The Utility of Feature Weighting in Nearest-Neighbor Algorithms

Ron Kohavi[1] and Pat Langley[2] and Yeogirl Yun[3]
ronnyk@sgi.com, langley@cs.stanford.edu, yygirl@cs.stanford.edu

[1] Silicon Graphics, Inc., 2011 N. Shoreline Blvd., Mountain View, CA 94043
[2] Robotics Laboratory, Stanford University, Stanford, CA 94305
[3] Electrical Engineering Dept., Stanford University, Stanford, CA 94305

Abstract. Nearest-neighbor algorithms are known to depend heavily on their distance metric. In this paper, we investigate the use of a weighted Euclidean metric in which the weight for each feature comes from a small set of options. We describe DIET, an algorithm that directs search through a space of discrete weights using cross-validation error as its evaluation function. Although a large set of possible weights can reduce the learner's bias, it can also lead to increased variance and overfitting. Our empirical study shows that, for many data sets, there is an advantage to weighting features, but that increasing the number of possible weights beyond two (zero and one) has very little benefit and sometimes degrades performance.

1 Introduction

In recent years, instance-based methods [5, 13, 1] have emerged as a promising approach to machine learning, with researchers reporting excellent results on many real-world induction tasks. The basic approach involves storing training cases and their associated classes in memory and then, when given a test instance, finding the training cases nearest to that test instance and using them to predict the class. One drawback of such methods is that, when combined with a naive distance metric that weights attributes equally, they can suffer from the "curse of dimensionality," in which the number of cases needed to maintain a given error rate grows rapidly with the number of features [2].

This observation has led some researchers to augment nearest-neighbor methods with techniques for determining distinct weights for each feature. For instance, Cost and Salzberg [3] use differences in probability distributions across classes to modify the distance metric for nominal attributes, whereas Daelemans, Gillis, and Durieux [4] and Wettschereck [13] use mutual information to compute coefficients on numeric attributes. Aha [1] reports an incremental scheme that alters feature weights depending on their distance and predicted class. All of these researchers report improvement over the simple version of nearest neighbor that gives attributes equal weight.

In this paper we explore a different approach to determining feature weights for nearest-neighbor classification. Rather than considering a continuous weight

space, we restrict weights to a small, finite set. This reduces representational power and hence increases bias, but it should also lower the variance and thus reduce chances of overfitting. We report on DIET, a system that incorporates a *wrapper* method [7] to search the weight space using cross-validation error from the nearest neighbor algorithm. Experimental studies suggest that, on many natural data sets, restricting the set of weights to only two alternatives—which is equivalent to feature subset selection—gives the best results. A decomposition of errors into bias and variance helps explain these results.

2 The DIET Algorithm

To implement this idea, we had to adapt the wrapper approach, originally developed for feature selection, to determine appropriate weights for use in nearest-neighbor classification. Along the way, we needed to make a number of design decisions about its operation.

For example, we selected best-first search, which always expands the best node that has not yet been expanded, to explore the discrete weight space. We also needed some evaluation function to direct the search. As in earlier work with wrappers, we used the estimated error of the induction algorithm, in this case nearest neighbor[4], as measured by ten-fold cross-validation over the training data.

Our discrete approach to selecting feature weights required us to specify the number of k nonzero weights being considered. This determined the set of weights: 0, $1/k$, $2/k$, ..., $(k-1)/k$, 1. For the starting point in the search, we decided to use the assignment closest to the middle weight and to use a zero weight when $k = 1$ (i.e., when zero and one were the only possible weights). Note that, when $k = 1$, our approach is equivalent to feature subset selection and the initial node ignores all attributes, thus predicting the most frequent class.

Heuristic search also requires operators for moving through the space. Here we adopted operators that replaced the current weight for a feature with either the next larger or smaller value in the allowed set, unless the maximum or minimum has been reached. Finally, we selected a halting criterion that stopped search when it encountered five consecutive nodes with no children having scores more than 0.1% better than their parent.

