Artificial Intelligence Review **11:** 273–314, 1997.

A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms

DIETRICH WETTSCHERECK[1], DAVID W. AHA[2] and TAKAO MOHRI[3]
[1] *GMD (German National Research Center for Information Technology), Schloß Birlinghoven, 53754 Sankt Augustin, Germany*
E-mail: dietrich.wettschereck@gmd.de
[2] *Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory, Washington, DC USA*
E-mail: aha@aic.nrl.navy.mil
[3] *Hidehiko Tanaka Lab. Department of Electric Engineering, The University of Tokyo, 7-3-1 Hongo Bunkyo-ku, Tokyo 113 JAPAN*
E-mail: mohri@mtl.t.u-tokyo.ac.jp

Abstract. Many lazy learning algorithms are derivatives of the k-nearest neighbor (k-NN) classifier, which uses a distance function to generate predictions from stored instances. Several studies have shown that k-NN's performance is highly sensitive to the definition of its distance function. Many k-NN variants have been proposed to reduce this sensitivity by parameterizing the distance function with feature weights. However, these variants have not been categorized nor empirically compared. This paper reviews a class of weight-setting methods for lazy learning algorithms. We introduce a framework for distinguishing these methods and empirically compare them. We observed four trends from our experiments and conducted further studies to highlight them. Our results suggest that methods which use performance feedback to assign weight settings demonstrated three advantages over other methods: they require less pre-processing, perform better in the presence of interacting features, and generally require less training data to learn good settings. We also found that continuous weighting methods tend to outperform feature selection algorithms for tasks where some features are useful but less important than others.

Key words: lazy learning, k-nearest neighbor, feature weights, comparison

1. Introduction

The k-nearest neighbor (k-NN) classifier (Dasarathy 1991) is the basis of many *lazy learning* algorithms. k-NN is purely lazy; it simply stores the entire training set and postpones all effort towards inductive generalization until classification time. k-NN generalizes by retrieving the k least distant (i.e., most similar) instances of a given query and predicting their weighted-majority class as the query's class. The quality of k-NN's generalization therefore depends on which instances are deemed least distant, which is determined by its distance function.

In Section 2, we argue that k-NN's distance function is biased: it allows redundant, irrelevant, interacting, or noisy features to have as much effect on distance computations as other features. k-NN can perform poorly when such features are present. This observation motivated the creation of many k-NN variants that compute feature weights, which we review in Section 3. Several of these variants improve its accuracy on some learning tasks (e.g., Kelly and Davis 1991; Aha 1992; Wettschereck 1994). However, many empirical evaluations frequently compare sophisticated algorithms with standard k-NN (e.g., Gorman and Sejnowski 1988; Kohonen et al. 1988; Weiss and Kapouleas 1989; Bounds et al. 1990; Yau and Manry 1991; Bottou and Vapnik 1992; Wettschereck and Dietterich 1992; Michie et al. 1994). While k-NN is frequently shown to have lower classification performance, its own more sophisticated variants are typically ignored.

This paper defines and relates k-NN to the large family of lazy learning algorithms in Section 2, and then introduces a framework and reviews a subclass of k-NN weight learning methods in Section 3. Our objective is to bring attention to these weight learning variants, and their relative merits. No weight learning method can learn optimal weight settings for all learning tasks since each task requires a different learning bias for optimal performance (Mitchell 1990). Therefore, we empirically evaluate a subclass of weight learning methods, present general trends that contrast their capabilities, and investigate these trends in Section 4. We found that weight learning methods which incorporate performance feedback from the classifier showed several advantages, although some weighting methods can locate good weight settings without such feedback. Section 5 discusses the implications of our results in the context of the framework described in Section 3, while Section 6 addresses related work.

2. Context

The focus of our paper is necessarily constrained to a small class of lazy learning algorithms due to lack of space. We examine this constraint by defining the lazy learning paradigm (Section 2.1), by defining a subclass of k-NN classifiers (Section 2.2), and by explaining how this subclass relates to this paradigm. Our subsequent review and empirical study in Sections 3 and 4 examine variants of lazy learners in only this small subclass.

2.1. *Lazy Learning Algorithms*

Purely lazy learning algorithms are characterized by three behaviors:

1. *Defer*: They store all training data and defer processing until queries are given that require replies.
2. *Reply*: Queries are answered by combining the training data, typically by using a *local learning* approach (Bottou and Vapnik 1992) in which (1) instances are defined as points in a space, (2) a similarity function is defined on all pairs of these instances, and (3) a prediction function defines an answer to be a monotonic function of query similarity.
3. *Flush*: After replying to a query, the answer and any intermediate results are discarded.

This definition is purposely vague; it does not define how training instances are stored or represented, nor how they are combined during querying, nor even how similarity is defined. Nonetheless, the k-NN classifier is obviously a "pure" lazy algorithm.

Purity can be compromised in many ways. For example, some training data can be discarded or permanently combined, such as by averaging into prototypes. Many forms of pre-processing can be performed, often to enhance efficiency for incremental learning tasks (e.g., normalizing continuous data, caching similarity or prediction function parameter settings, discretizing continuous data, feature construction). Performance feedback might be recorded and used to guide decision making during query-answering, and intermediate results can be profitably cached for some tasks.

We focus on impure lazy learners in which settings for feature weights are cached (i.e., rather than dynamically computed at query time) and, in some cases, updated via performance feedback information. We also constrain our study in two respects. First, we assume that instances are represented by a set of feature-value pairs rather than, for example, directed acyclic graphs, which are sometimes used in the case-based reasoning literature (Kolodner 1993). Second, we focus on classification as the performance task and ignore issues concerning function learning (Atkeson et al. 1996a), class density estimation (e.g., using parzen windows (Duda and Hart 1973)), or problem solving (Kolodner 1993). Furthermore, we assume classes are disjoint.

2.2. *The k-NN Classifier*

A *classifier* inputs a query instance $\mathbf{q}$ and outputs a prediction for its class. Each *instance* $\mathbf{x} = \{x_1, x_2, \ldots, x_{|\mathbf{F}|}\}$ is a point in a multidimensional space defined by a feature set $\mathbf{F}$ whose class, x_c, is a member of a set of classes $\mathbf{J}$. Each feature is either continuous or ranges over a fixed set of discrete values.

A classifier's performance objective is to minimize *expected loss*, or *misclassification risk*, for each class $c_j \in \mathbf{J}$:

$$R_j = \sum_{c'_j \in \mathbf{J}} L_{c_j c_{j'}} p(c_{j'}|\mathbf{q}) \tag{1}$$

where $L_{c_j c_{j'}}$ is the loss (cost) associated with mistakenly classifying an instance of class c_j as in class $c_{j'}$ ($j \neq j'$), and $p(c_{j'}|\mathbf{q})$ is the probability of classifying $\mathbf{q}$ in class $c_{j'}$. k-NN generally assumes that all misclassifications have equal cost:

$$L_{c_j c_{j'}} = \begin{cases} 0 & j = j' \\ 1 & j \neq j' \end{cases} \tag{2}$$

Since k-NN is not given $\mathbf{q}$'s class, it instead outputs the most probable class:

$$k\text{-NN}(\mathbf{q}) = \max_{c_j \in \mathbf{J}} p(c_j|\mathbf{q}) \tag{3}$$

k-NN differs from other classifiers in how it defines these posterior class probabilities:

$$p(c_j|\mathbf{q}) = \frac{\sum_{\mathbf{x} \in \mathbf{K_q}} 1(x_c = c_j) \cdot \mathrm{K}(d(\mathbf{x},\mathbf{q}))}{\sum_{\mathbf{x} \in \mathbf{K_q}} \mathrm{K}(d(\mathbf{x},\mathbf{q}))} \tag{4}$$

where $1(\ )$ yields 1 iff its argument is true, $\mathrm{K}(\ )$ is a *kernel* function, here defined as

$$\mathrm{K}(d(\mathbf{x},\mathbf{q})) = \frac{1}{d(\mathbf{x},\mathbf{q})}, \tag{5}$$

and where $\mathbf{K_q}$ is the set of $\mathbf{q}$'s k-nearest neighbors among a set $\mathbf{X}$ of (previously supplied) *training* instances as determined by the distance function $d(\ )$. That is, k-NN computes the distance $d(\mathbf{x},\mathbf{q})$ of $\mathbf{q}$ to each $\mathbf{x} \in \mathbf{X}$ using:

$$d(\mathbf{x},\mathbf{q}) = \left(\sum_{f \in \mathbf{F}} w(f) \cdot \delta(x_f, q_f)^r\right)^{\frac{1}{r}} \tag{6}$$

where k-NN defines $r = 2$ (i.e., Euclidean distance), function $\delta(\ )$ defines how values of a given feature differ:

$$\delta(x_f, q_f) = \begin{cases} |x_f - q_f| & f \text{ is continuous} \\ 0 & f \text{ is discrete and } x_f = q_f \\ 1 & f \text{ is discrete and } x_f \neq q_f \end{cases} \tag{7}$$

and $w(f)$ defines the *feature weighting* function. k-NN defines this as a constant function:

$$w(f) = w_f \tag{8}$$

Finally, k-NN defines

$$\forall_{f \in \mathbf{F}} \quad w_f = s \tag{9}$$

for some scalar constant s. Equation 9's strong constraint provides the motivation for our survey and empirical study.

Some implementation details need mentioning. If there is a tie among the maximal $p(c_j|\mathbf{q})$, then one of the most probable classes is randomly selected. k is set using leave-one-out cross-validation on $\mathbf{X}$ (Weiss and Kulikowski 1991), where ties are broken in favor of smaller values for k. We used a standard function (i.e., subtract the minimum and divide by the observed range) to normalize all continuous values. This ensures that the range of $\delta(\ )$ is [0, 1] for all features. Thus, they have equal maximum and minimum potential effects on distance computations. Unfortunately, this also means that each redundant, irrelevant, and noisy feature has as much potential impact on k-NN's distance function as does any other feature.

2.3. *Scope of our Review*

Arguably, k-NN's description in Section 2.2 is needlessly complex. We had two reasons for showing this detail for what appears, at first, to be a simple classifier. First, this description demonstrates that k-NN is, in fact, but one algorithm in the paradigm of lazy learners. Second, we want to clarify that k-NN's typical description as a *non*-parametric (except for k) classifier is misleading; it simply eliminates many parameters by incorporating them as design decisions.

For example, k-NN's constant loss function is not *class sensitive*; its value for $L_{c_j c_{j'}}$ is invariant for different pairs of classes. Cost-sensitive learning (Turney 1995) variants of k-NN exist (e.g., Tan 1993) that minimize a locally weighted error criterion (Vapnik 1992). The loss function can be arbitrarily complex (e.g., it could vary per class or instance).

Second, k-NN's equation for computing posteriors (Equation 4) is a form of *kernel regression* in statistics (Nadaraya 1964), or the *probability choice model* in cognitive psychology (Luce 1963). It could take on many other forms, such as by replacing $1(\ )$ with a function that relates classes or varies on each instance. Also, many different kernel functions (K( )) have been investigated (Atkeson et al. 1996a).

Third, while we examine various distance functions, we only examine the effects of replacing Equation 9 with weighting functions that assign unequal weights to features. Many other types of distance functions and weighting methods have been studied. For example, rather than Euclidean or even Minkowskian, distance could be defined using a set-theoretic definition (Tversky 1977; Biberman 1994) or by a function other than one which sums independent contributions of the features. A different distance function could be applied at each query (*query-based*) (Atkeson et al. 1996a) or each instance (*point-based*) (Aha and Goldstone 1992). Similarly, the definition of feature difference (Equation 7) is one of many; functions other than absolute difference for continuous values and more elaborate functions for defining similarity on discrete features have been proposed (Stanfill and Waltz 1986).

