



Article

Locally-Scaled Kernels and Confidence Voting

Elizabeth Hofer [†] and Martin v. Mohrenschildt ^{*,†} Department of Computing and Software, McMaster University, Hamilton, ON L8S 4L8, Canada;
hoferel@mcmaster.ca

* Correspondence: mohrens@mcmaster.ca

[†] These authors contributed equally to this work.

Abstract: Classification, the task of discerning the class of an unlabeled data point using information from a set of labeled data points, is a well-studied area of machine learning with a variety of approaches. Many of these approaches are closely linked to the selection of metrics or the generalizing of similarities defined by kernels. These metrics or similarity measures often require their parameters to be tuned in order to achieve the highest accuracy for each dataset. For example, an extensive search is required to determine the value of K or the choice of distance metric in K -NN classification. This paper explores a method of kernel construction that when used in classification performs consistently over a variety of datasets and does not require the parameters to be tuned. Inspired by dimensionality reduction techniques (DRT), we construct a kernel-based similarity measure that captures the topological structure of the data. This work compares the accuracy of K -NN classifiers, computed with specific operating parameters that obtain the highest accuracy per dataset, to a single trial of the here-proposed kernel classifier with no specialized parameters on standard benchmark sets. The here-proposed kernel used with simple classifiers has comparable accuracy to the ‘best-case’ K -NN classifiers without requiring the tuning of operating parameters.

Keywords: classification; similarity measure; kernel methods; KNN



Citation: Hofer, E.; v. Mohrenschildt, M. Locally-Scaled Kernels and Confidence Voting. *Mach. Learn. Knowl. Extr.* **2024**, *6*, 1126–1144.
<https://doi.org/10.3390/make6020052>

Academic Editor: Massimo Ferri

Received: 10 April 2024

Revised: 16 May 2024

Accepted: 17 May 2024

Published: 23 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Discerning the label y_* for an unlabeled data point x_* using information from data points with known labels (x_j, y_j) is a question fundamentally based on ‘similarity’. It is expected that points of the same class are similar, so it is reasonable to map similar data points to the same class; thus, the label of x_* could be determined using the class information of similar labeled data points. One way to interpret similarity is through a distance metric; if two points are ‘close’ by some distance metric then they are similar, and if they are far, then they are dissimilar. This leads to the concept of the K -nearest neighbors (K -NN) and the K -NN classifier.

The choice of distance metric has a significant impact on the accuracy of traditional K -NN classifiers. This impact has been explored in literature with [1–6] reporting the success of a variety of distance metrics used with a K -NN classifier on diverse sets of high-dimensional datasets. No singular distance metric performs superiorly across all datasets, but rather the best-performing distance metric is specific to each dataset.

The problem statement is thus, when presented with a labeled dataset $(x_j, y_j) \in \mathcal{X}$ ($|\mathcal{X}| = J$) where $x_j \in \mathbb{R}^N$ with associated class label $y_j \in \{1, \dots, C\}$, construct a function $\text{class}(x_j)$ that maps x_j to its label y_j . Using this, the class of a ‘new’ unlabeled point x_* can be discerned. This function should perform universally well across all datasets without requiring the tuning of its operating parameters.

A distance metric that, when used in classification, performs universally well across all datasets should adjust to the data on a local scale. Ideally, the distance between data points can be measured according to the topological structure of the data. From our previous experience in data mining, we understand the importance of feature extraction in

classification [7]—even the best classifier will be unable to make meaningful classifications if the features of the data are indistinguishable. No distance metric can measure meaningful similarity if the feature space of the data does not separate data in a meaningful way. In machine learning, kernels are a tool used to pull the data to a new feature space called $\mathcal{H}$; this space is designed to better capture the similarities between points of the same class and vice versa [8]. Conveniently, the inner product (which induces a distance metric) in this space is a measure of similarity [9], making the kernel a natural extension of distance-based methods of classification.

Many approaches to classification only depend on the pairwise distance/similarity/correlation of the input data. As such, these algorithms can be *kernelized*, meaning the algorithm operates on the values of the kernel. This is further motivation for the use of a kernel to map to a new feature space that better captures similarity. Furthermore, kernel methods provide an advantage when working with high dimensional datasets as the Gram kernel matrix size is proportional to the number of data points J —once it is computed we no longer have to perform any calculations in the space of $\mathbb{R}^N$.