Recall that each node evaluation requires a separate cross-validation run of nearest neighbor, which appears expensive. However, because one can implement nearest neighbor in an incremental manner, we can get the cross-validation error by removing each instance in a fold, classifying it, reinserting it, and repeating for other folds. Also, given n features, there are at most $2 \cdot n$ branches at each step of the search, which should keep computational costs reasonable.

We will refer to the above collection of design decisions as the DIET algorithm, since sometimes they cause features to lose weight, sometimes to gain weight,

[4] Throughout our studies we fixed the number of neighbors used in classification to one, since our goal was to investigate feature weighting rather than this factor.

and sometimes to remain the same. Now that we have described our approach to feature weighting, we can consider some hypotheses about its behavior and some experiments designed to test them.

3 Hypotheses and Experimental Design

Naturally, we did not expect DIET to outperform other methods in all domains. For data sets that contain few or no irrelevant features, we would expect it to behave comparably to simple nearest neighbor, or slightly worse due to the increased size of the hypothesis space. For domains in which relevant features have equal importance, we would expect DIET with few weights to outperform DIET with many weights.

To test this hypothesis, we designed three artificial domains that involved five continuous features in the range $[0 - 1]$ and two classes, each incorporating a target concept based on two prototypical instances, one for each class. The prototypes were at $(0.25, 0.25, \ldots, 0.25)$ and at $(0.75, 0.75, \ldots, 0.75)$. The three domains relied on the same prototypes but used different feature weights. To determine the class for a randomly generated training or test instance from a uniform distribution, we found the prototype P nearest to that instance (as defined by the domains' weights) and assigned it P's class. Domain R assigned weights of 1 to all five features (making them all equally relevant), whereas domain I assigned the weight 1 to three features and 0 to the other two, making the latter irrelevant. Domain W assigned the weight 1 to two features, $\frac{1}{2}$ to two features, and 0 to the last feature. We predicted that DIET would outperform nearest neighbor on the latter two domains and that, for domain W, running it with two non-zero weights (0.5 and 1 besides 0) would outperform the variant with only one non-zero weight.

To ensure relevance, we also wanted to study DIET's behavior on natural domains. To determine candidates on which feature weighting would likely yield improvements, we inspected learning curves (which plot error against the number of training cases) for domains from the UCI repository [11] and identified six data sets (`Anneal`, `Chess`, `Segment`, `Soybean-Large`, `Vehicle`, and `LED24`) in which nearest neighbor did not reach asymptote early in the curve. This suggested there was room for improvement on these data sets. We selected points on the learning curve where nearest neighbor had not yet reached asymptote and used them to determine the number of cases in our training sets. We predicted that DIET would outperform nearest neighbor in these situations. We also selected five additional UCI domains—`Breast`, `Hypothyroid`, `Mushroom`, `Vote`, and `LED7`—that did not have the above characteristic. We did not expect better results than nearest neighbor in these cases.

For each domain, we randomly selected ten training sets and ten associated (and disjoint) test sets; DIET did not have access to these test sets when computing its cross-validation estimates. Table 1 shows the characteristics of the various domains. In order to test our basic hypotheses, for each training-test partition we also ran simple nearest neighbor, using equal weights for each feature. We

Table 1. A summary of the domains used in the experimental study. Group A consists of our artificial domains, group B contains data sets in which nearest neighbor had not reached asymptote, and group C contains data sets in which it appeared close to asymptotic performance.

Group	Data Set	Train Size	Test Size	No. Features	No. Classes
A	Relevants (R)	300	2700	5	2
	Irrelevants (I)	300	2700	5	2
	Weights (W)	300	2700	5	2
B	Anneal	100	798	38	6
	Chess	100	3096	36	2
	Segment	300	2010	19	7
	Soybean-Large	100	583	35	19
	Vehicle	200	646	18	4
	LED24	200	3000	24	10
C	Breast	100	599	10	2
	Hypothyroid	200	2963	25	2
	Mushroom	100	8024	22	2
	Vote	100	335	16	2
	LED7	200	3000	7	10

also ran Quinlan's C4.5 [12], a well-known decision-tree algorithm, and Aha's IB4 [1], an instance-based algorithm which incorporates a different weighting schemes that assigns weights to features incrementally. Our aim here was not to test specific hypotheses but simply to see how DIET fares compared to other well-known induction algorithms.

