Weighting Unusual Feature Types

Nicholas Howe and **Claire Cardie**
{nihowe, cardie} @cs.cornell.edu
Department of Computer Science
Cornell University

Abstract

Feature weighting is known empirically to improve classification accuracy for k-nearest neighbor classifiers in tasks with irrelevant features. Many feature weighting algorithms are designed to work with symbolic features, or numeric features, or both, but cannot be applied to problems with features that do not fit these categories. This paper presents a new k-nearest neighbor feature weighting algorithm that works with any kind of feature for which a distance function can be defined. Applied to an image classification task with unusual set-like features, the technique improves classification accuracy significantly. In tests on standard data sets from the UCI repository, the technique yields improvements comparable to weighting features by information gain.

Keywords: feature weighting, k-nearest neighbors, unusual feature types.

1 Introduction

Automatic classification is one of the most well-studied areas in machine learning. Researchers regularly introduce new supervised learning algorithms in an attempt to improve accuracy on specialized tasks. Yet one of the oldest classification algorithms is still successfully used in many applications: the k-nearest-neighbors algorithm. Work continues on refinements and variations of the basic algorithm. For example, Wilson and Martinez recently introduced several new distance functions that increase classification accuracy on a range of data sets from the UCI machine learning repository (Merz and Murphy, 1996).

K-nearest-neighbors (k-NN) belongs to the family of case-based or instance-based learning algorithms. Such algorithms typically store instances seen during training in a case base, and then use information in the stored cases to predict the class of previously unseen test cases. K-NN, for example, retrieves the k training cases that are most similar to the instance to be classified. The most common class among the k retrieved cases is taken as the class of the new instance, with ties broken arbitrarily or by voting. The major difference between various flavors of k-NN algorithms is in the distance function that is used to determine the k most similar training cases.

Typically, cases to be classified are specified as a collection of attribute-value pairs, and the distance function is some simple combination of the distances between the individual features. The Euclidean distance is a typical example of such an arrangement, using a sum of squares to combine the difference of each of the coordinates. Distance functions of this form may be divided conceptually into two components: the *intrinsic distance function* applied to each feature, and the *distance combination function*, which takes the differences for individual features and combines them into a single overall distance between the cases. Often the latter includes some type of feature weighting, where some features contribute more towards the final combined distance. In fact, the feature weighting is generally the most important part of the combination process, which otherwise usually consists of taking a sum or Euclidean norm. Feature weighting also includes feature selection, which may be viewed as an extreme form of weighting in which some features are ignored entirely.

Making a distinction between intrinsic distance functions and the method of combination allows research on each to proceed separately. Some researchers have proposed distance functions that include both components (e.g., the value difference metric, or VDM (Stanfill and Waltz, 1986)), but these have often been separated again by later researchers (e.g., the modified value difference metric, or MVDM (Cost and Salzberg, 1993)). Occasionally, researchers conflate the two, perhaps unnecessarily. For example, Wilson and Martinez engage in a series of experiments to ensure that the intrinsic distance functions they use for discrete and continuous attributes do not unfairly weight one type of feature over the other. This task more properly belongs to the combination stage of distance calculation. If standard combination methods existed to counteract any unintended bias introduced by the use of diverse intrinsic distance functions, then the necessity for tedious calibration like that performed by Wilson and Martinez would be greatly reduced. One of the goals of this paper is to propose one such standard method.

A second issue that can be addressed by the proper choice of distance combination function is how to weight features for their importance in classification. Some features are highly indicative of an instance's class and should be weighted heavily during combination. Conversely, some features may be irrelevant and should receive little or no weight. Ideally, the combination function should automatically determine optimal feature weights regardless of the number or type of features involved.

This paper presents a distance combination function that addresses both of the above concerns. First, it automatically calibrates individual features to correct for any inadvertent bias introduced by the intrinsic distance functions chosen. Second, it applies automatic feature weighting to the calibrated features to emphasize features useful in classification and minimize the impact of irrelevant features. Previous approaches to this problem often use combination methods that are inseparable from the intrinsic distance functions, or that assume a particular type of feature. Our method, called *mean difference weighting (MDW),* is universal in that it may be applied regardless of the types of the features involved and regardless of the choice of intrinsic distance function. It is automatic in that it has no parameters that need be tuned to each particular application. Furthermore, empirical tests show that it raises classification accuracy across a wide range of data sets when compared to an unweighted distance function. MDW is comparable in terms of performance to weighting features by their information gain, except that it is applicable to a wider selection of data sets. The specifics of the algorithm are described in the next section. In Section 3 we present the results of tests comparing MDW to unweighted classification and to various existing weighting methods. The last two sections discuss related work, and areas where more research on feature weighting is needed.