Finally, many other classes of weight-learning methods have been examined, frequently in the context of statistical regression, where distance functions on continuous functions have been carefully examined. For example, we restrict our survey to algorithms that use a diagonal weight matrix where there is one weight per feature and no interaction between features. This assumes that the target concept can be best modelled by stretching and shrinking the instance space along its axes. Upper triangular weight matrices, which permit oblique warpings of the instance space, are more appropriate for many tasks. The weighting function (Equation 8) need not be constant, but could be a polynomial or any arbitrarily complex function (e.q., a connectionist network). Furthermore, instead of employing a single set of weights for the entire instance space, separate sets of weights could be associated with specific queries, feature values, instances, classes, or some function of them. For example, a query-specific weighting method would modify Equation 6 by defining the weighting function based on the given query:

$$d(\mathbf{x},\mathbf{q}) = \left(\sum_{f \in \mathbf{F}} w(f,\mathbf{q}) \cdot \delta(x_f, q_f)^r\right)^{\frac{1}{r}} \tag{10}$$

In summary, many of k-NN's "fixed" design decisions are, in fact, optimizable parameters. However, we will focus on only one such decision: the definition of (constant) weighting functions. This is cause for concern. Ideally, these design decisions, and others such as the definition for normalizing continuous values (Turney 1993), should be optimized when comparing the efficacy of alternative constant weighting methods. We leave this as a goal for future research, and fix these other design decisions in our experiments (Section 4). Further information related to lazy classification algorithms can be found elsewhere (e.g., Vapnik 1992; Bottou and Vapnik 1992; Friedman 1994; Atkeson et al. 1996a).

Table 1. Dimensions For Distinguishing Feature Weighting Methods

Dimension	Possible Values
Bias	{Performance, Preset}
Weight Space	{Continuous, Binary}
Representation	{Given, Transformed}
Generality	{Global, Local}
Knowledge	{Poor, Intensive}

3. A Framework for Feature Weighting Methods

The feature weighting methods reviewed in this section are embedded in lazy algorithms that employ variants of the distance function shown in Equation 6. In particular, they differ in that they do not enforce the constraint shown in Equation 9 (i.e., they allow weight settings to differ among the features). These types of algorithms have been frequently examined in the machine learning literature, but it is not obvious how they all relate nor on what tasks their biases are particularly appropriate.

Feature weighting methods can be organized and dichotomized along several dimensions. We focus on the dimensions shown in Table 1.

By *bias*, we refer to whether the weight learning bias is guided by feedback from the *performance* algorithm (i.e., here, the classifier) or whether it is instead a *preset* bias (e.g., maximize intra-class similarity and minimize inter-class similarity) that does not incorporate performance feedback. By *weight space*, we distinguish *feature weighting* from *feature selection* algorithms. The latter are a proper subset of feature weighting algorithms that employ binary weights (i.e., 0 or 1), meaning that the feature is either retained or deleted. By *representation*, we distinguish algorithms that use the *given* representation from those that *transform* the given representation into one that might yield better performance. Feature weighting algorithms can also be distinguished by their *generality*; while most algorithms learn settings for a single set of weights that are employed *globally* (i.e., over the entire instance space), other algorithms assume weights differ among *local* regions of the instance space. Finally, the *knowledge* dimension distinguishes knowledge-poor algorithms from others that employ domain specific knowledge to set feature weights.

We use these dimensions to frame our discussion of feature weighting methods in this section. Further decompositions of algorithms are described as needed. Some alternatives to feature weighting are mentioned briefly to

provide additional intuition. We elaborate on related work in Section 6. We include more detail for algorithms distinguished by the first dimension than the others because our empirical study in Section 4 focuses primarily on this dimension. Space constraints prevent a more detailed discussion for the other dimensions.

3.1. *Bias: Performance vs. Preset*

The distinction between *performance* and *preset* biases is described as *open loop* and *closed loop* in statistics, and as *wrapper* and *filter* models in machine learning (e.g., John et al. 1994). We view this as an issue of learning bias. Weighting methods that use feedback from the performance function during training attempt to incorporate the classifier's bias during weighting. Those that do not incorporate some alternative, preset bias. We further distinguish these two biases into several sub-categories, which are described in Sections 3.1.1 and 3.1.2, respectively. We selected one algorithm from each of the following subsections for our experiments.

3.1.1. *Performance Bias*

Performance bias methods presumably have an advantage; their search for feature weight settings is guided by how well those settings perform. Thus, there should be no mismatch between the biases of the weighting and performance algorithms. We distinguish weighting methods that incorporate performance feedback into two groups: those that perform *online* search in the space of weights (i.e., sequentially processing each training instance once) and those that perform *batch* optimization (i.e., repeatedly pass through the training set).

Online Optimizers

Several lazy algorithms optimize feature weight settings using one pass through the training set. Given a query **q** to classify, these algorithms iteratively modify feature weights to (a) decrease its distance to nearby[1] instances with the same class and (b) increase its distance with similar instances in other classes. These algorithms are sensitive to the presentation ordering of the training data.

An early example of this approach was used by Salzberg (1991) in EACH, a k-NN variant that updates feature weights by a fixed amount after classifying

[1] As defined by the current distance function.

each training instance. The objective of feature weighting was to improve EACH's tolerance of noise. For correct classifications, the weights of all matching features are incremented, using

$$w(f) = w(f) + \Delta, \tag{11}$$

and all mismatching features' weights are decremented by this same amount. Incorrect classifications cause the weights of *mismatching* feature to be incremented, while the weights of matching features are decremented. This procedure was expected to assign high weights to relevant features and lower weights to others. Salzberg found that, by selecting good values for Δ, this algorithm consistently improved classification performance vs. Equation 9.

Salzberg's algorithm influenced the design of IB4 (Aha 1992), which updates weight settings using[2]

$$w(f) = \max\left(\frac{\text{CumulativeWeight}_f}{\text{WeightNormalizer}_f} - 0.5, 0\right), \tag{12}$$

where $\text{CumulativeWeight}_f$ is assumed to asymptote to half of the $\text{WeightNormalizer}_f$ for seemingly irrelevant features. When classifying an instance **x** with another (training) instance **y**, the degree to which IB4 updates $\text{CumulativeWeight}_f$ depends on the concept distribution. Let $\Lambda(\mathbf{x}, \mathbf{y}) = \max(p(x_c), p(y_c))$ (i.e., the probability of the more probable class among **x** and **y**'s classes). Then $\text{CumulativeWeight}_f$ is incremented by

$$\text{CumulativeWeight}_f \stackrel{+}{=} \begin{cases} 1 - \delta(x_f, y_f) \cdot (1 - \Lambda(\mathbf{x}, \mathbf{y})) & \text{if } (x_c = y_c) \\ \delta(x_f, y_f) \cdot (1 - \Lambda(\mathbf{x}, \mathbf{y})) & \text{Otherwise} \end{cases} \tag{13}$$

and WeightNormalizer is always incremented by $(1 - \Lambda(\mathbf{x}, \mathbf{y}))$. This procedure sensitizes IB4 to skewed concept distributions and avoids the need for a fixed weight adjustment parameter. Aha (1990) reported good results for IB4 vs. 1-NN on tasks involving irrelevant features.

Kira and Rendell (1992) noted that this algorithm assumes a uniform distribution for irrelevant feature values. They introduced a binary (i.e., feature selection) weighting algorithm named RELIEF that removes this constraint. It iterates through a weight-updating procedure m times that (1) selects a random training instance **x**, (2) locates **x**'s most similar positive (**p**) and negative (**n**) training instances, and (3) updates each feature's weight using

[2] IB4 learns a separate set of weights per concept, but this is ignored here to simplify the presentation.

$$w(f) = w(f) - \delta(x_f, p_f) + \delta(x_f, n_f) \tag{14}$$

When classifying, RELIEF maps these weights to binary values; if $w_f \geq \tau$, then f's weight is mapped to 1, and otherwise 0, where τ is a user-specified *relevance* parameter. Kira and Rendell reported good results for RELIEF on parity tasks. Kononenko (1994) reported good results when modifying RELIEF to average the contributions of $\mathbf{x}$'s k nearest positive and negative instances, so long as k was properly tuned. He also extended it for application to noisy, incomplete, and multiclass data.

We selected this modification, RELIEF-F, for inclusion in our experiments in Section 4. However, we did not map w_f to a value in $\{0, 1\}$, and allowed it to process each training instance only once.

Online optimizers generally assign low weights to completely irrelevant features and outperform standard k-NN for some applications with many irrelevant features. However, it is not clear to what extent they can recognize redundant or highly interacting features. For example, IB4 performed poorly on a task with many partially relevant features (Aha and Bankert 1994).

Batch Optimizers

These weighting methods optimize feature weights by repeatedly processing instances. Some of these methods require only knowledge of the function value to be approximated at each problem state (e.g., *simulated annealing* and *genetic algorithms*) while others use knowledge of the function's gradient (e.g., Lowe 1995).

Kelly and Davis (1991) and Skalak (1994) used genetic algorithms (GAs) to learn continuous feature weights for lazy learning algorithms. GAs loosely mimic processes of biological evolution. They repeatedly apply genetically-inspired operators (e.g., crossover, mutation) to a population of current problem solutions in the hope of obtaining higher scores on a given *fitness function*, and terminate after a fixed number of iterations or when a heuristic stopping criterion indicates no recent improvement. Kelly and Davis (1991) designed GA-WKNN to learn continuous feature weights using five genetic operators and a fitness function based on both the number of misclassified training instances and recency. GA-WKNN attained lower error rates than k-NN for three datasets.

Skalak (1994) instead used a degenerate GA to select features (i.e., binary weights) for 1-NN. His algorithm repeatedly mutates a single bit sequence, keeps the string with the higher classification accuracy on the training set, and terminates after a fixed number of iterations since finding a new best string.

This algorithm attained higher accuracies than 1-NN on four datasets while halving the number of features used to compute distances.

Other batch weight learning methods increase learning speed by exploiting knowledge of the function's gradient, which can substantially increase learning rates when the target function is reasonably smooth. Lowe (1995) employed this approach in the *variable kernel similarity metric* (VSM), which computes distances using a function similar to Equation 6. Feature weights are optimized using conjugate gradient (Press et al. 1992) to minimize summed leave-one-out classification error (LOOCE) on the training set.[3] The derivative of this error with respect to each feature weight is used to guide the conjugate gradient procedure. Lowe reported that VSM performed as well as or better than several other algorithms on two datasets yet required far less training time than some of the other algorithms.

While we selected a similar weight-optimization method for our empirical evaluation, we did not chose the VSM because it introduces additional complexity that prevents isolation of its feature-weighting algorithm. That is, the VSM assigns weights to instances as defined by an optimized Gaussian function of their distances, and then uses these weights to bias classification predictions. We instead selected a simplification of the VSM, named k-NN$_{VSM}$ (Wettschereck 1995a), which replaces the Gaussian kernels of Lowe's VSM with a differentiable k-NN function and eliminates the use of pre-assigned instance weights.

k-NN$_{VSM}$ employs a distance-weighted voting scheme (e.g., Dudani 1975, see also Equation 4). This algorithm first computes the distances between all pairs of training instances using Equation 9. It then assigns to k the value that minimizes LOOCE on the training set. Finally, it, like the VSM, uses conjugate gradient to optimize feature weights so as to minimize LOOCE training error. The error function is

$$E = \sum_{\mathbf{x} \in \mathbf{X}} \sum_{c_j \in \mathbf{J}} \left(1(\mathbf{x} \in c_j) - p(c_j|\mathbf{x})\right)^2 \tag{15}$$

where $p(c_j|\mathbf{x})$ is defined as in Equation 4. Wettschereck (1995a, 1994) discusses further details on k-NN$_{VSM}$, including the derivation of the gradient of E with respect to each feature weight. He found that k-NN$_{VSM}$ can learn good feature weights in a variety of domains although a relatively large number of design decisions can heavily influence its performance.