4 Experimental Results

Let us now consider the results of our comparative studies on the artificial domains described above, followed by those for the various UCI domains. In an effort to gain a deeper understanding of these results, we also analyze the bias-variance tradeoff for different sets of possible weights.

4.1 Experiments on Artificial Domains

The results from our runs on the three artificial domains appear in Figure 1, which shows the mean error for each combination of algorithm and domain. (The notation "DIET $W = k$" indicates our feature-weighting algorithm when given k non-zero weights.) As predicted, nearest neighbor does produce slightly better results than the variants on domain R; its assumption that all features

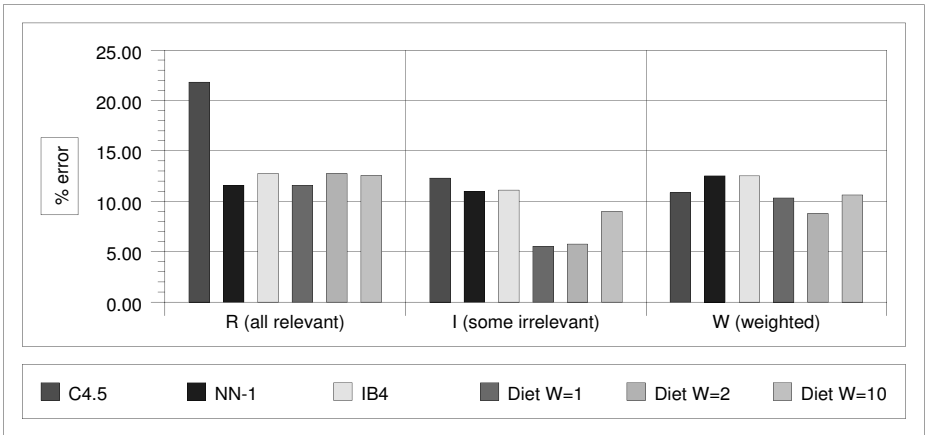

Fig. 1. Absolute error rates on the three artificial domains.

are equally relevant is perfect for the domain, whereas the other methods search a larger space and thus have increased variance.

However, the situation is quite different for domain I, in which two features are irrelevant. As expected, all versions of DIET outperform nearest neighbor here, but we also see that increasing the number of weights actually increases the error. Inspection of individual runs revealed that, when told to use a single non-zero weight, DIET found the correct weights on all ten runs, whereas versions that allowed more weights sometimes generated 0.5 and 1.0 in the same run, thus giving poorer performance. IB4's weighting heuristic also failed to find the optimal weights for this domain.

We anticipated that DIET with two non-zero weights would outperform both one non-zero weight and simple nearest neighbor on domain W, which we had generated using different weights. Indeed, this variant fares better than the others, but does not achieve the lowest possible error (7.44%), which we obtained by running nearest neighbor with the correct weights. Examination of weights found in individual runs showed the system sometimes found nonoptimal values.

Further experiments we have done show that in half the runs, the chosen weights had higher estimated accuracy than the estimated accuracy on the true weights. In cases where the estimated accuracy of the chosen weights is lower than the true weights, further search might have helped, as it is clear that the search was stuck in a local optimum. However, in those cases where the estimates were higher, the search was overfitting the weights to the data and increasing the amount of search would not have helped; the true weights would not be selected as other weights were already estimated to be superior.