The success of traditional and weighted K -NN classifiers also depends on how confident x_i is of its label y_i . It can be challenging to discern the label of x_* using information from the surrounding neighbors when the neighbors are outliers or mislabeled data points, not uncommon in real-world datasets [10]. Thus, a secondary goal is to design a method of classification that diverges from the traditional K -NN classifier and introduces some measure of confidence to each x_j . This allows a trade-off between supervised classification (i.e., classification based on the known labels) and semi-supervised classification (i.e., classification based on some inferred class-structure of the data).

To evaluate the proposed approach, we perform tests using standard benchmark data sets from the University of California Irvine (UCI) Machine Learning Repository [11]. Using a leave-one-out testing strategy, we compare classifiers, based on the here-proposed similarity kernel with other weighted and non-weighted K -NN approaches. In contrast to approaches that study how to obtain the most accurate tuning for a classifier on a particular data set individually, we are interested in designing a similarity kernel that when used for classification requires one tuning that works well for many/all data sets.

2. Background

We begin by giving two motivations for the use of kernels in classification, one based on arguments generalizing K -NN, and the other based on mathematical arguments.

Classification has been a well-studied problem in the area of machine learning with myriad tools existing for a diverse array of applications. One popular classifier is called K -nearest neighbors [12], which uses the K nearest labeled points (by some distance metric d) of x_* to draw conclusions about the label of x_* . For ease of notation, $n_K(x_j, x_k)$ is a binary function that returns 1 if x_k is one of the K -nearest neighbors of x_j and 0 if it is not. A $J \times C$ matrix G called the ‘class matrix’ stores the class information of the labeled samples such that $G_{j,c} = 1$ if x_j belongs to class c and zero; otherwise, classification can be performed as follows:

$$\text{class}(x_*) = \operatorname{argmax}_c \sum_{j=1}^J G_{j,c} n_K(x_*, x_j) \quad (1)$$

A common choice for the distance metric is the Euclidean metric d_{EUC} . However, there are many possible choices of distance metrics for K -NN classification. An in-depth exploration of metrics can be found in the work of Abu Alfeilat et al. [1], which measures the classification accuracy, precision, and recall for a 1-NN classifier using 27 distance metrics from 13 distance metric families across 23 benchmark datasets from the UCI Machine Learning Repository [11]. From this study, it was apparent that the distance metric with the highest accuracy depends highly on the dataset, i.e., there is no single distance metric that proves to be the ‘best’ across all benchmark datasets. Some of the top-performing distance metrics from this work can be found below in Table 1.

Table 1. A selection of distance metrics.

Equation	Name
$d_{\text{EUC}}(\mathbf{x}_j, \mathbf{x}_k) = \sqrt{\sum_{n=1}^N (\mathbf{x}_{k,n} - \mathbf{x}_{j,n})^2}$	Euclidean
$d_{\text{HAS}}(\mathbf{x}_j, \mathbf{x}_k) = \sum_{n=1}^N \begin{cases} 1 - \frac{1 + \min(\mathbf{x}_{j,n}, \mathbf{x}_{k,n})}{1 + \max(\mathbf{x}_{j,n}, \mathbf{x}_{k,n})} & \min(\mathbf{x}_{j,n}, \mathbf{x}_{k,n}) \geq 0 \\ 1 - \frac{1 + \min(\mathbf{x}_{j,n}, \mathbf{x}_{k,n}) + \min(\mathbf{x}_{j,n}, \mathbf{x}_{k,n}) }{1 + \max(\mathbf{x}_{j,n}, \mathbf{x}_{k,n}) + \min(\mathbf{x}_{j,n}, \mathbf{x}_{k,n})} & \text{else} \end{cases}$	Hassanat
$d_{\text{COS}}(\mathbf{x}_j, \mathbf{x}_k) = 1 - \frac{2 \sum_{n=1}^N \mathbf{x}_{k,n} \mathbf{x}_{j,n}}{\sqrt{\sum_{n=1}^N \mathbf{x}_{k,n}^2} \sqrt{\sum_{n=1}^N \mathbf{x}_{j,n}^2}}$	Cosine
$d_{\text{MAN}}(\mathbf{x}_j, \mathbf{x}_k) = \sum_{n=1}^N \mathbf{x}_{k,n} - \mathbf{x}_{j,n} $	Manhattan

The paper concludes that the Hassanat distance [2] is the best-performing distance metric in general.