[3] Error is a function of the differences between the probabilities computed by the VSM and the target class. A stabilizing term is added to this error to prevent large weight changes for small training sets.

3.1.2. *Preset Bias*

This section summarizes approaches that do *not* use feedback from the classifier to assign weight settings. Instead, they use a pre-existing model's bias. Three groups of these *preset bias* methods are described in this section: those based on (simple) *conditional probabilities*, *class projection*, and *mutual information*, respectively.

Conditional probabilities

Creecy et al. (1992) introduced two simple feature-weighting methods that use conditional probabilities to assign feature weights. Both algorithms work only with binary features, so they discretize continuous features and *binarize* discrete features (i.e., each discrete feature f defined over a set V_f of values was replaced with $|V_f|$ binary features).

First, their *cross-category feature importance* (CCF) method averages weights across classes using

$$w(f) = \sum_{c_j \in \mathbf{J}} p(c_j|f)^2 \tag{16}$$

Creecy et al.'s (1992) motivation for designing CCF was to assign higher weights to features that occurred in fewer classes. However, this algorithm is not sensitive to the distribution of a feature's values across classes (i.e., a feature's weight is independent of the class), which seems unrealistic for practical applications. Therefore, they designed a second weighting method, named *per category feature importance* (PCF), which assigns feature weights using

$$w(f, c_j) = p(c_j|f) \tag{17}$$

That is, the weight value of a feature f for a class c_j is defined as the conditional probability that an instance is a member of c_j given its value for f, averaged across all values for f. This algorithm assigns high weight values to features that have high correlations with the given class. They concluded that their weighted methods outperform non-weighted methods, but did not compare these two weighting methods.

Mohri et al. (1993) found that PCF is sensitive to concept distributions; it tends to classify too many instances according to the majority class. PCF performed poorly in their experiments (Mohri and Tanaka 1994). For example, the simpler CCF attained higher accuracies than PCF on six of eight tasks.

Class projection

Stanfill and Waltz (1986) introduced the *value-difference metric* (VDM), a more sophisticated similarity function defined for discrete features. This was the predecessor of CCF and PCF; it assigns higher weights to features whose distribution of values across classes are highly skewed. Although it also computes conditional probabilities, it does not binarize features. Previously, most similarity functions on discrete values either binarized them or employed the simple *overlap* function, which simply counts the number of mismatching features. In contrast, the VDM allows similarity to vary among individual feature values. Distance is defined as the sum of feature differences for two instances using

$$d(\mathbf{x}, \mathbf{y}) = \sum_{f \in \mathbf{F}} w(f, x_f) \cdot \delta(f, x_f, y_f) \tag{18}$$

$$w(f, v) = \sqrt{\sum_{c_j \in \mathbf{J}} p(\mathbf{x} \in c_j | \mathbf{x} \in \mathbf{X}, x_f = v)^2} \tag{19}$$

$$\delta(f, v_1, v_2) = \sum_{c_j \in \mathbf{J}} \left(p(\mathbf{x} \in c_j | \mathbf{x} \in \mathbf{X}, x_f = v_1) - p(\mathbf{x} \in c_j | \mathbf{x} \in \mathbf{X}, x_f = v_2)\right)^2 \tag{20}$$

where Equation 19 computes f's weight when its value is v, and $p(\mathbf{x} \in c_j | \mathbf{x} \in \mathbf{X}, x_f = v)$ is the observed relative frequency that instances in the training set with value v for feature f are in class c_j. Equation 20 computes the difference of two values for a given feature. It assigns greater differences to values whose corresponding sets of instances have highly disparate class distributions. Thus, two instances are similar if they have feature values whose respective projections on the training set have similar class distributions.

Stanfill and Waltz (1986) used the VDM on a vowel pronunciation task, but did not compare it with other weighting methods. A few non-weighting variants of the VDM (e.g., Cost and Salzberg's (1993) MVDM and its extension to continuous features by Ting (1994)) have performed well on some tasks. However, surprisingly little evidence exists that feature-weighting variants of the VDM improve classification performance compared to unweighted variants. Furthermore, Daelemans and van den Bosch (1992) report that a method that assigns weights using information gain (see Section 3.1.2) outperforms the VDM on a grapheme to phoneme conversion task, and Aha (1990) reported that even the overlap function outperforms the VDM on some classification tasks. We also found little performance difference for the MVDM with and without weights in our experiments in Section 4.3. Thus,

additional research is required to determine the conditions under which feature weighting can improve the classification performance of VDM variants.

Mutual information

This section describes a third approach for assigning feature weights using conditional probabilities. Like class projection, discrete features are not binarized. This approach also easily admits extensions for use with continuous features, which we discuss after describing the basic algorithm.

A feature weighting algorithm should assign low weights to (less relevant) features that provide little information for classification and higher weights to features that provide more reliable information. Towards this goal, the *mutual information* (MI) (Shannon 1948; McGill 1955) between the values of a feature and the class of the training examples can be used to assign feature weights. The MI of two variables is the reduction in uncertainty of one variable's value given knowledge of the other's value (Cover and Thomas 1991). This can be computed using

$$w(f) = \sum_{v \in \mathbf{V}_f} \sum_{c_j \in \mathbf{J}} p(c_j, x_f = v) \cdot \log \frac{p(c_j, x_f = v)}{p(c_j) \cdot p(x_f = v)} \qquad (21)$$

where $p(c_j)$ is the frequency of class c_j among the training set $\mathbf{X}$ and $p(x_f = v)$ is the frequency of value v for f among instances in $\mathbf{X}$. This equation assigns zero to features that provide no information about the class, and a value proportional to $\log(|J|)$ to features that completely determine the class (i.e., assuming a uniform distribution on classes).

Daelemans and van den Bosch (1992) introduced an extension of this approach that assigns a feature's (normalized) *information gain* (Quinlan 1986) as its weight rather than Equation 21:

$$\begin{aligned} w(f) \;=\; & -\sum_{c_j \in \mathbf{J}} p(c_j) \log(p(c_j)) \\ & -\sum_{v \in \mathbf{V}_f} \sum_{c_j \in \mathbf{J}} -p(c_j | x_f = v) \log p(c_j | x_f = v) p(x_f = v) \end{aligned} \qquad (22)$$

This equation subtracts the average information entropy of a feature from the information entropy of the training set $\mathbf{X}$. They reported that this weighting method substantially improved k-NN's accuracy on a word hyphenation task. Subsequently, van den Bosch and Daelemans (1993) found similarly good

results for this weighting method on a grapheme-to-phoneme conversion task, and Daelemans et al. (1993) reported that it obtained the best performance, among three algorithms, for a stress assignment task.

These two equations do not define how to compute the MI for *continuous* features. Wettschereck and Dietterich (1995) used a simple approach to do this; it divides continuous features into a pre-determined number I of intervals, treating all values within a given interval as equal. They found that this batch weighting method improved the performance of EACH (Salzberg 1991) compared to its online algorithm for setting weights (Equation 11). We use a similar approach in Section 4 that avoids the need to predetermine I. Features are instead discretized using Fayyad and Irani's (1993) algorithm[4] and then treated in the same manner as discrete features. Both Mohri and Tanaka (1994) and Ting (1994) have shown the utility of this discretization method for lazy learning algorithms.

3.2. *Weight Space: Weighting vs. Selection*

In Section 3.1, we discussed the first dimension for distinguishing feature weighting algorithms (i.e., whether they tune weights using feedback from the classifier). This section addresses the second dimension, which concerns the space of values explored for assigning feature weights.

Feature selection algorithms assign binary weights to features. Thus they are restricted to a subset of the weight assignments learnable using continuous feature weights. Feature selection has been studied for several decades (e.g., Fu 1968; Mucciardi and Gose 1971; Cover and van Campenhout 1977). Several researchers have recently used various feature selection methods for lazy learning algorithms. These methods select features using

- an induced decision tree (Cardie 1993; Kibler and Aha 1987),
- random mutation hill-climbing (Skalak 1994),
- parallel search (Moore and Lee 1994),
- beam search with stepwise selection (Aha and Bankert 1994), and
- stepwise feature removal in oblivious decision trees (Langley and Sage 1994).

The first of these methods employs a preset weighting bias while the others exploit performance feedback. All report accuracy and/or speed improvements over 1-NN or k-NN. Feature selection algorithms can often reduce the dimensionality of a learning task. If the de-selected features are completely irrelevant, then this could significantly increase an algorithm's learning rate.

[4] This discretization algorithm computes the entropy of a set of possible discretization points and recursively accepts binary splits that yield maximal entropy reduction as long as the minimum description length principle is satisfied.

Aha and Bankert (1994) reported that their feature selection algorithm had benefits over IB4, the online feature weighting algorithm, for a task with a large number (204) of features. They hypothesized that IB4 fares poorly under such conditions because it trades off learning rate for a much larger space of continuous weights. Thus, its learning rate is slow.

More recently, Kohavi et al. (1995) empirically compared a weighting and selection method. Their DIET algorithms, performance biased methods that respectively use best first and hill climbing searches through a constrained space of feature weights, outperformed a similar feature selection algorithm in tasks where features vary in their relevance for classification. They also anticipate the tradeoff between learning rate and weight space size.

In summary, these studies and similar ones using other learning algorithms suggest that feature selection algorithms perform best when the features used to describe instances are either highly correlated with the class label or completely irrelevant. Feature weighting is more appropriate for tasks where features vary in their relevance, but such methods search larger spaces of weight assignments. Additional research is required to further investigate this tradeoff.

3.3. *Representation: Given vs. Transformed*

A third dimension for distinguishing feature weighting algorithms concerns whether the set of features used to represent the instances is transformed (i.e., replaced with a different set) before weighting. A substantial shortcoming of all methods that simply assign weights to individual features is their insensitivity to interacting or correlated features. This can be addressed either by using distance functions that combine weights (e.g., using upper triangular weight matrices) or by transforming the given representation before weighting features.

QM2m (Mohri and Tanaka 1994) is an example of this latter approach. It assigns the absolute values computed by *Quantification Method II* (QM2) (Hayashi 1952) to set its feature weights in the following variant of Equation 6:[5]

$$d(\mathbf{x}, \mathbf{q}) = \sqrt{\sum_{f \in \mathbf{F}} \left(\sum_{f' \in \mathbf{F'}} |w(f', f)| \cdot \delta(x_f, q_f) \right)^2} \tag{23}$$

where $\mathbf{F}'$ is the set of transformed (i.e., new) features and $w(f', f)$ is the weight of the given feature f for the transformed feature f'. This representation

[5] Symbolic features with V_f values are replaced with $|V_f|$ binary features before this transformation.

transformation is performed using QM2, which is a supervised version of *principal components analysis* (PCA).[6] QM2 locates a new set of features such that the sums of the squared distances of each instance to its projections on the successive feature dimensions are minimized. The value of transformed feature f' for transformation of $\mathbf{x}$ is computed using

$$\sum_{f \in \mathbf{F}} w(f', f) x_f \tag{24}$$

where these weights are calculated to maximize, for each new feature f', the ratio of the variance between each class' instances to the variance of all instances. Kawaguchi (1978) describes this algorithm in more detail.

Mohri and Tanaka (1994) reported good performance for QM2m. QM2 can be used to reduce the dimensionality of the data by removing the transformed features with the lowest variation (Mohri and Tanaka 1995; Wettschereck 1994). Mohri and Tanaka (1995) investigate how to determine the number of transformed features that can be removed.