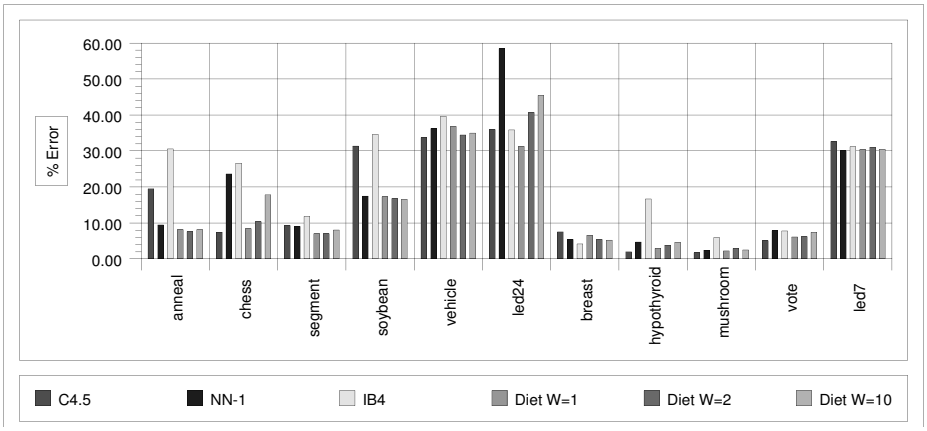

Fig. 2. Absolute error rates on the 11 domains from the UCI repository.

4.2 Experiments on UCI Data Sets

Figure 2 presents the experimental results, in terms of absolute errors, for the 11 domains from the UCI repository, whereas Figure 3 shows the analogous errors measured relative to those for simple nearest neighbor. The most obvious result is that DIET with one non-zero weight typically gives lower errors than nearest neighbor. In five domains (chess, segment, led24, hypothyroid, and vote) the differences were significant ($p < 0.05$), whereas in five others (anneal, led7, mushroom, soybean, vehicle) they were not. In only one domain (breast cancer) was the difference significant in favor of nearest neighbor. DIET usually performed substantially better than IB4, the other instance-based weighting method we ran for comparison. In some cases, our system also outperformed C4.5, even for some domains in which it fared better than nearest neighbor.

Also, note that DIET did better than nearest neighbor not only in the domains we predicted, where the latter had not yet reached asymptote, but also on some data sets where we thought it had: hypothyroid and vote. Apparently, nearest neighbor was still improving on these domains, but the large number of features (24 and 16) gave a very low learning rate, making it appear as if it had reached the asymptotic error level. The figure also reveals that increasing the number of non-zero weights above two rarely reduced classification error. We noted minor improvements when going from one to two non-zero weights on the `anneal`, `breast`, `soybean-large`, and `vehicle` domains, but using more weights led to large increases in error on other data sets. This effect is especially evident for the ten-weight version of DIET, which failed to significantly outperform the version with two non-zero weights on any domain.

Despite our efforts at efficient implementation, DIET's reliance on the wrapper approach makes it generally slow. Running times for a single training-test

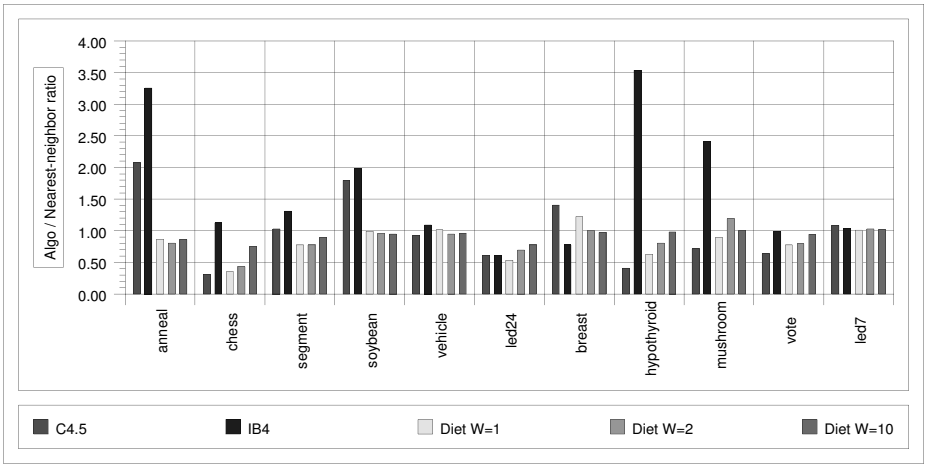

Fig. 3. Relative error rates for the data sets. Quantities indicate the error of each algorithm divided by that of nearest neighbor, with values less than one indicating improved performance over this method.