Notably, in the study by Abu Alfeilat et al., the authors only perform classification with a 1-NN classifier. This experimental design choice could have been influenced by the well-reported deterioration of K -NN classifiers as the number of neighbors increases [13]. For an unknown dataset, testing must be performed to find the most suitable distance metric and value of K . The authors were inspired to find a classifying technique that is metric-insensitive and stable with regard to operating parameters such as K .

Similar work in [9] found that computing the weights of a K -NN classifier on a local-scale can temper the classifier's sensitivity to K .

Aside from exploring the effect of choice of distance metrics on K -NN classifier accuracy, another method of improving classification accuracy concerns changing the influence of the neighbors' votes, following the idea that some neighbors might be more relevant to the classification of $\mathbf{x}_*$ and therefore their vote should be weighted more heavily. Thus, weights w are introduced. The choice of weights can be guided using either combinatorial considerations or using metric/topological arguments.

An optimal weighting scheme based on a combinatorial argument is presented by Samworth in [13]. This technique compares the risk of a classification using weights $w_{n,m}$ to the optimal Bayesian classifier, and uses an asymptotic expansion to derive an optimality criterion. Samworth first determines the optimal value for K^* -nearest neighbors based on the dimension of the data N and the number of samples J : $K^* = \lfloor B^* J^{4/(N+4)} \rfloor$. The weight for the k th nearest neighbor is then

$$w_k^* = \frac{1}{K^*} \left[1 + \frac{N}{2} - \frac{N}{2K^{*2/N}} \{k^{1+2/N} - (k-1)^{1+2/N}\} \right].$$

Note that this weighting, while proven combinatorially optimal, is not local to each neighborhood, as all neighborhoods use the same weight for their k th nearest neighbor.

Kernels can be used as a generalization of distance metrics. We are operating in an inner product space (Hilbert space), with some scalar product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. The scalar product is seen as a measure of similarity between two data points; it is 0 if the data points are dissimilar, usually called orthogonal, and it increases as the data 'aligns'. The scalar product induces a norm in this space as $\|\mathbf{x}\| = \langle \mathbf{x}, \mathbf{x} \rangle^{\frac{1}{2}}$, and the norm induces a distance metric $d(\mathbf{x}_j, \mathbf{x}_k) = \|\mathbf{x}_k - \mathbf{x}_j\|$. Now, given a possible non-linear feature map Φ that maps data into a new feature space, we define $\mathcal{K}(\mathbf{x}_j, \mathbf{x}_k) = \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_k) \rangle$ meaning the kernel gives the value of the scalar product. The mathematical machinery with the representation theorem and Mercer's theorem at the center is extensively studied in reference [8]. Conveniently, we only need to define a good kernel, and the similarity measure is implied. The task of choosing the correct kernel is still regarded as an open problem [14].

Weights chosen following a metric argument are computed from a function of the distance d_{jk} between $\mathbf{x}_j$ and its k th nearest neighbor $\mathbf{x}_k$, following the idea that points that measure closer to $\mathbf{x}_j$ should have a more influential vote. Some examples of weights $w_{j|k}$

that are associated with each j -neighbor- k pair are outlined in Table 2. The use of the $|$ symbol in $w_{j|k}$ is to emphasize that these weights are not symmetric, as they are all locally scaled in some way with respect to the defining point of the neighborhood x_j . The weights are applied in the classification function as follows:

$$\text{class}(x_*) = \operatorname{argmax}_c \sum_{j=1}^J G_{j,c} w_{*|j}. \quad (2)$$

Following the ideas of weights and kernels, we can construct a measure of similarity. Points that have a high measure of similarity are more likely to belong to the same class, and the inverse is true as well. To construct a measure of similarity, we look to dimensionality reduction algorithms such as UMAP [15], Laplacian Eigenmaps [16] and t-SNE [17]. These techniques are designed to capture a measure of similarity in high dimensional space and preserve that measure of similarity in an embedding of the data in a lower dimensional, primarily for the purpose of visualization. However, the measure of similarity can also be used for classification following the idea that similar data points should belong to the same class.