Mohri and Tanaka (1994, 1995) also introduced QM2y, which uses a different distance function:

$$d(\mathbf{x}, \mathbf{q}) = \sum_{f' \in \mathbf{F'}} \left(\sum_{f \in \mathbf{F}} w(f', f) x_f \cdot \sum_{f \in \mathbf{F}} w(f', f) q_f \right)^2 \tag{25}$$

This algorithm performed best among their selected set of lazy algorithms, but hypotheses explaining why have not yet been investigated.

Another lazy learning algorithm that supports feature transformation is IB3-CI (Aha 1991). This is a knowledge-intensive extension of the noise-tolerant IB3 algorithm (Aha et al. 1991). It uses a Bayesian approach, adapted from (Schlimmer 1987), to direct its search through a space of logical feature combinations, and uses a competitive feature selection approach to assign binary weights. Aha (1991) reported good results for IB3-CI in comparison with lazy algorithms that do not perform representation change. However, it requires domain-specific knowledge to constrain and intelligently prune the space of feature combinations.

A significant disadvantage of feature transformation methods is that the transformed features are often not meaningful. This can constitute a significant shortfall of these methods if inspection of the (transformed) classifier

[6] QM2 is preferable to PCA, which can lose information since (1) PCA orders principal components by decreasing functions of their input data variations and (2) the variable with the lowest variation might actually be the one with the highest predictive relevance (Kshirsager 1972). QM2 also is sensitive to concept skew when ordering the new variables.

is necessary (i.e., additional constraints on the transformation process are required to guarantee comprehensibility).

3.4. *Generality: Global vs. Local*

A fourth dimension for distinguishing feature weighting algorithms concerns whether the weights apply *globally* (i.e., over the entire instance space) or *locally* (i.e., differ in different parts of the instance space). Although many feature-weighting algorithms use a global scheme, their assumption that feature relevance is invariant over the instance space is constraining and sometimes inappropriate.

Two types of local weighting schemes are popular. The first assigns a different weight to each value of a feature (e.g., the VDM (Section 3.1.2)). Although this allows feature relevance to vary over the values of a feature, it still constrain weights to be identical for all instances with the same feature value. The second local weighting scheme removes this constraint by allowing feature weights to vary as a function of the instance. We discuss several examples of this approach below.

The asymptotic error rate of first nearest neighbor is no more than twice that of the Bayes optimal classifier (Cover and Hart, 1967). Short and Fukunaga (1980, 1981) and Fukunaga and Flick (1982, 1984) utilized this fact to compute feature weights for a weighted distance function. They estimated the finite sample risk from the local neighborhood of a given instance. They then minimized the difference between the finite sample risk and the asymptotic risk to obtain a local distance function for each instance. Hence, to classify a query the authors first find its k nearest neighbors, as defined by k-NN, and then compute each neighbor's local distance function to find the nearest neighbor. Myles and Hand (1990) extended this approach for multiclass problems.

Fukunaga and Flick (1984) noted two disadvantages of this local weighting approach. First, it lacks regularization; one noisy sample can cause an improper distance computation. Second, the large number of distinct local distance functions can obscure useful feature information. Therefore, they propose a global distance function that combines the information from all local functions into one global set of weights for a weighted Euclidean distance function.

In this vein, Aha and Goldstone (1992) combined local with global distance functions to compute instance-specific weights for their GCM-ISW algorithm. Distance for continuous features was defined as

$$d(\mathbf{x}, \mathbf{q}) = \sqrt{\sum_{f \in \mathbf{F}} w(f, \mathbf{x}, \mathbf{q}) \cdot \delta(x_f, q_f)^2} \tag{26}$$

where instance specific weights are dynamically assigned using

$$w(f, \mathbf{x}, \mathbf{q}) = w(f, \mathbf{x}) \cdot \sqrt{1 - |x_f - q_f|} + w(f) \cdot \left(1 - \sqrt{1 - |x_f - q_f|}\right) \quad (27)$$

Equation 27 adds the contributions of stored instance **x**'s instance-specific weight $w(f, \mathbf{x})$ and the global weight $w(f)$. As the difference between the values of f for these instances decreases, **x**'s instance-specific weight contributes more to the dynamically assigned weight $w(f, \mathbf{x}, \mathbf{q})$. Likewise, the global weight $w(f)$ is used more as this difference increases. Thus, this distance function depends more on instance-specific weights when the new instance is similar to the stored instance, and depends more on the global weights when they are not similar. Similarly, the updating algorithm for instance-specific weights has more influence when these instances are similar.

GCM-ISW was shown to correlate significantly better with subject data than did its non-weighting and global weighting variants (Aha and Goldstone 1992). The subjects' target concept was designed such that feature relevance varied in different parts of the instance space.

GCM-ISW's definition of similarity is not symmetric; frequently, it yields $d(\mathbf{x}, \mathbf{y}) \neq d(\mathbf{y}, \mathbf{x})$. Ricci and Avesani (1995) noted that asymmetric local similarity functions defined on continuous features have another degree of freedom: a feature's weight can differ depending on whether a query's value for that feature is greater or less than the value of the stored instance. Assuming that a feature f is continuous, they defined an anisotropic and asymmetric distance measure with *directed* weight settings using

$$w(f, \mathbf{x}, \mathbf{q}) = \begin{cases} w_{x_{f_>}} & \text{if } x_f \geq q_f \\ w_{x_{f_<}} & \text{if } x_f < q_f \end{cases} \quad (28)$$

where $w_{x_{f_<}}$ and $w_{x_{f_<}}$ are the weights of instance **x** at feature f on the respective "sides" of x_f. They obtained favorable results with their local weighting scheme as compared to standard 1-NN and Salzberg's (1991) EACH on four data sets from the UCI repository (Murphy 1995).

Other purely local (and lazy) approaches were recently introduced by Hastie and Tibshirani (1994) and by Friedman (1994). Hastie and Tibshirani (1994) compute a separate distance metric for each query through an iterative process. The main idea is to employ discriminant analysis to shrink the neighborhood

around each query in directions orthogonal to the decision boundary (i.e., thus giving greater weight to features whose axes are closer to perpendicular with this boundary). Their weighting algorithm works as follows:

1. Initialize a weight matrix **D** with the identity matrix.
2. Find $\mathbf{K_q}$, the k nearest neighbors to a query **q** using **D**.
3. Use $\mathbf{K_q}$ to compute the weighted within and between sum of squares matrices **W** and **B**.
4. Update the weight matrix using $\mathbf{D} = \mathbf{W}^{-1/2}[\mathbf{W}^{-1/2}\mathbf{B}\mathbf{W}^{-1/2} + \epsilon\mathbf{I}]\mathbf{W}^{-1/2}$, where ϵ is a user-chosen parameter. (Steps 2-4 can be repeated indefinitely.)
5. Use **D** to classify **q** using k-NN.

The local statistics **W** and **B** used to compute the new local metric about **q** are the deviation of each pattern from its class-mean (**W**) and the deviation of each of the class-means from the mean of all k nearest neighbors (**B**).

A computationally more efficient approach is taken by Friedman (1994). He employs recursive partitioning to find the k nearest neighbors to a query. His *scythe* algorithm recursively zooms in on the query along the most relevant feature. That is, the most relevant feature is scaled at each step such that a fixed fraction of the given training examples fall outside of a predetermined range around the query. The training examples outside of that range are then discarded, the new "most relevant" feature is determined, and the process is repeated until only k training examples remain. The (local) relevance of each variable is estimated from the estimated reduction in prediction error that the knowledge of the value of that variable would yield.

Both Hastie and Tibshirani (1994) and Friedman (1994) report favorable results for their local approaches, in comparison to unweighted k-NN, on synthetic and "real" data sets. Atkeson, Moore, and Schaal (1996a, 1996b) survey the literature on locally weighted learning algorithms and report similar results for several robotic control tasks using locally weighted regression (LWR) algorithms for learning numeric functions. They distinguish *point-based* and *query-based* weighting methods, based on whether distances are computed for each stored instance or dynamically, for a specific query, as is done in LWR algorithms (Cleveland and Loader 1994). Finally, several case-based reasoning researchers have advocated using local distance functions with pre-determined weight assignments (e.g., Ashley and Rissland 1988; Skalak 1992). We expect that local weighting methods will continue to be a fruitful area for future research.

3.5. *Knowledge: None vs. Domain-Specific*

The fifth, final, and most important dimension for distinguishing feature weighting algorithms concerns their use of domain-specific knowledge. Such

valuable information is frequently used to constrain the representation of instances and selection of features (e.g., Stanfill and Waltz 1986). Several researchers have also demonstrated the utility of using knowledge to assign feature weights (e.g., Ashley and Rissland 1988; Skalak 1992). Approaches more closely related to the focus of this paper are algorithms that combine automated weight-learning components with domain intensive knowledge or heuristics (e.g., the set of possible transformations assumed when defining tangent distance functions (Simard et al. 1993)). We described some knowledge-intensive algorithms earlier (e.g., IB3-CI), and briefly detail two more below.

Cain et al. (1991) use a domain theory of rules to assign instance-specific weights. Like other algorithms that define local distance functions, their CBR+EBL algorithm combines instance-specific with global feature weights. They assign instance-specific weights to discrete features using an explanation-based learning (EBL) approach (Mitchell et al. 1986). Any feature appearing in an EBL tree that was generated to explain the instance's class is assigned a weight of 1. All other features for that instance are assigned an instance-specific weight of 0. Their distance function is

$$d(\mathbf{x}, \mathbf{q}) = -\text{similarity}(\mathbf{x}, \mathbf{q}) \tag{29}$$

where

$$\text{similarity}(\mathbf{x}, \mathbf{q}) = \frac{\sum_{\text{f}\in\mathbf{F}} \alpha \cdot (1 - \delta(\text{x}_\text{f}, \text{q}_\text{f})) + \sum_{\text{f}\in\mathbf{F}} \beta \cdot \text{w}(\text{f}, \mathbf{x}) \cdot (1 - \delta(\text{x}_\text{f}, \text{q}_\text{f}))}{\alpha|\mathbf{F}| + \sum_{\text{f}\in\mathbf{F}} \beta \cdot \text{w}(\text{f}, \mathbf{x})} \tag{30}$$

where α determines the degree to which the nearest neighbor algorithm is used to classify instances and β determines the degree to which EBL-determined feature weights are used to influence its similarity function. That is, previous instances are ignored when $\alpha = 0$ while the domain theory is ignored when $\beta = 0$. In an experiment with a sparse dataset (i.e., 50 instances, 76 features), they reported substantial leave-one-out accuracy improvement when using both previous instances and the domain theory in comparison with using only one of these. Thus, they demonstrated the utility of using knowledge to set instance-specific weights for a lazy learning algorithm.

PROTOS (Porter et al. 1990) is a sophisticated *case-based reasoning* system designed initially for a clinical audiology classification task. It builds a semantic network whose links relate features, instances, and classes. It uses feedback from the user to refine its knowledge. The initial settings for PROTOS' instance-specific feature weights are determined from the certainty with which the feature's presence can be inferred from category membership when PROTOS builds explanations of a given instance's classification.

Knowledge-based pattern matching is used to generate these explanations; it determines whether two values of a feature match by searching for chains of relations in the semantic network linking the two instances' features. Domain-specific heuristics help determine the degree to which two features match. Feature weights can be subsequently modified by the domain expert whenever PROTOS fails to retrieve the correct case to a query. PROTOS' distance function uses a variant of the *context model* (Medin and Schaffer 1978): it subtracts, from 1.0, the contributions of non-matching features according to their relevance weights. Bareiss (1989) reported that PROTOS recorded higher accuracies than did knowledge-poor k-NN on the audiology task.

In summary, knowledge can be used to assign feature weights for use in a k-NN variant, as in CBR+EBL, or through an expert's critique of the algorithm's predictions for feature relevance, as is done in PROTOS. In both cases *instance-specific* weights are used, and we expect this trend to continue in knowledge-intensive approaches.