partition on an SGI Challenge varied from two minutes to about 30 minutes for DIET with ten non-zero weights, whereas simple nearest neighbor took less than 40 seconds. Speeding this process is an important direction for future research, but it seems even more essential to understand the reasons why simple feature selection fared so well relative to DIET with larger sets of weights.

4.3 Bias and Variance Decomposition

One way of understanding the behavior of algorithms is to decompose the error rates into their components. Given a target concept, the average error of an algorithm for different data sets of size m can be decomposed into a squared bias component and a variance component [6].

The squared **bias** measures how closely, for a random instance, the average prediction of the learning algorithm matches the target value for that instance. An algorithm that uses a small set of models (e.g., only two weights) might not be able to model the target appropriately due to lack of representational power. In such cases, the predictions will be biased. The **variance** term in the decomposition measures how much the learning algorithm's guess changes around its mean prediction as one varies the training set. If minor changes in training data produce changes in predictions, it will have a high variance (colloquially, it overfits the training set).

Within the nearest-neighbor framework, feature weighting can reduce the bias because the distances between features shrink and nearest-neighbor then better model local effects. If the feature weights selected do not vary much with

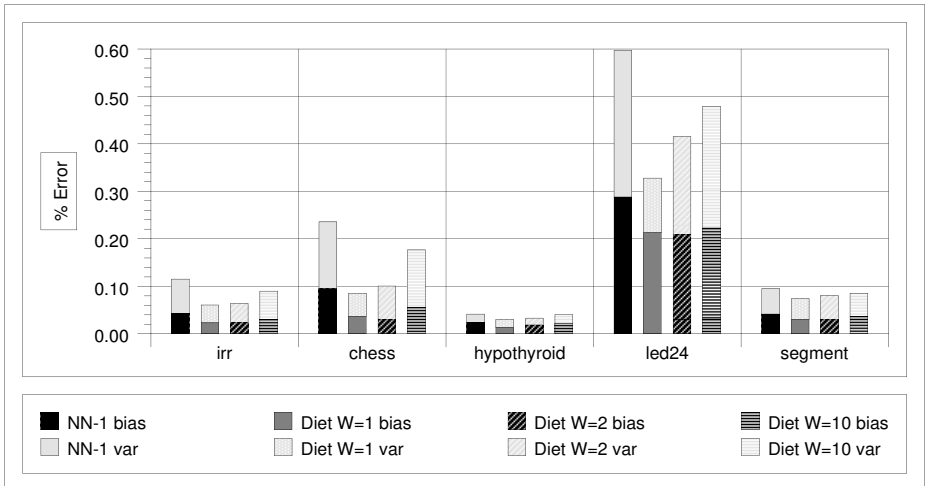

Fig. 4. The bias and variance decomposition of error.

changes in the training set, then the overall variance can be reduced as well, because the space is more densely populated when some weights are less than one.

For classification problems, researchers have recently proposed different decompositions of bias and variance. We used the decomposition proposed by Kohavi and Wolpert [9], which has the desirable property that it is equivalent to the standard squared error loss (except for a leading constant of one-half) if one views the label as a vector of zero-one indicator variables and computes squared error loss on the vector.

Figure 4 shows the bias-variance decomposition of error for selected data sets on which the error varied widely across different versions of DIET. We generated these results by taking 25 samples from a data set five times the original size of the training set, keeping the rest of the instances to compute the bias and variance.

