } \mathrm{Val}(f_i, t_{q_k}) = v_h \\ 0 \text{ otherwise} \end{cases} . \tag{4}$$

In other words, a_h is the fraction of instances across S where f_i takes on the value v_h.

Let $C_1, ..., C_p$ be subsets of T grouped by class (i.e., C_1 consists of all training cases in class 1, etc.). To find the raw weight for feature f_i and class C_j, we compare the distribution of values for C_j to that of the rest of the training examples:

$$R_{f_i C_j} = \|\Psi(f_i, C_j) - \Psi(f_i, T - C_j)\|_1 = \sum_{h=1}^{r_i} |a_h(f_i, C_j) - a_h(f_i, T - C_j)| \ . \tag{5}$$

This yields a raw weight vector $\left\langle R_{f_1 C_j}, ..., R_{f_m C_j} \right\rangle$ for each class C_j.[1] The final weights $\left\langle W_{f_1 C_j}, ..., W_{f_m C_j} \right\rangle$ used are simply the raw weights normalized to sum to 1.

During classification, the k nearest neighbors are calculated using (1). In case of ties, more than k cases may be returned. (In fact, to account for floating point rounding errors, all cases with scores within a small ϵ of the kth closest case are returned.) The retrieved cases then vote on the classification, and ties are broken by taking the first instance returned.

2.1 CDW Variants

CDW weights locally by class level groupings. Thus it should perform well in domains with homogeneous classes that are distinguished by different features particular to each class. The same approach can be applied to compute weights for groups more finely or coarsely grained than individual classes. For example, if a particular class was defined by the disjunction of two distinct subconcepts, then one might wish to compute different weights for the two sets of instances belonging to each subconcept. Unfortunately, subclass groupings are generally not known a priori, and computing appropriate groupings is not a straightforward problem. Because of these difficulties, we have not pursued this particular approach further. Instead, we examined three variants of the CDW algorithm: one that eliminates locality by using global weights derived from the local CDW weights, another that uses finer-grained locality by associating different weights with each individual feature value, and a third that is a straightforward combination of the first two.

[1] In (5), we could use the 2-norm or any Minkowski p-norm. Our tests indicate that the resulting algorithm behaves similarly to standard CDW on most data sets. In all results presented in this paper we use the standard form given above.

Global Mean CDW To go from the local CDW weights to a global weight for all features, we average the feature weight vectors across all classes to get a single global weight vector. This variant can be expected to perform well in domains where the relevant features are the same for all classes (e.g., *LED-24* from the UCI repository (Merz and Murphy, 1996)). The global weights can be computed by taking a simple mean over all classes or by an average weighted by the class frequency in the training data. The latter approach gives comparable overall results, but tends to bias predictions toward the most common classes. Because recent work has emphasized the importance of minority class predictions (Fawcett, 1996), we present only results for the simple mean here. We call this method *global mean CDW* (GM-CDW).

In domains with only two possible classes, CDW and GM-CDW will calculate the same set of weight vectors because CDW itself produces identical weights for the two classes. An examination of (5) reveals the reason: $T - C_1 = C_2$ and $T - C_2 = C_1$ if C_1 and C_2 are the only classes. Thus (5) degenerates to equivalent expressions for each class.

Expanded Feature CDW For some classes, particular feature values may be more significant than other values of the same feature. For example, the target class in the *Monks-2* data set is defined by the concept "exactly two features have the value 1." Thus another potentially useful form of local feature weighting allows the weights to vary locally according to the values of the test instance. A simple transformation of the case base allows the CDW algorithm to exploit this form of locality. Specifically, the feature set is expanded so that each instance is described by a set of binary features corresponding to all the feature value possibilities in the original training set. If instances in the training set T are described by the set of features $F = \{f_1, ..., f_m\}$, and feature f_i takes on values $V_{f_i} = \{v_{f_i1}, ..., v_{f_ir_i}\}$ across the entire training set, then instances in the transformed training set T' are described by the feature set $F' = V_{f_1} \cup V_{f_2} \cup ... \cup V_{f_m}$.