4. Comparative Evaluation

It is difficult to predict the comparative behavior of feature weighting methods for lazy learning algorithms. Although large scale empirical comparisons exist for other classes of algorithms (e.g., Michie et al. 1994), they do not exist for this class. Instead, most previous comparisons among feature weighting algorithms tend to focus on a specific pair of algorithms (e.g., Wettschereck and Dietterich 1995; Kohavi et al. 1995).

In this section we investigate the comparative capabilities of some of the algorithms described in Section 3. The purpose of this empirical study is to formulate and evaluate a set of trends[7] regarding the relative strengths and weaknesses of groups of feature weight learning algorithms. We focus on the first two dimensions described in Section 3 (i.e., bias and weight space), ignoring the others due to space constraints. The following subsections describe the selected algorithms, the selected datasets, our empirical methodology, and the results. We then summarize these results with explanatory hypotheses and evaluate them. Section 5 includes a discussion on our findings with respect to our framework in Section 3.

[7] We prefer the term "trend" over "hypothesis" to indicate that the design and execution of this as well as many other empirical studies does not allow the formulation of hypotheses that withstand statistical scrutiny.

Table 2. Algorithms Selected for Experimentation

Name	Category	Sub-category
k-NN	Control	–
RELIEF-F	Performance	Online Optimizer
k-NN$_{VSM}$	Performance	Batch Optimizer
CCF	Preset	Conditional probabilities
VDM	Preset	Class projection
MVDM	Preset	Class projection
MI	Preset	Mutual information

4.1. *Selected Algorithms*

No weight learning algorithm will perform best for all applications since each implements a different bias; they will have substantially different performance on some problems (Mitchell 1990; Schaffer 1994). We selected one weight-learning method from each sub-category of the bias dimension (Section 3.1) to compare their capabilities. These algorithms, including the baseline k-NN algorithm, are shown in Table 2.

The MVDM (i.e., VDM without feature weights) is also included so we can evaluate the influence of the modified feature difference function (Equation 20) independent of the feature weighting method (Equation 19) on VDM's behavior.[8]

To allow for controlled experimentation along the second dimension (i.e., continuous vs. binary weight space), selected experiments were conducted with one preset bias method (MI) and one performance bias method (RELIEF-F).[9] In these experiments, we set a varying number of the lowest-valued weights to 0 and all other weights to 1.

4.2. *Selected Data Sets*

We selected fourteen datasets for our study. Ten of these were chosen to evaluate the selected algorithms' capabilities under controlled conditions for specific data characteristics. These were designed to evaluate the selected algorithms in the context of tasks with irrelevant features, interacting features, redundant features, and/or features with varying relevance. Four of these datasets were created for this study, three of which are shown in Figure 1:

[8] This is necessary since the VDM is the only method that differs from k-NN in more than the definition of Equation 9.

[9] These two were selected because they are the most computationally efficient methods in their respective categories.

- The *banded* task has axis-parallel decision boundaries. The horizontal dimension of this task is completely irrelevant.
- The decision boundary in the *sinusoidal* task is a sine curve. The vertical dimension in this task is nearly completely irrelevant.
- The *gauss-band* task was constructed by combining the input features of the *banded* task and four Gaussian distributions (variance 0.025). A fifth boolean feature indicates whether the inputs from the *banded* task or the Gaussian distribution determine the output. Therefore, this task is an example of a task with interacting features.
- Another task with highly interacting features is the parity problem. We selected a *parity* task with 11 boolean features; seven are irrelevant while the sum of the other four determines the output (i.e., if the sum is an even number, then the class is 1 and otherwise is 0).

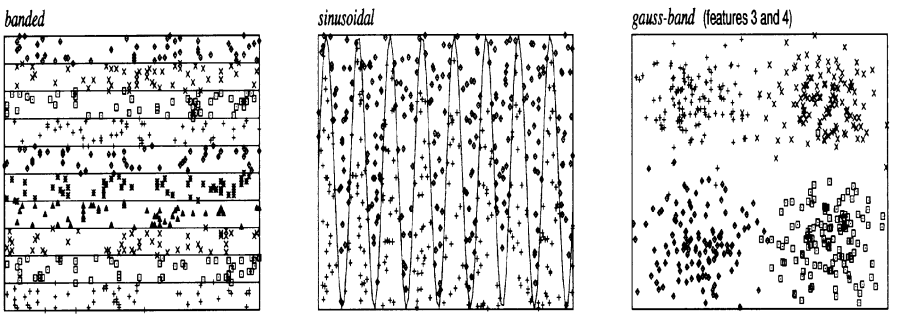

Figure 1. The distribution of examples from the different classes in three two-dimensional artificial data sets. The *banded* task has ten classes. *Sinusoidal* is a binary classification task. The rightmost graph depicts two of the five features of the *gauss-band* task with data points drawn from four Gaussian distributions (variance = 0.025), where different symbols indicate different classes. Lines represent the decision boundaries used to label the data.

The remaining datasets were drawn from the UC Irvine Repository (Murphy 1995). Some characteristics of these datasets are shown in Table 3. Additional dataset characteristics that are relevant to the evaluation are:

- The LED Display and Waveform datasets are also constructed from a data generator. The relevant features of the LED task are interacting, which allows us to further compare the abilities of performance and preset bias methods on such tasks. Some features in the Waveform task are more relevant than others; we will use this knowledge to examine the effect of continuous vs. binary feature weights.
- The Waveform-40 (LED-7+17B, LED-7+17C) task is identical to the Waveform-21 (LED-7) task with the addition of 19 (17) irrelevant features (i.e., having random values). These tasks were chosen to compare how

well different feature weighting algorithms tolerate irrelevant features when a relatively large number of features are irrelevant.

- The Cleveland, Hungarian, and Voting databases contain redundant features (i.e., some features can be removed in these datasets without any significant effect on the performance of k-NN (Wettschereck 1994)). In contrast, some datasets have no redundant features (e.g., Waveform, Isolet, and NETtalk).

Table 3. Characteristics of the selected datasets. B = Boolean, C = Continuous, D = Discrete. The relevant features in the datasets located above the horizontal divider are approximately equally relevant.

Domain	Set Size		Number and Type of Features	Number of Irrelevant Features	Classes
	Training	Test			
Banded	350	150	2 C	1	10
Sinusoidal	350	150	2 C	1	2
Gauss-band	350	150	4 C, 1 B	2	14
Parity	350	150	11 B	7	2
LED-7 Display	200	1000	7 B	0	10
LED-7+17B	200	1000	24 B	17	10
LED-7+17C	200	1000	7 B, 17 C	17	10
Waveform-21	300	100	21 C	0	3
Waveform-40	300	100	40 C	19	3
Cleveland	212	91	5 C, 3 B, 5 D	0	2
Hungarian	206	88	5 C, 3 B, 5 D	1	2
Voting	305	130	16 B	0	2
Isolet	1040	1040	617 C	0	26
NETtalk*	5000	2500	7 D	0	54

* Phonemes only

In summary, these datasets were selected to evaluate the selected algorithms' ability to tolerate different types of problematic features. We suspect that performance feedback methods will attain higher accuracies than the preset bias weighting methods discussed in Section 3.1 for datasets with interacting or redundant features because these preset bias methods compute weights independently for each feature.[10] Feature weighting methods should outperform feature selection methods for datasets with no irrelevant or redundant features (e.g., the NETtalk, Isolet, and Waveform-21 datasets).

[10] John, Kohavi, and Pfleger (1994) argued that performance feedback (*wrapper*) methods are preferable for feature selection algorithms, and empirical evidence now exists that supports their hypothesis (e.g., Doak 1992; Aha and Bankert 1994).

Finally, irrelevant features are often seen as *the* cause for k-NN's poor performance. Hence, several datasets with different numbers and types of irrelevant features were selected. However, we expect no substantial performance differences among the different feature weight learning algorithms for datasets with completely irrelevant features since such features are easily detected.

4.3. *Methodology and Initial Results*

We used the training/test set methodology to evaluate the generalization performance of the selected learning algorithms. Each dataset was randomly partitioned into a training and a test set. After training, the percentage of correct classifications on the test set was measured. This procedure was repeated 25 times to reduce statistical variation. The same training and test sets were used for each algorithm.

Leave-one-out cross-validation was used to tune the free parameters of the selected algorithms for all but the two largest datasets. Due to computational restrictions, the training sets for the NETtalk and Isolet datasets were split into two subsets: a sub-training and a cross-validation set. The algorithms were then trained on the sub-training set with various parameter settings and tested on the cross-validation set. The best parameter settings were then employed during classification in combination with the entire training set. The optimal value of k was estimated for all preset bias methods after feature weights were computed and for all performance feedback methods before and after learning feature weights. For k-NN$_{VSM}$, the number of training epochs was limited to the number of epochs required for minimization along one conjugate direction (see Press et al. 1992; Wettschereck 1995a). CCF, VDM, and MVDM have no free parameters. Fayyad and Irani's (1993) discretization algorithm was used to discretize continuous features for CCF, VDM, MVDM, MI (i.e., during feature weight computation), and RELIEF-F (i.e., when computing distance in the presence of missing feature values).

Using this methodology, we applied each algorithm to each dataset. Table 4 summarizes the results.

4.4. *Summary: Trends and their Evaluation*

These results, combined with additional insights gained during informal testing, suggest the following trends,[11] which we investigate later in this section:

- **T1**: Preset bias methods can suffer substantially when the data are not carefully pre-processed.

[11] Of course, these trends may be limited to the datasets tested.

Table 4. The effect of different feature weight learning algorithms on the generalization accuracy of k-NN. Shown are the average accuracy (and standard deviations) of k-NN with uniform feature weights and the relative percentage point differences in average accuracy attained by several feature weighting variants. We use boldfaced numbers in our tables to indicate that differences between the weighted and unweighted approach are more than three standard errors.

	Feature Weight Learning Algorithm						
	Control	Performance Bias Method		Preset Bias Method			
Dataset	none	Relief-F	k-NN$_{VSM}$	CCF	VDM	MVDM	MI
Banded	83.0±0.4	**11.2**	**12.8**	**12.8**	**12.8**	**12.8**	**10.8**
Sinusoidal	74.2±0.8	**5.9**	**14.4**	**-9.1**	**-9.1**	**-9.2**	**-4.6**
Gauss-band	78.3±0.5	**8.6**	**16.6**	**14.9**	**15.5**	**17.5**	**12.1**
Parity	67.3±0.1	**32.7**	**32.7**	1.3	1.7	1.9	2.2
LED-7 Display	72.7±0.4	-1.0	0.0	-1.5	-1.4	-1.3	-1.2
LED-7+17B	52.5±0.5	**19.2**	**15.5**	**9.2**	**19.4**	**18.9**	**19.4**
LED-7+17C	68.8±0.6	**3.3**	**2.0**	**-5.6**	**2.0**	**2.3**	**3.6**
Waveform-21	82.1±0.4	0.3	-0.5	**-6.1**	**-3.7**	**-3.9**	0.5
Waveform-40	81.3±0.9	1.7	1.2	**-3.4**	-0.7	-0.4	1.0
Cleveland	82.4±0.8	-0.5	0.0	-1.3	0.2	0.7	-0.6
Hungarian	82.6±0.7	**-2.5**	-0.4	-0.1	0.1	0.0	0.1
Voting	92.6±0.7	**2.9**	**2.5**	1.0	**2.1**	**2.1**	**2.0**
Isolet	84.2±0.3	0.4	**1.9**	**-1.1**	**-3.9**	**1.6**	**1.6**
NETtalk	69.6±0.2	**9.2**	**6.6**	**7.7**	**10.0**	**12.1**	**9.7**

- **T2**: Performance bias methods attain higher accuracies than preset bias methods for tasks with interacting features.
- **T3**: Performance bias methods have faster learning rates than preset bias methods.
- **T4**: Feature weighting algorithms achieve higher generalization accuracies than feature selection algorithms for tasks where some features are useful but less important than others.