2 Mean Difference Weighting Algorithm

The mean difference weighting (MDW) algorithm is designed to provide a universal approach to feature weighting for classification tasks. It places minimal restrictions on the type of the features, and is compatible with most intrinsic distance functions. The algorithm adjusts for numeric biases either present in the data or introduced by the intrinsic distance functions used, while simultaneously assigning greater weight to features that are most useful for categorization. It produces different weight vectors for different regions of the instance space. By default, the regions used are simply the individual classes to be predicted, but in principle other groupings can be used if information is available to motivate doing so. Detailed investigations of similar types of class-based feature weighting have been performed by Howe and Cardie (1997).

The MDW algorithm proceeds in two stages, *scaling* and *significance weighting*. Each stage produces a weight vector; the final weight vector is the componentwise product of the two. Intuitively, the scaling stage equalizes the features so that each contributes equitably to the difference metric. This step removes biases due to differences in feature type (e.g., continuous vs. discrete), choice of feature measurement units (e.g., feet vs. cm), and choice of intrinsic distance function. In turn, the significance weighting stage adjusts the weight of each feature according to its calculated importance. This step reduces or eliminates the effect of irrelevant features and increases the effect of important ones.

Both stages of the algorithm rely on estimating the expected value returned by the intrinsic difference function δ_{f_i} when comparing a particular feature f_i of cases randomly chosen from two specified sets, S_1 and S_2. (These sets may be, for example, the sets of instances belonging to two different classes.) The expected differences $\overline{\Delta_{f_i}}$ are estimated by observing the mean difference on that feature between all pairs of cases in the two sets intersected with the training set T. Let

$$\overline{\Delta_{f_i}}(S_1, S_2) \doteq E[\delta_{f_i}(s_1 \in S_1, s_2 \in S_2)] = \frac{1}{|S_1|\,|S_2|} \sum_{s_1 \in S_1} \sum_{s_2 \in S_2} \delta_{f_i}(s_1, s_2). \tag{1}$$

Then

$$\overline{\Delta_{f_i}}(S_1, S_2) \approx \overline{\Delta_{f_i}}(S_1 \cap T, S_2 \cap T). \tag{2}$$

The value computed by this procedure will be an accurate reflection of the actual expectation value across the entire instance space insofar as the training set accurately reflects the entire set of possible cases.

The Scaling Algorithm The scaling algorithm first estimates the expected difference between feature values for two randomly chosen cases. As described above, this is taken to be the mean difference between two cases selected at random from the training set, $\overline{\Delta_{f_i}(T,T)}$. The scale weight for a feature is simply the inverse of this mean feature difference. This weighting scheme ensures that each weighted feature has an (estimated) expected difference of 1.

The scale weights can be naively computed as a simple $O(n^2)$ loop in the size of the training set if n is not too large. Although the algorithm can be executed off-line, there still may be applications where the size of n makes the naive algorithm prohibitive. In this case, a randomized approximation technique can give an accurate estimate of the mean with high probability as long as some upper bound on the feature difference is available[1]. For standard symbolic and numeric features, such bounds are easy to obtain over the test set. There also exists an exact $O(n \log n)$ algorithm to determine the mean difference for numeric features, and an exact $O(n + m^2)$ algorithm for symbolic features, where m is the number of discrete values the feature can take on. All the results reported in this paper are based on exact calculations.

Significance Weighting Whereas the scaling stage computes a single scale weight vector for the entire instance space, the significance weighting stage computes different weight vectors in different regions. Specifically, there will be a distinct weight vector associated with each class of instance in the training set. Vectors for sub-class groupings may be computed instead, if information is available to motivate such groupings. We chose to focus only on grouping by classes. During classification, when a test instance is compared to a training instance, the weight vector associated with the class of that training instance is used.