Since the transformed data set has a separate feature for each original feature value in the training cases, the CDW algorithm applied to it generates weights that vary for individual feature values. This can be described as a new form of local distance metric on the original data set, where the distance contribution from each feature is weighted according to the class of the training instance, and the feature's value in the two cases being compared:

$$D(t_k, \tau) = \sum_{i=1}^{m} d_{f_i}(t_k, \tau) \ . \tag{6}$$

Here the separate weight and δ function from (1) have been subsumed into the single weighted distance term

$$d_{f_i}(t_k, \tau) = (\delta_{f_i}(t_k, \tau) - 1)\left(W_{\mathrm{Val}(f_i,t_k)\mathrm{Cl}(t_k)} + W_{\mathrm{Val}(f_i,\tau)\mathrm{Cl}(t_k)}\right) + \sum_{j=1}^{r_i} W_{v_{f_ij}\mathrm{Cl}(t_k)} \tag{7}$$

where $W_{v_{f_i j}\mathrm{Cl}(t_k)}$ is the weight assigned to the binary expanded feature $v_{f_i j}$ for the class of the training case t_k. In other words, $d_{f_i}(t_k, \tau)$ is the sum of all the value weights $W_{v_{f_i j}\mathrm{Cl}(t_k)}$ minus the value weights for the training and test case if their values differ. This method, which we call *expanded feature CDW* (EF-CDW) has finer locality than the standard CDW since it varies with individual feature values. It should perform best on tasks in which some but not all values of a feature are important, and where the importance of each value varies from class to class.

EF-CDW is identical to CDW for domains whose features all have binary values, e.g., *LED-7* and *LED-24*. This is because feature expansion on binary features makes two new features with similar distributions. (One is the mirror of the other.) The two expansion features are each assigned weights that are half the normal CDW weights, and the relative distances are unchanged. The relative ranks of unweighted k-NN distances are also unchanged by feature expansion.

Global Mean Expanded Feature CDW This variant is a straightforward combination of GM-CDW and EF-CDW. The instances are transformed to the expanded-features format, the standard CDW algorithm is applied, and the expanded-feature weights on the classes are averaged to get global weights for each expanded feature. This variant exhibits test case locality but not class locality. It should do especially well on tasks where only certain feature values are relevant, but the relevant values do not vary from class to class (e.g., *Monks-2*).

3 Results of Testing

We use a variety of classification tasks to test the different weighting algorithms. Because we hypothesize that different data sets will require differing degrees of locality for greatest accuracy, we include a range of artificial and real-world domains. The data sets used are shown in the leftmost column of Table 1. The first six of these (*LED-7*, *LED-24*, *Monks-2*, *Lymph*, *Promoters*, and *Soybean*)[2] are from the UCI machine learning repository (Merz and Murphy, 1996). The tasks selected are a subset of those proposed as a benchmark by Zheng (1993). Tasks

[2] *LED-7* is the task of identifying the digit on a standard 7-LED digital display, with approximately 10% of the features flipped (to simulate random noise). Due to the noise, the optimal probability of a correct classification is about 74%. *LED-24* is the same task with an additional 17 irrelevant features that serve as distractors. The data sets used for these tasks each have 250 instances. *Monks-2* is an artificial data set with 6 features, where the class description to be learned is "exactly two features have the value 1." It has 432 test instances, of which 169 are designated as training cases. *Lymph* is a set of 159 medical cases provided by Zwitter and Soklic. The task is to predict a medical diagnosis given 18 descriptive features. *Promoters* is a set of 106 *E. coli* DNA sequences, where the task is to predict whether the sequence will act as a promoter. *Soybean* is a collection of crop disease records. The task is to predict the disease given 35 features providing information about the growing conditions. This task has 307 designated training instances, and 376 designated test instances.

from the benchmark with continuous features were discarded. Also, *NetTalk* was not used because of its similarity to the NLP datasets described below, and *Mushroom* was found to be too easy.