We also found that most feature weight learning algorithms can tolerate completely irrelevant features unless there are many highly interacting features, which agrees with findings from previous studies of weight learning algorithms (e.g., Aha 1992; Kira and Rendell 1992). An exception is the performance of the preset bias methods on the *sinusoidal* task, which we investigate in **T1** below. A further interesting result was obtained for the Waveform-40 task; k-NN's average performance dropped by less than one percentage point when the 19 irrelevant continuous features were added. Simultaneously, the *relative* accuracies of all the feature weighting algorithms increased as expected. However, despite the large number of irrelevant features, and that RELIEF-F, k-NN$_{VSM}$, and MI correctly computed low weights

for them (e.g., Figure 2), none of the algorithms substantially outperformed k-NN. We surmised that k-NN's surprisingly good performance was because 300 training instances suffice to create a densely populated manifold in the instance space for this task. Therefore, we ran experiments on this task with a smaller training set (i.e., 100 training examples). The results (Table 5) reveal that 300 training examples are indeed too many to illustrate the faster learning rates of feature weighting algorithms for these Waveform tasks.

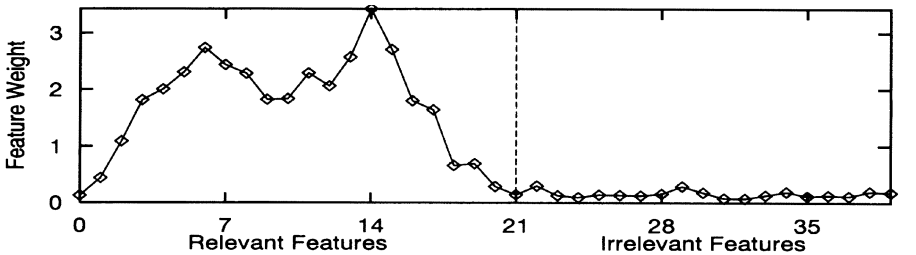

Figure 2. Feature weights computed by MI for the Waveform-40 task.

Table 5. Average accuracies for the Waveform-21 and Waveform-40 tasks for differently sized training sets.

Training Set Size	Feature Weight Learning Algorithm			
	control	Performance bias method		Preset bias method
	none	Relief-F	k-NN$_{VSM}$	MI
Waveform-21				
300	82.1±0.9	82.4	81.6	82.6
100	77.0±1.0	79.1	77.2	78.0
Waveform-40				
300	81.3±0.9	83.0	82.5	82.3
100	73.4±1.0	**78.4**	**76.7**	**78.6**

T1: This trend explores an explanation for the poor performance of the preset bias methods on the *sinusoidal* task. One form of pre-processing involves discretizing continuous features, which is required by the four preset bias algorithms (i.e., CCF, VDM, MVDM, and MI). Inspection of the discretizations computed by Fayyad and Irani's (1993) method revealed that features were improperly discretized for the *sinusoidal* task.[12] The comparatively

[12] The vertical, more relevant, dimension was generally split into only three intervals of which one interval covered nearly the entire range.

lower accuracies recorded by these four methods on this task (i.e., in comparison to k-NN$_{VSM}$) indicates their dependence on proper pre-processing methods, in this case discretization.

To test this hypothesis, we repeated our experiments for the *sinusoidal* task but manually provided the correct discretizations (i.e., 16 (4) equally sized intervals for the horizontal (vertical) feature, see Figure 6). The accuracies of all four preset bias methods then improved from at least three standard errors below k-NN's accuracy to at least three standard errors above (Table 6). Discretization had no positive effect on either k-NN$_{VSM}$'s or RELIEF-F's performance.

Table 6. Average accuracies for the *sinusoidal* task with Fayyad and Irani (1993) discretization intervals and manually assigned discretization intervals.

Discretization Method	Feature Weight Learning Algorithm					
	control	Performance bias method	Preset bias method			
	none	k-NN$_{VSM}$	CCF	VDM	MVDM	MI
none	74.2±0.7	**88.6**				
Irani and Fayyad			**65.1**	**65.1**	**65.0**	**69.6**
Manually			**81.6**	**82.8**	**83.3**	**88.5**

T2: The accuracy differences between the performance and preset bias methods for the *parity* task indicate that a main advantage of performance bias methods might be higher accuracy in the presence of interacting features. An additional experiment in a boolean task with ten input features, where the output was computed as the parity of a varying number of input features, supported this trend; k-NN$_{VSM}$ substantially outperformed MI when using two to four parity features. For larger numbers of interacting features, both algorithms had difficulty learning the concept given the small training set employed (i.e., 50 training examples).

T3: The experiment described in Table 5 showed that, for one task, feature weighting methods may have a substantially higher learning rate than k-NN. An issue closely related to this is whether different weighting methods have different learning rates. In particular, we hypothesize that performance bias methods have faster learning rates than preset bias methods. We selected the two most computationally efficient methods from each of these categories (i.e., MI and RELIEF-F) to investigate this hypothesis. We selected three tasks where these two methods achieved approximately equal accuracies as reported in Table 4. Results from these experiments indicate that the generalization accuracy of RELIEF-F is indeed higher than MI's for small training sets (Figure 3).

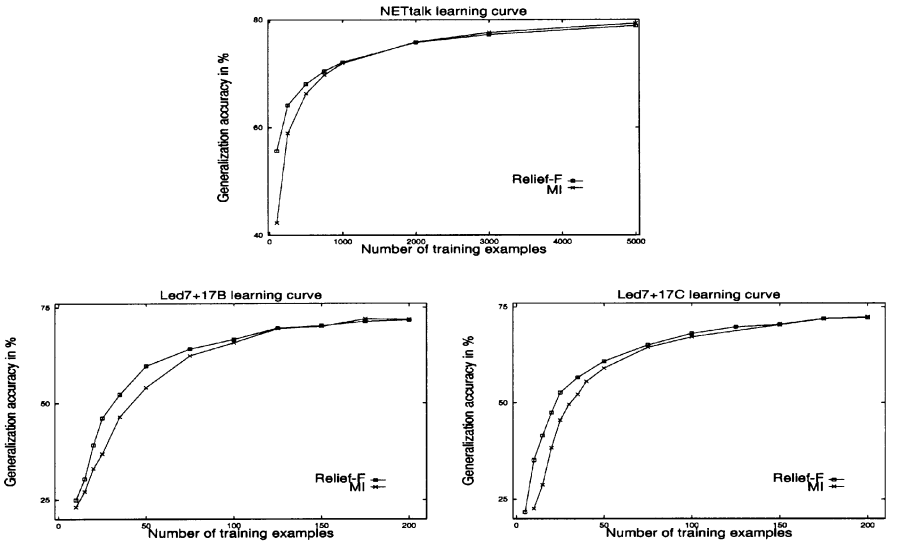

Figure 3. Learning curves for MI and RELIEF-F from three tasks.

T4: The second dimension for distinguishing feature weighting methods as described in Table 1 is the weight space employed by the algorithms. Kohavi et al. (1995) describe evidence that feature weighting methods lead to superior performance as compared to feature selection methods for tasks where some features are useful but less important than others.

We investigated this hypothesis by examining the performance of RELIEF-F and MI with continuous vs. binary weight settings (i.e., feature selection), where we removed an increasing number of lowest-weighted features. They were applied to two learning tasks: Waveform-40 (19 irrelevant, 21 relevant features, where feature relevance varies) and LED-7+17B (17 irrelevant features, seven others with approximately equal relevance). Figure 4 displays the results for RELIEF-F. Similar results were achieved for MI (not shown). For both tasks, the weighted approach is generally superior to the feature selection approach. The only improvement achieved by feature selection methods was for the LED-7+17B task when all 17 irrelevant features were removed. These results provide further support for **T4**. Our evidence suggests that, despite searching a larger weight space, feature weighting methods may outperform feature selection methods even in domains that are thought to be most suited to feature selection methods, i.e., domains that contain either approximately equally relevant or completely irrelevant features. This claim might hold unless the correct subset of (relevant) features is located by the feature selection methods. Additional research is needed to investigate this claim.

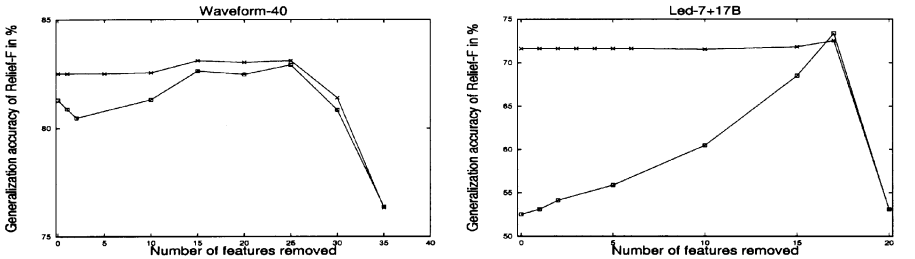

Figure 4. The performance of RELIEF-F when an increasing number of features with the lowest computed weights are removed. The remaining features are either continuously weighted (×) or assigned a weight value of 1 (□).

4.4.1. *Intra-model comparisons*

This section addresses differences in performance within the families of performance feedback and preset bias methods.

Performance bias methods

RELIEF-F and k-NN$_{VSM}$ had substantial performance differences on several datasets. We suspect that one cause is that k-NN$_{VSM}$ appears to be insufficiently biased towards giving zero weight to irrelevant features. We tested this hypothesis by varying the number of irrelevant features in the LED-7 task, and our results supports this claim. When the number of irrelevant features is increased from zero to 17, k-NN$_{VSM}$'s performance is initially superior, but RELIEF-F outperforms k-NN$_{VSM}$ when there are more than two irrelevant features. Furthermore, several experiments showed that RELIEF-F has a faster learning rate than k-NN$_{VSM}$.

Preset bias methods

VDM, MVDM, and CCF performed poorly compared with MI on the Waveform and *sinusoidal* tasks. A possible cause is that these methods retain the discretization of continuous features even after feature weights are learned. We tested this claim using the Waveform tasks. When feature values are left discretized for MI, even after feature weights are computed, its performance drops by 6.0% and 3.8% for the Waveform-21 and Waveform-40 tasks, respectively. This indicates that a substantial amount of information is lost when these datasets are discretized. Ting (1994) reported that discretization in lazy learning algorithms can be useful for noisy tasks. Further research is needed to explain why discretization does not improve performance for these noisy Waveform tasks.

A comparison of VDM and MVDM's accuracies in Table 4 reveals that the feature weights computed by Equation 19 have no beneficial effect on VDM's performance. Furthermore, feature weights computed via mutual information (Equation 21) did not improve MVDM'S performance for the *banded*, *sinusoidal*, and Waveform tasks. The MVDM's good performance for the *banded*, Led-7+17B, and Led-7+17C tasks indicates that the class projection method is an alternative to using feature weights when irrelevant features are present and continuous features are properly discretized. MVDM recorded the best result for the NETtalk task, which is not surprising since the VDM was designed for this task.

4.4.2. *Summary*

In summary, these results provide strong but sometimes incomplete evidence for our four trends. For example, our evidence for trend **T1** requires additional investigations with other forms of pre-processing steps that could decrease the performance of the preset bias methods (e.g., normalization).

Our trends were designed to help determine which algorithms should perform well for some given task characteristics. Specifically, these trends predict the behavior of several feature weight learning algorithms – in the presence of irrelevant, redundant or interacting features, or features with varying relevance – according to their categorization in our framework in Section 3.

5. Discussion and Implications

The dimensional framework in Section 3 can be used to relate different methods for weighting features, motivate experiments to distinguish their comparative abilities, and suggest future work (e.g., combining multiple approaches, such as a feature selection followed by a weighting algorithm). For example, new weighting methods could be categorized according to this framework, which could simplify their comprehension and provide a context for understanding their comparative abilities.