As expected, the figure shows that feature weighting reduces the bias, especially in chess and led24, which are known to have many irrelevant features. Introducing weights also drastically lowers the variance over nearest neighbor, especially when there is only one non-zero weight. Extreme examples of this effect occur in the domains I, chess, and led24. As we allow more weights, the variance increases because the algorithm becomes unstable, due to the large number of weight settings that fit the training set well but do not match the true weights of the underlying distribution.

The decomposition shows that the extra power given by an increased set of weights (beyond one or two non-zero weights) did not further reduce the bias, while it usually increased the variance. For our setting, the overall utility of increasing the number above one or two non-zero weights was negative. Unless

there is good reason to believe that different weights can help (as in the artificial domain W), or unless the data sets are much larger, we recommend using only a few weights in conjunction with the wrapper approach or similar methods.

5 Summary

In this paper we described DIET, an algorithm that uses a simple wrapper approach to heuristically search through the set of weights used for nearest-neighbor classification. DIET typically gave lower errors than simple nearest neighbor, which weights features equally, and usually did better than Aha's IB4, which incorporates an incremental weighting method. However, our most interesting finding was that considering only a small set of weights typically gave better results than using a larger set. In fact, for the real-world data sets we examined, running DIET with one non-zero weight, which assumes each feature is either relevant or irrelevant, was difficult to outperform.

More detailed analysis suggested that restricting the set of weights can reduce the algorithm's variance and thus lower its error rate. The naive assumption that using more weights will reduce classification error is simply false: increasing the set of possible weights will increase the variance, and the bias-variance tradeoff will determine whether the overall performance improves. In our study, there were no domains in which using more than two non-zero weights significantly reduced the classification error. We should stress that our conclusions about the superiority of using one non-zero weight (feature subset selection) hold only in the context of our wrapper approach to feature weighting and for small training sets.

In future work, we should repeat our experiments with training sets of different sizes, to determine whether more weights prove useful at later points in the learning curve. We should also run comparative studies between DIET and other approaches to feature weighting. For example, Kelly and Davis [8] report using a genetic algorithm to search the space of feature weights, and Lowe [10] presents an alternative scheme that employs conjugate gradient descent through the weight space. Such comparisons will help determine the relative benefits of the two key ideas behind DIET: restricting the number of weights used in nearest-neighbor classification and using a wrapper method to search the space of such weights.

Acknowledgments

This research was supported in part by Grant No. N00014-94-1-0448 from the Office of Naval Research and Grant No. IRI-9116399 from the National Science Foundation, for development of the MLC++ software used in the experiments, and in part by Grant No. F49620-94-1-0118 from the Air Force Office of Scientific Research.

References

1. David W. Aha. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(1):267–287, 1992.
2. R. E. Bellman. *Adaptive Control Processes : A Guided Tour*. Princeton University Press, NJ, 1961.
3. Scott Cost and Steven Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10(1):57–78, 1993.
4. W. Daelemans, S. Gillis, and G. Durieux. The acquisition of stress: A data-oriented approach. *Computational Linguistics*, 20, 1994.
5. Belur V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1990.
6. Stuart Geman, Eli Bienenstock, and Reneé Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–48, 1992.
7. George John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Machine Learning: Proceedings of the Eleventh International Conference*, pages 121–129. Morgan Kaufmann, July 1994.
8. J. D Kelly and L. Davis. A hybrid genetic algorithm for classification. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 645–650. Morgan Kaufmann, 1991.
9. Ron Kohavi and David H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In Lorenza Saitta, editor, *Machine Learning: Proceedings of the Thirteenth International Conference*. Morgan Kaufmann, July 1996. Available at `http://robotics.stanford.edu/users/ronnyk`.
10. D. G. Lowe. Similarity metric learning for a variable-kernel classifier. *Neural Computation*, 7:72–85, 1995.
11. Christopher J. Merz and Patrick M. Murphy. UCI repository of machine learning databases. At `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1996.
12. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.
13. Dietrich Wettschereck. *A Study of Distance-Based Machine Learning Algorithms*. PhD thesis, Oregon State University, 1994.

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