We also include an artificial task constructed specifically to exhibit feature importance that varies locally at the class level, and several problems from natural language processing (NLP) (Cardie, 1993a; Cardie, 1993b). *Construct* is an artificially constructed 200-instance data set designed to showcase the strength of the CDW algorithm. It consists of ten features with random values from 0 to 9, with one feature set at random to 10. The class of an instance is the number of the feature that is set at ten. *POS*, *Gen-Sem*, *Spec-Sem*, and *Concept* are NLP data sets of unknown words and are described in detail in Cardie (1993a). Briefly, the learning task involves predicting the part of speech, general and specific semantic class, and concept activation respectively for unknown words drawn from the MUC business joint venture corpus (MUC-5, 1994). In addition to the class value, each case is described by 34 features that encode information about the local and global context of the unknown word (Cardie, 1993b).

In the experiments below, ten-fold cross-validation was used for all tasks, except for two domains with designated training and test sets (*Monks* and*Soybean*). For these tasks, the designated sets were used to provide consistency with previous work.

3.1 Discussion

The results of the tests performed are shown in Table 1. Parentheses around an entry indicate that a particular test is degenerate with one of the other variants, as noted above. Bold face type indicates significance of at least .10 with respect to k-NN in a chi-square test, and footnotes indicate greater significance where applicable. Italic type indicates a significant decrease in accuracy.

Except for two tasks for which no results are statistically distinguishable (*LED-7* and *Soybean*), at least one of the CDW variants significantly outperforms the k-NN baseline. While the results tend to favor increased locality in weighting, no single variant is clearly superior for all tasks. Although Schaffer has shown theoretically that no single learning algorithm is best for every task (Schaffer, 1994), our results show empiricially the effects of local variation in the distance metric on typical tasks. For some of the tasks we use, locally varying metrics bring no significant improvement in accuracy. Because the methods we compare are all variants of a single technique, we suggest that the lack of improvement stems from the intrinsic nature of these domains.

In particular, the CDW algorithm should attain higher accuracies for tasks where feature importance varies according to the class. For example, CDW shows high accuracy on *Construct* which is designed to exhibit varying feature importance. Interestingly, CDW performs significantly worse on all the NLP tasks. We suspect that the important features for these tasks are unrelated to the class of the instance. CDW may be basing its weights on spurious patterns in the training data, lowering its accuracy.

Table 1. Accuracy of CDW Variants Compared to 1-NN and 10-NN

Data Set	NN	CDW	GM-CDW	EF-CDW	GMEF-CDW	k Value
LED-7	64.80	63.20	63.20	(63.20)	(63.20)	$k=1$
	72.40	70.40	68.80	(70.40)	(68.80)	$k=10$
LED-24	44.40	**64.00**‡	**64.40**‡	(**64.00**)‡	(**64.40**)‡	$k=1$
	59.60	**73.60**‡	**77.20**‡	(**73.60**)‡	(**77.20**)‡	$k=10$
Monks-2	70.83	68.52	(68.52)	**75.00**†	(**75.00**)†	$k=1$
	66.90	65.97	(65.97)	*62.73*†	(*62.73*)†	$k=10$
Lymph	81.43	84.29	83.57	**86.43**	**86.43**	$k=1$
	82.86	83.57	81.43	83.57	83.57	$k=10$
Promoters	85.00	**90.00**	(**90.00**)	**91.00**	(**91.00**)	$k=1$
	78.00	**87.00**†	(**87.00**)†	**89.00**‡	(**89.00**)‡	$k=10$
Soybean	88.03	88.83	88.56	87.50	89.89	$k=1$
	80.59	80.32	81.91	78.72	83.24	$k=10$
Construct	64.00	**98.00**‡	*58.00*†	**100.00**‡	**99.50**‡	$k=1$
	85.50	**100.00**‡	*76.50*‡	**100.00**‡	**100.00**‡	$k=10$
POS	89.45	*88.04*†	**91.00**†	**91.73**‡	**93.63**‡	$k=1$
	88.67	*86.87*‡	**90.61**‡	**93.63**‡	**93.53**‡	$k=10$
Gen-Sem	64.59	*61.72*‡	**67.46**‡	**71.25**‡	**75.63**‡	$k=1$
	67.02	*62.40*‡	**69.36**†	**75.63**‡	**75.19**‡	$k=10$
Spec-Sem	71.79	*69.36*‡	72.28	**81.81**‡	**78.70**‡	$k=1$
	75.92	*72.91*‡	76.85	**78.70**‡	**80.64**‡	$k=10$
Concept	91.39	*88.57*‡	91.44	**93.43**‡	92.17	$k=1$
	92.95	*92.07*	93.24	*92.17*	93.39	$k=10$

Key: Results in parentheses are duplicates of other tests. **Bold face type** indicates significance with respect to NN of at least .10. *Italic type* indicates a significant decrease with respect to NN. † Indicates significance with respect to NN of at least .05. ‡ Indicates significance with respect to NN of at least .01.