We also used this framework to organize our investigation in Section 4, whose purpose was to investigate the comparative abilities of feature weighting methods for a subclass of lazy algorithms with respect to two of this framework's dimensions. However, instead of simply reporting case study results (i.e., from applying each algorithm to each dataset), we also introduced and briefly evaluated trends that attempt to explain these results.

First, we introduced trends **T1**, **T2**, and **T3** to address the first dimension. That is, they concern the distinctive capabilities of performance and preset bias methods. Although we provide some evidence for them, they are limited

to the selected algorithms and datasets. For example, trend **T2** suggests performance biases are preferred for tasks with interacting features. This trend could be invalidated by methods that use preset biases designed to account for such features (e.g., a mutual information method that considers all combinations of features). Thus, in situations when a preset bias might otherwise be preferable (e.g., for computational reasons), yet the task is known to involve interacting features with high probability, steps can be taken to either modify the dataset's characteristics through representation change or on designing a preset bias method that tolerates such characteristics. The initial studies described in Section 4.4.1 likewise require further investigation since their implications might also be limited to the algorithms and datasets involved.

Second, trend **T4** addresses the second dimension of our framework defined in Section 3. Like Kohavi et al. (1995), we found some evidence that weighting is preferable to binary selection in some tasks where feature relevance varies. This provides motivation for designing algorithms that explore the tradeoffs of searching larger weight spaces, as done by continuous weighting algorithms, versus the computational efficiency gained by reducing the size of this space.

Finally, our framework can be used to suggest and direct future research efforts. For example, we envision algorithm designs that profitably exploit aspects of multiple framework dimensions. Algorithms could be designed to address the space/time tradeoff mentioned above by locally estimating where continuous weighting could be profitably explored. Larger weight spaces could then be searched for those regions of instance space. Alternatively, this could provide a focus for extracting domain-specific knowledge (i.e., for those local regions) (e.g., Domingos 1996), or for applying local feature transformation methods so as to reduce the size of the weight space (e.g., Hastie and Tibshirani 1994).

As another example, comparative evaluations could be focussed according to the sub-category structure. Our trends did not address differences *within* sub-categories of the framework (i.e., a comparison of two types of preset bias methods), although we briefly addressed these issues in Section 4.4.1. More detailed studies of this nature, which are strongly suggested by our framework, are left for future research.

6. Related Work

6.1. *Similar Studies*

Several studies have introduced feature weighting algorithms. Some compared new algorithms to unweighted k-NN (e.g., Kelly and Davis 1991; Aha 1992). Some studies have also compared weighting algorithms in a specific

context. For example, Wettschereck and Dietterich (1995) showed that a mutual information method traded off higher computational complexity for higher accuracies when compared with an online algorithm in the context of learning hyperrectangles (Salzberg 1991). Mohri and Tanaka (1994) reported a more extensive comparison, in which they review several feature weighting algorithms while motivating the introduction of QM2. This algorithm assigns weight values by optimizing specific statistical criteria. They reported that two lazy variants of QM2 attain good results in comparison with four other feature weighting algorithms, although they (also) have higher computational costs.

6.2. *Performance vs. Preset Biases*

Doak (1992), among others, noted the utility of using the classifier to guide feature selection. John et al. (1994) clarified this distinction. Aha and Bankert (1994), among others, subsequently provided additional empirical evidence for preferring performance feedback biases, and Kohavi et al. (1995) advocated using performance biases for continuous weighting methods.

Kohonen's (1990) *learning vector quantization* (LVQ) algorithm employs a hill-climbing performance bias method to cluster instances defined by continuous features. Although no current LVQ variant explicitly computes weights for input features, one could argue that these algorithms implicitly learn the relevance of features. Wettschereck and Dietterich (1992) showed how LVQ-type algorithms can be used to adjust the coordinates of irrelevant input features for all stored exemplars such that they are identical, which effectively eliminates them. They showed this for generalized radial basis networks (Poggio and Girosi 1990), where the centers of basis functions are moved during training. After training, the mean distances between the center locations' features reflected the relevance of the original input features.

6.3. *Information Theory*

In Section 3.1.2 we described feature weighting algorithms that use preset biases based on information theory. Several other learning algorithms have a similar basis. Quinlan (1986) used an information gain measure to select features when inducing decision trees in ID3. Wolpert (1990, 1994) used an information theoretic approach to set feature weights for a four-nearest neighbor algorithm and reported favorable results in comparison with Back-propagation on a word pronunciation task. Bakiri (1991) employed a modification of a MI weighting procedure proposed by Lucassen and Mercer (1984) that ranked features. This ranking was then used to determine which ones were used to induce decision trees.

The MI approach described in Section 3.1.2 assumes that features are independent (i.e., in their correlation with class). This can lead to inferior performance for tasks with interacting or redundant features. Battiti (1994) describes an approach that addresses this problem by first computing the MI between each pair of features and then decreasing the weight of highly interacting features.

Another concern is that the MI computed for many-valued features will frequently be larger than the MI of features with few distinct values, even if both features carry the same amount of information. This can be counteracted by normalizing each feature's MI value by a function of its number of possible values. However, this can still lead to sub-optimal behavior, such as when two continuous features differ greatly in relevance, yet the more relevant feature is discretized into a much larger number of intervals.

6.4. *Instance Weighting*

The topic of this paper concerns methods that set parameter values (i.e., feature weights) in the distance functions in a subclass of lazy learning algorithms. An alternative and frequently used approach for enhancing distance functions involves assigning weights to instances themselves. Weights can be assigned either before computing distances (e.g., Salzberg 1991; Aha et al. 1991) or afterwards (e.g., Connell and Utgoff 1987; Atkeson 1989). Both approaches bias the prediction of lazy algorithms by emphasizing the contributions of some instances over others. Wettschereck (1995b) described evidence that non-equal instance weights are often preferable. Atkeson et al. (1996a) survey alternative kernel functions, which effectively modify instance weightings. We expect that future research will determine which feature weighting methods are also useful for weighting instances.

6.5. *Alternative Architectures for Lazy Algorithms*

Finally, we note that several lazy learning algorithms have been implemented using alternative computational architectures. For example, connectionist network architectures have been used to implement many lazy learning algorithms. Volper and Hampson (1987) were early advocates of using specific instance information in such networks. Radial basis networks (e.g., Poggio and Girosi 1990; Broomhead and Lowe 1988) are closely related to lazy learning algorithms that cache weight settings; they replace a sigmoidal squashing function with a Gaussian whose activation is a function of its distance to the inputs. Kruschke (1992), among others, reported that his modified radial basis network correlates significantly well with a surprising amount of subject data collected over several decades. Carpenter et al. (1992) modified and

implemented Salzberg's (1991) EACH algorithm as a connectionist network. Several of these systems use some form of feature weighting that could be more closely compared with the algorithms reviewed here.

Several other algorithms blur the distinction between lazy and eager processing. For example, some incremental decision tree induction algorithms retain specific instances (e.g., Utgoff 1989), and some algorithms combine rules with specific instances to represent concepts (e.g., Zhang 1990). Each architecture highlights a unique perspective on weighting features, which may provide insights not easily obtained when using a traditional k-NN architecture.

7. Conclusions

In this paper we investigated issues on estimating feature weight parameters for the distance functions in a subclass of lazy learning algorithms. We reviewed several such feature weighting methods, outlined a framework composed of five dimensions for distinguishing them, and described example algorithms defined by each dimension. Our empirical evaluation compared several such algorithms, and suggested several trends. We described additional supporting evidence for each trend.

These trends suggest certain directions for future research. For example, since most of the algorithms tested successfully assign low weights to irrelevant features, empirical demonstrations of new feature weighting algorithms in the presence of irrelevant features are not particularly valuable. Instead, they should be compared with existing feature weighting algorithms, where the framework used in our review can be used to both categorize the new algorithm and motivate the selection of algorithms in empirical comparisons. Alternatively, researchers could use some of these trends to motivate the design of algorithms that contradict them (i.e., by avoiding some problems with their predecessors). Finally, we investigated only the first two dimensions of our framework, and did not address function learning tasks. This suggests additional research directions based on this framework.

This article focussed on empirical evaluations. Although several mathematical analyses exist for lazy learning algorithms (e.g., Cover and Hart 1967; Langley and Iba 1993) few address feature weighting (e.g., Satoh and Okamoto 1994; Ling and Wang 1996). Therefore, a theoretical analysis of these algorithms would provide a valuable companion for this article.

Acknowledgements

Thanks to our reviewers, whose comments helped us to greatly improve this article. We thank Igor Kononenko for providing the source code for Relief-F. Thanks also to Gert Durieux and Steven Gillis for testing whether the difference in performance between MVDM and MVDM with MI was statistically significant. Finally, we thank Patrick Murphy for maintaining the UCI Repository of ML Databases and Robert Detrano for making available the datasets on heart disease diagnoses.

References

Aha, D. W. (1991). Incremental constructive induction: An instance-based approach. In *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 117–121. Evanston, IL: Morgan Kaufmann.

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

Aha, D. W. & Bankert, R. L. (1994). Feature selection for case-based classification of cloud types: An empirical comparison. In D. W. Aha (ed.) *Case-Based Reasoning: Papers from the 1994 Workshop* (Technical Report WS-94-01). Menlo Park, CA: AAAI Press.

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

Aha, D. W., Kibler, D. & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, **6**: 37–66.

Ashley, K. D. & Rissland, E. L. (1988). Waiting on weighting: A symbolic least commitment approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 239-244. St. Paul, MN: Morgan Kaufmann.

Atkeson, C. (1989). Using local models to control movement. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*. San Mateo, CA: Morgan Kaufmann.

Atkeson, C., Moore, A. & Schaal, S. (1996a). Locally weighted learning. *Artificial Intelligence Review*, this issue.

Atkeson, C., Moore, A. & Schaal, S. (1996b). Locally weighted learning for control. *Artificial Intelligence Review*, this issue.

Bakiri, G. (1991). *Converting English text to speech: A machine learning approach.* Doctoral dissertation, Department of Computer Science, Oregon State University, Corvallis, OR.

Bareiss, R. (1989). The experimental evaluation of a case-based learning apprentice. In *Proceedings of a Case-Based Reasoning Workshop*, pp. 162–167. Pensacola Beach, FL: Morgan Kaufmann.

Battiti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks* **5**: 537–550.

Biberman, Y. (1994). A context similarity measure. In *Proceedings of the European Conference on Machine Learning*, pp. 49–63. Catania, Italy: Springer-Verlag.

Bottou, L. & Vapnik, V. (1992). Local learning algorithms. *Neural Computation* **4**: 888–900.

Bounds, D., Lloyd, P. & Mathew, B. (1990). A comparison of neural network and other pattern recognition approaches to the diagnosis of low back disorders. *Neural Networks* **3**: 583–591.

Broomhead, D. S. & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems* **2**: 321–355.

Cain, T., Pazzani, M. J. & Silverstein, G. (1991). Using domain knowledge to influence similarity judgement. In *Proceedings of the Case-Based Reasoning Workshop*, pp. 191–202. Washington, DC: Morgan Kaufmann.

Cardie, C. (1993). Using decision trees to improve case-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 25–32. Amherst, MA: Morgan Kaufmann.

Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H. & Rosen, D.B. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks* **3**: 693–713.

Cleveland, W. S. & Loader, C. (1994). *Computational methods for local regression* (Technical Report 11). Murray Hill, NJ: AT&T Bell Laboratories, Statistics Department. Available by FTP from netlib.att.com in /netlib/att/stat/doc/94/11.ps.