The raw significance weights are computed by comparing the expected feature difference between objects in a class C_j (called the *mean in-class difference*, $\Delta_{in}(f_i, C_j)$) with the expected difference between an object in the class and another not in the class (called *mean out-of-class difference*, $\Delta_{out}(f_i, C_j)$).

$$\Delta_{in}(f_i, C_j) \doteq \overline{\Delta_{f_i}}(T_{C_j}, T_{C_j}) \tag{3}$$

$$\Delta_{out}(f_i, C_j) \doteq \overline{\Delta_{f_i}}(T_{C_j}, T - T_{C_j}) \tag{4}$$

If the mean in-class difference is larger than the mean out-of class difference, then

[1]This result can be obtained through application of Chernoff bounds.

the feature is assigned a raw significance weight of zero.

$$\vec{W}_{RawSig} = \begin{cases} \Delta_{out}(f_i, C_j) - \Delta_{in}(f_i, C_j) & \text{if } \Delta_{out}(f_i, C_j) > \Delta_{in}(f_i, C_j) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

To avoid bias due to significance vectors of different sizes, the raw significance weight vector is normalized so that its components sum to 1. The final weight vector for each class is the componentwise product of the scaling and significance weight vectors. The weight vectors for each class can be precomputed and stored, making their use during classification extremely fast. Furthermore, if additional information is stored, such as the number of training cases seen already in each class, then the weights can be incrementally updated at small cost as new training examples are added.

Note that the MDW algorithm may be applied regardless of the underlying feature type, and the intrinsic distance functions used need not be the same for each feature. MDW only requires that each intrinsic distance function returns a number, and is independent of the values of the other features.

Although theory shows that no learning algorithm can claim a priori to generalize better than another (Schaffer, 1994), under reasonable assumptions one would expect that MDW would in fact improve classification accuracy over another algorithm that omitted either the scaling or significance weighting steps. That is, if one makes the critical assumption that the average differences calculated from the training set are reasonably close to the actual values across the entire instance space, then MDW will assign highest weight to those features that are most useful for classification. This should in fact lead to higher accuracy. The next section verifies this prediction with empirical studies.

3 Empirical Results

A major motivation behind the development of MDW was the need for weighting algorithms that can be applied to problems with unusual feature types. Therefore, we focus our testing initially on a domain where many standard feature weighting algorithms are not easily applicable. An image classification task was chosen for this purpose. We also test MDW on a few benchmarks from the UCI repository, comparing it to standard k-NN and to a method that weights each feature by the information gain from knowing its value (Daelemans *et al.*, 1994).

The Image Classification Task The image classification task selected is described in detail in Howe (1998). A feature extraction algorithm was run on a set of 1011 images distributed among twelve classes, producing a feature description for each image. The features describing an image are composed of 36 numeric features and 8 "set" features. Each set feature contains one or more members, where the members are vectors of 11 numeric sub-features. The difference between two sets is defined to be the minimum of the distances between any pair of members representing each set:

$$\delta(S_1, S_2) = \min_{v_1 \in S_1, v_2 \in S_2} \delta(v_1, v_2) \tag{6}$$

Here $\delta(v_1, v_2)$ is the ordinary vector difference between the two sub-feature vectors.

Nearest neighbor classification using the feature set described above is highly successful, identifying 73.7% of the images correctly in a leave-one-out cross validation test. This is significantly better than other published results using the same image set, including standard computer vision methods like color histograms. Yet it was desired to further improve the accuracy by applying appropriate feature weights. Because of the set features, it is unclear how to apply information gain to this problem, or many other common weighting algorithms for that matter. However, MDW is easily applied, producing significant improvements in accuracy. With mean difference weighting, classification accuracy rises to 80.0%. This difference is highly significant ($p \ll .01$).

UCI Data Sets In order to assess MDW relative to other weighting algorithms, a series of tests were also run on eight domains taken from the UCI repository (Merz and Murphy, 1996). As baselines, we compared MDW to a strict k-NN algorithm and an algorithm that weights features by their information gain (IG). Where necessary for the latter, continuous features were converted into eight discrete bins of equal size. For all experiments, the best value of k was learned by leave-one-out cross validation on the training set. Runs used the training and test set specified in the repository documentation if available, or 10-fold cross validation otherwise. Results for these runs are summarized in Table 1. In broad terms, MDW outperforms ordinary k-NN, and performs as well as information gain for seven out of the eight tasks.

The first four data sets listed, *LED-7*, *LED-24*, *Waveform-21*, and *Waveform-40* are synthetic tasks meant to test performance on irrelevant features. Each pair represents data sets with and without irrelevant features. The LED experiments show that feature weighting methods can reliably select the important features for this task. The waveform experiments fail to show this. Upon closer examination of the results, it appears that k-NN is able to compensate for the irrelevant features by selecting a large enough value of k. Because k is tuned individually for each fold, the accuracy of k-NN does not suffer in comparison to the other methods.

The next three data sets are taken from naturally occurring tasks, where feature relevance is not known *a priori*. Feature importance probably varies between each feature, and may depend on the particular value of the feature. MDW's accuracy is again comparable to information gain weighting for all three tasks, and is significantly better than k-NN on a majority.