The GM-CDW algorithm should do best in domains where the important features are the same regardless of class. As expected, it performs well in *LED-24*, demonstrating its ability to discard globally irrelevant features. In tasks like *LED-7*, where all features are important, neither GM-CDW nor any of the other CDW variants should have any particular advantage over k-NN, and the results reflect this. The remarkable uniformity of the *Soybean* results may seem to indicate a similar uniformity in feature importance. More probably, however, the case space is densely populated enough for this task that k-NN does not suffer in comparison with more sophisticated techniques. Wettschereck et al. (1997) report unexpectedly high accuracy for k-NN on a different task due to this effect.

The majority of the domains show the most improvement for the two binarized variants (EF-CDW and GMEF-CDW), which yield the most locally-tailored feature weights. The *Monks-2* results favor EF-CDW, which is not surprising because its concept definition explicitly refers to specific feature values.

Promoters, *Construct*, *Lymph*, and the four NLP data sets also respond well to the expanded-feature variants. It is worth noting that GMEF-CDW tends to work well for precisely the same tasks as GM-CDW and EF-CDW. This suggests that the relationships of feature importance to class and to particular feature values are independent of each other.

It may seem likely that many real-world tasks, due to their complexity, will respond well to increased locality in feature weighting. Our results show that while this conjecture is often true, it does not hold in all cases. In the results for $k = 1$, the most localized algorithm (EF-CDW) yields the highest accuracies for seven of the data sets tested (*LED-7*, *Monks-2*, *Lymph*, *Promoters*, *Construct*, *Spec-Sem*, and *Concept*). Four other tasks (*LED-24*, *Soybean*, *POS*, and *Gen-Sem*) show the best results with other algorithms for $k = 1$, and the difference is significant for the two NLP tasks. Choosing k carefully may help, since the pattern for $k = 10$ is slightly different. Still, the LED and NLP tasks provide evidence that allowing for variation that does not exist in the data can decrease accuracy on some tasks. Naturally these results may not extend to other tasks and algorithms not tested. However, based upon our results, we recommend pre-testing via cross-validation with varying types of locally-dependent metrics in order to empirically determine the optimum for a particular task. Alternately, expert knowledge, if available, can be used to select the best approach to use.

Each of the CDW variants has several tasks at which it performs well. This lends support to the "family of algorithms" approach. Overall, we find that CDW and GM-CDW are good at tasks with irrelevant features, and EF-CDW and GMEF-CDW are particularly good at tasks with partially relevant or interacting features. Together, they can handle many types of classification problems.

4 Related Work

Several surveys consider locality in k-NN variants. Atkeson et al. (1997a) survey locally weighted learning algorithms for numeric (e.g., continuous-valued) functions, including k-NN variants. A companion paper examines the application of various locally weighted techniques to practical robotic control problems (Atkeson *et al.*, 1997b). Researchers have also examined local similarity metrics based upon domain-specific knowledge (Cain *et al.*, 1991; Skalak, 1992).

Wettschereck et al. (1997) survey lazy learning algorithms, which include k-NN algorithms. One of their dimensions for comparing algorithms is the generality of the weighting scheme. They cite several studies reporting good results for locally weighted techniques (see Hastie and Tibshirani (1994) and Friedman (1994)), and conclude that the subject merits more research. Although they compare algorithms along several of the dimensions they define, the test results they present do not focus on comparing algorithms with differing degrees of locality.