Connell, M. E. & Utgoff, P. E. (1987). Learning to control a dynamic physical system. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pp. 456–460. Seattle, WA: Morgan Kaufmann.

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

Cover, T. M. & Hart, P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory* **13**: 21–27.

Cover, T. M. & Thomas, J. (1991). *Elements of Information Theory.* New York: John Wiley and Sons.

Cover, T. M. & van Campenhout, J. M. (1977). On the possible orderings in the measurement selection problem. *IEEE Transactions on Systems, Man, and Cybernetics* **7**: 657–661.

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

Daelemans, W., Gills, S. & Durieux, G. (1993). *Learnability and markedness in data-driven acquisition of stress* (Technical Report 43). Tilburg, Netherlands: Tilburg University, Institute for Language Technology and Artificial Intelligence.

Daelemans, W., van den Bosch, A. (1992). Generalization performance of backpropagation learning on a syllabification task. In *Proceedings of TWLT3: Connectionism and Natural Language Processing*, pp. 27–37. Enschede, The Netherlands: Unpublished.

Dasarathy, B. V. (Ed.). (1991). *Nearest neighbor(NN) norms: NN pattern classification techniques.* Los Alamitos, CA: IEEE Computer Society Press.

Doak, J. (1992). *An evaluation of feature selection methods and their application to computer security* (Technical Report CSE-92-18). Davis, CA: University of California, Department of Computer Science.

Domingos, P. (1996). Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, this issue.

Duda, R. O. & Hart, P. E. (1973). *Pattern classification and scene analysis.* New York, NY: Wiley.

Dudani, S. (1975). The distance-weighted k-nearest neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics* **6**: 325–327.

Fayyad, U. M., & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1022–1029. Chambery, France: Morgan Kaufmann.

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

Fu, K. S. (1968). *Sequential methods in pattern recognition and machine learning.* New York: Academic Press.

Fukunaga, K. & Flick, T. (1982). A parametrically-defined nearest neighbor distance measure. *Pattern Recognition Letters* **1**: 3–5.

Fukunaga, K. & Flick, T. (1984). An optimal global nearest neighbor metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**: 314–318.

Gorman, R. & Sejnowski, T. (1988). Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks* **1**: 75–89.

Hastie, T. J. & 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.

Hayashi, C. (1952). On the prediction of phenomena from qualitative data and the quantification of qualitative data from the mathematical-statistical point of view. *Annals of the Institute of Statistical Mathematics* **3**: 69–98.

John, G., Kohavi, R. & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Machine Learning Conference*, pp. 121–129. New Brunswick, NJ: Morgan Kaufmann.

Kawaguchi, M. (1978). *Introduction to Multivariate Analysis II* (in Japanese). Morikita-Shuppan.

Kelly, J. D., Jr. & Davis, L. (1991). A hybrid genetic algorithm for classification. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 645–650. Sydney, Australia: Morgan Kaufmann.

Kibler, D. & Aha, D. W. (1987). Learning representative exemplars of concepts: An initial case study. In *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 24–30. Irvine, CA: Morgan Kaufmann.

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

Kohavi, R., Langley, P. & Yun, Y. (1995). Heuristic search for feature weights in instance-based learning. Manuscript submitted for publication.

Kohonen, T., Barna, G. & Chrisley, R. (1988). Statistical pattern recognition with neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pp. 61–88. IEEE Press.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE* **78**: 1464–1480.

Kolodner, J. (1993). *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of RELIEF. In *Proceedings of the 1994 European Conference on Machine Learning*, pp. 171–182. Catania, Italy: Springer Verlag.

Kruschke, J. K. (1992). ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review* **99**: 22–44.

Kshirsager, A. (1972). *Multivariate Analysis.* New York: Dekker.

Langley, P. & Iba, W. (1993). Average-case analysis of a nearest neighbor algorithm. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 889–894. Chambery, France: Morgan Kaufmann.

Langley, P. & Sage, S. (1994). Oblivious decision trees and abstract cases. In D. W. Aha (ed.), *Case-Based Reasoning: Papers from the 1994 Workshop* (Technical Report WS-94-01). Menlo Park, CA: AAAI Press.

Ling, X. C. & Wang, H. (1996). Towards optimal weights setting for the 1-nearest neighbour learning algorithm. *Artificial Intelligence Review*, this issue.

Lowe, D. (1995). Similarity metric learning for a variable-kernel classifier. *Neural Computation* **7**: 72–85.

Lucassen, J. & Mercer, R. (1984). An information theoretic approach to the automatic determination of phonemic base forms. In *Proceedings of the International Conference on Acoustics Speech Signal Processing* (42.5.1-42.5.4).

Luce, R. D. (1963). Detection and recognition. In R. D. Luce, R .R. Bush & E. Galanger (eds.), *Handbook of mathematical psychology.* New York, NY: Wiley.

McGill, W. (1955). Multivariate information transmission. *IEEE Transactions on Information Theory* **1**: 93–111.

Medin, D. L. & Schaffer, M. M. (1978). Context theory of classification learning. *Psychological Review* **85**: 207–238.

Michie, D., Spiegelhalter, D. J. & Taylor, C. C. (Eds.) (1994). *Machine learning, neural and statistical classification.* London: Prentice Hall.

Mitchell, T. M. (1990). The need for biases in learning generalizations. In J. W. Shavlik & T. G. Dietterich (eds.), *Readings in machine learning*. San Mateo, CA: Morgan Kaufmann.

Mitchell, T., Keller, R. & Kedar-Cabelli, S. (1986). Explanation-based learning: A unifying view. *Machine Learning* **1**: 47–80.

Mohri, T., Nakamura, M. & Tanaka, H. (1993). Weather forecasting using memory-based reasoning. In *Second International Workshop on Parallel Processing for Artificial Intelligence*, pp. 40–45.

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

Mohri, T. & Tanaka, H. (1995). Comparison between attribute weighting methods in memory-based reasoning and multivariate analysis. Manuscript submitted for publication.

Moore, A. W. & Lee, M. S. (1994). Efficient algorithms for minimizing cross validation error. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 190–198. New Brunswick, NJ: Morgan Kaufmann.

Mucciardi, A. N. & Gose, E. E. (1971). A comparison of seven techniques for choosing subsets of pattern recognition properties. *IEEE Transaction on Computers* **20**: 1023–1031.

Murphy, P. (1995). *UCI Repository of machine learning databases* [Machine-readable data repository @ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science.

Myles, J. & Hand, D. (1990). The multi-class metric problem in nearest neighbor discrimination rules. *Pattern Recognition* **23**: 1291–1297.

Nadaraya, E. A. (1964), On estimating regression. *Theory of Probability and its Applications* **9**: 141–142.

Poggio, T. & Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science* **247**: 978–982.

Porter, B. W., Bareiss, R. & Holte, R. C. (1990). Knowledge acquisition and heuristic classification in weak-theory domains. *Artificial Intelligence* **45**: 229–263.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. & Flannery, B. P. (1992). *Numerical Recipes in C*. Cambridge, UK: Cambridge University Press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning* **1**: 81–106.

Ricci, F. & Avesani, P. (1995). Learning a local similarity metric for case-based reasoning. In *Proceedings of the First International Conference on Case-Based Reasoning*, pp. 301–312. Sesimbra, Portugal: Springer-Verlag.

Salzberg, S. L. (1991). A nearest hyperrectangle learning method. *Machine Learning* **6**: 251–276.

Satoh, K. & Okamoto, S. (1994). Toward PAC-learning of weights from qualitative distance information. In D. W. Aha (ed.) *Case-Based Reasoning: Papers from the 1994 Workshop* (Technical Report WS-94-01). Menlo Park, CA: AAAI Press.

Schaffer, C. (1994). A conservation law for generalization performance. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 259–265. New Brunswick, NJ: Morgan Kaufmann.

Schlimmer, J. C. (1987). Incremental adjustment of representations for learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 79–90. Irvine, CA: Morgan Kaufmann.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technology Journal* **27**: 379–423.

Short, R. & Fukunaga, K. (1980). A new nearest neighbor distance measure. In *Proceedings of the Fifth International Conference on Pattern Recognition*, pp. 81–86. Los Alamitos, CA: IEEE Press.

Short, R. & Fukunaga, K. (1981). The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory* **27**: 622–627.

Simard, P., Le Cun, Y. & Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In Hanson, S. J., et al. (eds.), *Advances in Neural Information Processing Systems 5*. San Mateo, CA: Morgan Kaufmann.

Skalak, D. (1992). Representing cases as knowledge sources that apply local similarity metrics. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, pp. 325–330. Bloomington, IN: Lawrence Erlbaum.

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Machine Learning Conference*, pp. 293–301. New Brunswick, NJ: Morgan Kaufmann.

Stanfill, C. & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the Association for Computing Machinery* **29**: 1213–1228.

Tan, M. (1993). Cost-sensitive learning of classification knowledge and its application in robotics. *Machine Learning* **13**: 7–34.

Ting, K. M. (1994). *Discretization of continuous-valued attributes and instance-based learning* (Technical Report). Sydney, Australia, University of Sydney, Basser Department of Computer Science.

Turney, P. D. (1993). Exploiting context when learning to classify. In *Proceedings of the European Conference on Machine Learning*, pp. 402–407. Vienna, Austria: Springer-Verlag.

Turney, P. D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research* **2**: 369–409.

Tversky, A. (1977). Features of similarity. *Psychological Review* **84**: 327–352.

Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning* **4**: 161–186.

van den Bosch, A. & Daelemans, W. (1993). *Data-oriented methods for grapheme-to-phoneme conversion* (Technical Report 42). Tilburg, Netherlands: Tilburg University, Institute for Language Technology and Artificial Intelligence.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In R. P. Lippmann & J. E. Moody (eds.), *Advances in Neural Information Processing Systems 3*. Denver, CO: Morgan Kaufmann.

Volper, D. J. & Hampson, S. E. (1987). Learning and using specific instances. *Biological Cybernetics* **57**: 57–71.

Weiss, S. M. & Kapouleas, I. (1989). An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 781–787. Detroit, MI: Morgan Kaufmann.

Weiss, S. M. & Kulikowski, C. A. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems.* San Mateo, CA: Morgan Kaufmann.

Wettschereck, D. (1994). *A study of distance-based machine learning algorithms.* Doctoral dissertation, Department of Computer Science, Oregon State University, Corvallis, OR. Available via WWW at http://nathan.gmd.de/persons/dietrich.wettschereck.html

Wettschereck, D. (1995a). *A description of the mutual information approach and the variable similiarty metric* (Technical Report 944). Sankt Augustin, Germany, German National Research Center for Computer Science, Artificial Intelligence Research Division.

Wettschereck, D. (1995b). *Weighted kNN versus majority kNN: A recommendation* (Technical Report 943). Sankt Augustin, Germany, German National Research Center for Computer Science, Artificial Intelligence Research Division.

Wettschereck, D. & Dietterich, T. G. (1992). Improving the performance of radial basis function networks by learning center locations. In J. Moody, S. Hanson, & R. Lippmann (eds.), *Neural Information Processing Systems 4*. Denver, CO: Morgan Kaufmann.

Wettschereck, D. & Dietterich, T. G. (1995). An experimental comparison of the nearest neighbor and nearest hyperrectangle algorithms. *Machine Learning* **19**: 5–28.

Wolpert, D. H. (1990). Constructing a generalizer superior to NETtalk via a mathematical theory of generalization. *Neural Networks* **3**: 445–452.

Wolpert, D. H. (1994). Personal communication.

Yau, H. C. & Manry, M. T. (1991). Iterative improvement of a nearest neighbor classifier. *Neural Networks* **4**: 517–524.

Zhang, J. (1990). A method that combines inductive learning with exemplar-based learning. In *Proceedings for Tools for Artificial Intelligence*, pp. 31–37. Herndon, VA: IEEE Computer Society Press.