The final data set listed, *Ionosphere*, was included because it is the one example we came across during testing on which MDW's performance was uncharacteristically poor. Many classification algorithms perform poorly on one or more isolated data sets, so we wanted to understand the specific reasons behind this failure. Closer examination of the weights generated for this task showed that for one of the two classes, nearly all the weights were set to zero. Other data sets didn't produce such an imbalance in the number of features considered relevant for different classes. If a similar pattern shows up in other data sets where MDW's performance is poor, then this pattern may be a warning sign that MDW is inappropriate for a particular task.

Table 1: Accuracy of MDW Compared to k-NN and Information Gain

Data Set	k-NN	IG	MDW
Images	73.7	–	**80.0**
LED-7	**66.8**	**67.6**	**68.0**
LED-24	57.2	**78.4**	**74.4**
Waveform-21	**82.8**	74.0	**78.4**
Waveform-40	**82.4**	76.4	**78.4**
Soybean	86.4	**90.4**	**90.4**
Lymphography	**79.3**	**81.4**	**78.6**
Promoters	79.0	**87.0**	**87.0**
Ionosphere	**96.0**	**96.0**	80.1

Key: **Bold face type** identifies the method with the highest accuracy and those not statistically distinguishable from it at the 95% confidence level.

We also compared MDW with the value difference metric, because this is another popular weighting technique. Properly speaking, VDM combines an intrinsic feature difference function with a feature weighting algorithm. When the weights are not used it is sometimes called MVDM (Cost and Salzberg, 1993). Since MDW may be applied to any intrinsic feature difference function, it may be applied on top of the unweighted value difference metric. Our tests compared MVDM, VDM, and MDW+MVDM, i.e., MDW weights applied on top of MVDM. Table 2 shows accuracies on the same tasks for these techniques. The results are statistically indistinguishable in six of the eight data sets, but in two cases, the MDW+MVDM combination did not do as well. This may indicate some subtle incompatibility between the two methods, perhaps due to the fact that MDW computes a different set of weights for each class while VDM assumes all classes are treated equally. Interestingly, the feature weighting added for VDM also does not appear to help significantly.

Table 2: Comparison of MVDM, VDM, and MDW+MVDM

Data Set	MVDM	VDM	MDW+MVDM
LED-7	**64.4**	**66.0**	**68.0**
LED-24	**74.0**	**75.6**	**74.4**
Waveform-21	**79.6**	**80.8**	**79.6**
Waveform-40	**80.0**	**80.4**	**77.20**
Soybean	**92.3**	**92.0**	**93.4**
Lymphography	**83.6**	**84.3**	80.7
Promoters	**93.0**	**94.0**	89.0
Ionosphere	**95.4**	**95.4**	**92.7**

Key: **Bold face type** identifies the method with the highest accuracy and those not statistically distinguishable from it at the 95% confidence level.

4 Related Work

Several other approaches to feature weighting have already been mentioned, such as the value difference metric (Stanfill and Waltz, 1986) and information gain weighting (Daelemans *et al.*, 1994). Many variants of VDM have been proposed, among the most recent the heterogeneous, interpolated, and windowed value difference metrics (Wilson and Martinez, 1997). Numerous other approaches to feature weighting and feature selection have also been tried. Standard feature selection algorithms include both forward and backward sequential selection, which progressively add or subtract features from the set used in classification, until accuracy peaks. These approaches have the advantage that they are sensitive to interactions between features, but on the other hand they can be fooled by local maxima. Researchers have addressed this weakness by using different methods for feature selection, including wrappers (John *et al.*, 1994) and genetic algorithms (Skalak, 1994). Others have examined local feature selection, using different feature sets in different regions of the instance space (Domingos, 1997). Most of this work concentrates on homogeneous data sets with a single type of feature.

Recent work in local feature weighting includes DANN (Hastie and Tibshirani, 1994), the flexible metrics of Friedman (1994), and methods that learn local feature weights incrementally (Ricci and Avesani, 1995; Bonzano *et al.*, 1997) For a survey of local feature weighting algorithms used in regression, see Atkeson et. al. (1997). A smaller number of algorithms use weights that vary by class, as MDW does. These include PCF (Creecy *et al.*, 1992) and CDW (Howe and Cardie, 1997), which operate only on discrete features, and IB4 (Aha, 1992), which learns feature weights incrementally. Most, if not all, of these algorithms have been shown to perform well on a variety of learning tasks.