Several previously introduced classifiers share similarities with CDW and its variants. For example, Stanfill and Waltz's (1986) VDM computes the feature value distributions, but unlike CDW the weights it computes do not depend on the class of the training instance. VDM computes different distances between symbolic feature values, and also weights the features based on the feature value of the test case. This can be viewed as weighting features locally based on both the training and test cases, although the computed distance between any two given feature values is the same across the entire data set. Of the methods presented here, VDM is most similar to GMEF-CDW.

Several feature weighting classifiers have used weights that, like CDW, vary at the class level. Per-category feature importance (PCF) (Creecy *et al.*, 1992) binarizes features in the manner we have been calling "feature expansion", and then computes weights according to the formula

$$W_{f_i C_j} = P(C_j | f_i) \ . \tag{8}$$

Thus it assigns high weight to features that are highly correlated with the class. Unlike CDW, PCF fails to distinguish between a feature that tends to take on a particular value across the entire data set and one which tends to be on only for a particular class. Mohri and Tanaka (1994) report that PCF is biased toward predicting the majority class for data sets with a skewed class distribution.

Aha's IB4 classifier also calculates a different weight vector for each class (Aha, 1992). It attempts to learn feature weights by cycling through the training instances and adjusting their values. Weights are strengthened if feature values match for instances of the same class, and weakened if the values match but the instances are of different classes. Unlike CDW, IB4 is sensitive to the presentation order of training instances, and it assumes that the irrelevant feature values are uniformly distributed (Kira and Rendell, 1992). Aha reports that IB4 outperforms 1-NN in some domains with irrelevant features, and the fact that weights are learned allows it to change its bias to match that required by a particular task (Wettschereck *et al.*, 1997).

5 Conclusions

We have developed a family of feature weighting techniques that vary in the degree of locality with which the feature weights are calculated. We present results of tests showing that at least one of the CDW variants significantly improves classification accuracy for nine of eleven benchmark classification tasks. Because no single technique proved to be the best in every task, we conclude that different tasks require differing degrees of locality in feature weighting. This justifies the use of a family of techniques, and suggests that some pre-testing using cross-validation on a particular task is necessary in order to determine the amount of locality required.

We are considering a number of improvements and extensions to the CDW algorithms. First, the CDW weighting algorithm could be extended to process numeric features in addition to symbolic ones. The most straightforward way to do this is to partition numeric features into histogram buckets. However, this discards some of the information present in the numeric values. A better extension would take into account the continuous nature of numeric features while preserving the paradigm that the weight of a feature should be based directly on its usefulness in distinguishing classes.

In addition, researchers have noted the superior performance of adaptive weight learning techniques, which attempt to adjust their bias to match that of the task (Wettschereck *et al.*, 1997). Cross-validation on the training data to find the optimal level of locality may be flexible enough for most purposes. However, other feedback systems could be developed based upon the wrapper model introduced by John et al. (1994).

Finally, we would like to take advantage of the flexibility of CDW by using criteria other than the instance class to divide the training set into regions. Different criteria may divide the case base into more homogeneous groups in terms of feature importance. Applying the techniques of CDW to appropriate groupings should yield further improvements in accuracy.

6 Acknowledgements

We thank David Skalak and the anonymous reviewers for their advice and suggestions on this work. This work was supported in part by NSF CAREER Award IRI–9624639 and NSF Grant GER–9454149.

References

(Aha and Goldstone, 1992) Aha, D. W. and Goldstone, R. L. 1992. Concept learning and flexible weighting. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, IN. The Cognitive Science Society, Lawrence Erlbaum Associates. 534–539.

(Aha *et al.*, 1991) Aha, D. W.; Kibler, D.; and Goldstone, R.L. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.

(Aha, 1992) Aha, D. W. 1992. Tolerating noisy, irrelevant, and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies* 36:267–287.

(Atkeson *et al.*, 1997a) Atkeson, C. G.; Moore, A. W.; and Schaal, S. 1997a. Locally weighted learning. *Artificial Intelligence Review* Special Issue on Lazy Learning Algorithms.

(Atkeson *et al.*, 1997b) Atkeson, C. G.; Moore, A. W.; and Schaal, S. 1997b. Locally weighted learning for control. *Artificial Intelligence Review* Special Issue on Lazy Learning Algorithms.

(Cain *et al.*, 1991) Cain, T.; Pazzani, M. J.; and Silverstein, G. 1991. Using domain knowledge to influence similarity judgement. In *Proceedings of the Case-Based Reasoning Workshop*, Washington, DC. Morgan Kaufmann. 191–199.

(Cardie, 1993a) Cardie, C. 1993a. A Case-Based Approach to Knowledge Acquisition for Domain-Specific Sentence Analysis. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC. AAAI Press / MIT Press. 798–803.

(Cardie, 1993b) Cardie, C. 1993b. Using Decision Trees to Improve Case-Based Learning. In Utgoff, P., editor, *Proceedings of the Tenth International Conference on*

Machine Learning, University of Massachusetts, Amherst, MA. Morgan Kaufmann. 25–32.

(Creecy *et al.*, 1992) Creecy, R. H.; Masand, B. M.; Smith, S. J.; and Waltz, D. L. 1992. Trading mips and memory for knowledge engineering. *Communications of the ACM* 35:48–64.

(Fawcett, 1996) Fawcett, T. 1996. *Learning with Skewed Class Distributions — Summary of Responses*. Machine Learning List: Vol. 8, No. 20.

(Friedman, 1994) Friedman, J. H. 1994. Flexible metric nearest neighbor classification. Unpublished manuscript available by anonymous FTP from playfair.stanford.edu (see /pub/friedman/README).

(Hastie and Tibshirani, 1994) Hastie, T.J. and Tibshirani, R.J. 1994. Discriminant adaptive nearest neighbor classification. Unpublished manuscript available by anonymous FTP from playfair.stanford.edu as /pub/hastie/dann.ps.Z.

(John *et al.*, 1994) John, G. H.; Kohavi, R.; and Pfleger, K. 1994. Irrelevant features and the subset selection problem. In Cohen, W. and Hirsh, H., editors, *Proceedings of the Eleventh International Conference on Machine Learning*, Rutgers University, New Brunswick, NJ. Morgan Kaufmann. 121–129.

(Kira and Rendell, 1992) Kira, K. and Rendell, L. A. 1992. A practical approach to feature selection. In *Proceedings of the Ninth International Conference on Machine Learning*, Aberdeen, Scotland. Morgan Kaufmann. 249–256.

(Merz and Murphy, 1996) Merz, C. J. and Murphy, P. M. 1996. UCI repository of machine learning databases. [http://www.ics.uci.edu/ mlearn/MLRepository.html].

(Mohri and Tanaka, 1994) Mohri, T. and Tanaka, H. 1994. An optimal weighting criterion of case indexing for both numeric and symbolic attributes. In Aha, D. W., editor, *Case-Based Reasoning: Papers from the 1994 Workshop*. AAAI Press, Menlo Park, CA. Technical Report WS-94-01.

(MUC-5, 1994) *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. Morgan Kaufmann, San Mateo, CA.

(Schaffer, 1994) Schaffer, C. 1994. A conservation law for generalization performance. In Cohen, W. and Hirsh, H., editors, *Proceedings of the Eleventh International Conference on Machine Learning*, Rutgers University, New Brunswick, NJ. Morgan Kaufmann. 259–265.

(Skalak, 1992) Skalak, D. B. 1992. Representing cases as knowledge sources that apply local similarity metrics. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, Bloomington, IN. Lawrence Erlbaum Associates. 352–330.

(Stanfill and Waltz, 1986) Stanfill, C. and Waltz, D. 1986. Toward Memory-Based Reasoning. *Communications of the ACM* 29:1213–1228.

(Wettschereck *et al.*, 1997) Wettschereck, D.; Aha, D. W.; and Mohri, T. 1997. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review* Special Issue on Lazy Learning Algorithms.

(Zheng, 1993) Zheng, Z. 1993. A benchmark for classifer learning. Technical Report 474, Basser Department of Computer Science, The University of Sydney, N.S.W. Australia 2006.

(Zwitter and Soklic, 1988) Zwitter, M. and Soklic, M. 1988. Lymphography domain. [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Donated by I. Kononenko and B. Cestnik.

This article was processed using the LaTeX macro package with LLNCS style