In comparison with other algorithms, two properties of MDW stand out. The first is its ability to deal with a wide variety of feature types. Most weighting techniques can deal with both symbolic features and numeric features if discretization is applied. But the majority cannot handle the set-like features described as part of the image data set. Second, any method that uses discretization must make decisions about where to divide the continuous features and how many divisions to make. These decisions are not extremely difficult, but may require some tuning and testing to implement. In light of this, MDW is attractive for having no parameters that must be set. It also automates the scaling of features if features of different types are to be included.

5 Conclusions

Theoretical considerations indicate that no supervised classification algorithm performs better than any other in all situations. Even using cross validation to choose the best method provides no guarantees of optimality (Schaffer, 1994). Thus for practical applications it is important to choose an approach that has been shown empirically to have reasonable accuracy on typical data sets, such as MDW. MDW also has the advantages that it requires no parameter setting and is designed to work on a wide range of data sets.

The main drawback of the MDW algorithm is its $O(n^2)$ naive training time. Although approximation techniques and faster algorithms will work in most cases, they make it somewhat more difficult to apply MDW as a simple black box. Fortunately, the algorithm need only be executed once offline, and information may be retained to allow inexpensive incremental update if new training instances are to be encountered. Thus, except for the largest of data sets, the training time is not likely to be prohibitive even for the naive algorithm.

MDW, like many weighting schemes, also fails to account for redundancy in the original feature set. Thus, if a given feature is simply duplicated across the instance space, that feature will receive twice as much weight (counting both of its guises together) in spite of MDW's efforts to weight features equally. This effect may be countered in some domains by performing a principal components decomposition before applying MDW. Unfortunately, this solution will not work in all domains because the principle components decomposition is designed for numeric features and is not feasible for all feature types.

We are currently considering new algorithms inspired by MDW to address the issues raised above. These range from variations in the weighting algorithm to more comprehensive modifications of the method by which input from different features is combined to produce a classification. The approach used here may also prove fruitful in weighting features for other tasks, such as information retrieval. The challenge in applying the method to such tasks is to provide the algorithm with supervisory information in the absence of explicit classes. In this case we will need to make use of MDW's capacity to work with any meaningful division of the training case into subsets, rather than just subsets based on instance class.

Ultimately, given that there are already many feature weighting schemes to choose from, MDW will probably remain most useful in situations like the image data set we have described. Domains in natural language processing and computer vision naturally promote the use of complex features such as sets, lists, bit strings, and even trees. Because of the generality of MDW, it can easily be applied to such new features, regardless of what intrinsic distance function is chosen. In such situations, it offers several advantages. It requires no domain-specific knowledge to implement, and has no extra parameters that need setting and tuning. Most importantly, it can be applied to diverse feature types with confidence that it will account for their individual differences appropriately. In tasks where the features are not all simply symbolic or numeric, mean difference weighting can be a useful technique to apply.

6 Acknowledgements

This work was supported in part by NSF CAREER Award IRI–9624639 and NSF Grant GER–9454149. We thank M. Zwitter and M. Soklic for making the lymphography data available.

References

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

(Bonzano *et al.*, 1997) Bonzano, A.; Cunningham, C.; and Smyth, B. 1997. Using introspective learning to improve retrieval in case based reasoning: A case study in air traffic control. In Leake, D. B. and Plaza, E., editors, *Second International Conference on Case Based Reasoning*. Springer.

(Cost and Salzberg, 1993) Cost, S. and Salzberg, S. 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning* 10:57–78.

(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.

(Daelemans *et al.*, 1994) Daelemans, W.; Durieux, G.; and Gillis, S. 1994. The Acquisition of Stress: A Data-Oriented Approach. *Computational Linguistics* 20(3):421–451.

(Domingos, 1997) Domingos, P. 1997. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review* 11:227–253.

(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.

(Howe and Cardie, 1997) Howe, N. R. and Cardie, C. 1997. Examining locally varying weights for nearest neighbor algorithms. In Leake, D. B. and Plaza, E., editors, *Second International Conference on Case-Based Reasoning*. Springer.

(Howe, 1998) Howe, N. R. 1998. Percentile blobs for image similarity. Technical report, Cornell University. (Submitted for publication).

(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.

(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].

(Ricci and Avesani, 1995) Ricci, F. and Avesani, P. 1995. Learning a local similarity metric for case-based reasoning. In *First International Conference on Case-Based Reasoning*.

(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, 1994) Skalak, D. B. 1994. Prototype and feature selection by sampling and random mutation hill-climbing algorithms. In Cohen, W. and Hirsh, H., editors, *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, New Jersey. 293–301.

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

(Wilson and Martinez, 1997) Wilson, D. R. and Martinez, T. R. 1997. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research* 6(1):1–34.