Table 2. A selection of K -NN weights.

Equation	Name
$w_{j k} = \frac{d_{j(K+1)} - d_{jk}}{d_{j(K+1)} - d_{j1}}$	WKNN [18]
$w_{j k} = \frac{d_{j(K+1)} - d_{jk}}{d_{j(K+1)} - d_{j1}} \times \frac{d_{j(K+1)} + d_{jk}}{d_{j(K+1)} + d_{j1}}$	DWKNN [19]
$w_{j k} = e^{-d_{jk}/d_{jK}}$	EXPKNN
$w_{j k} = e^{-(d_{jk}/d_{jK})^2}$	NORKNN

The ‘go-to’ choice for similarity measures in the literature is Gaussian measures [17,20]. Interestingly, the dimensionality reduction community (in techniques such as UMAP [15] and Laplacian Eigenmaps [16]) use Laplacian measures of the form $w_{j|k} = e^{ad(x_j, x_k)}$. These techniques are based on the manifold hypothesis [21], which assumes that the data are distributed on a manifold $(\mathcal{M}, g)$ embedded in the input space. Thus, the similarity measure—which measures similarity based on some distance—should use the *geodesic distance* g as it best represents the underlying structure of the data.

However, as the manifold is not explicitly defined, [15,17,22] explore different ways of approximating (or in the case of [15], capturing) that geodesic distance. Specifically, UMAP works on the assumption that the dataset is distributed uniformly upon a Riemannian manifold, allowing for the geodesic distance to be measured by the Euclidean distance in sufficiently small local neighborhoods. This allows UMAP to ‘capture’ the geodesic distance instead of approximating it.

Finally comes the question of evaluating the success of the classifier. When evaluating a method of classification with a benchmark dataset, one must partition the data into a training set and a testing set. The training set is used for training the classifier. The classifier will make predictions on the testing set, and those predictions are compared back to the ground truth labels. From this, metrics such as accuracy can be derived. In the literature, when documenting the efficacy of methods of classification, data are usually divided using a train–test partition, or the data are divided into n -folds where each fold takes a ‘turn’ being the test set and all other folds are combined to create the training set. In both these methods, the success metrics (e.g., accuracy) are averaged across all the folds, or in the case of the test–train partition, averaged across a few trials of random partitions. Some common train–test split ratios are 66:33 [1], 70:30 [2,3], and 80:20 [23], and some common folds are 5-fold [24] and 10-fold [5,6,25,26]. Regardless of the ratio of the partition, the randomness of how the data points are divided affects the performance

of the classifier, which impacts the repeatability of the experiment [27]. Abu Alfeilat et al. report the common performance measures of accuracy, precision, and recall. The method of measuring multi-class accuracy varies across different sources [28–30], which is reviewed in Section 5.2 with Equations (8) and (9). We have found that studies of classifiers on benchmark datasets do not always publish their method of measuring accuracy, and the ambiguity of the success metrics and train–test split creates a challenge when comparing the results from different researchers.

3. The Similarity Kernel

A kernel is a positive semi-definite mapping, specifically a similarity kernel $\mathcal{K}(x_j, x_k)$, which maps two elements of the dataset to a value between zero and one (inclusive); the closer the value is to one, the more similar the data points are considered. However unlike the weights from Table 2, by Mercer’s theorem [8] kernels are required to be symmetric, meaning $\mathcal{K}(x_j, x_k) = \mathcal{K}(x_k, x_j)$. Another way to view the kernel approach is to consider Hilbert spaces and metrics. A kernel is the generalized idea of a distance metric. Given a mapping Φ , mapping the input space $\mathcal{X}$ to a new space (often called the feature space $\mathcal{X}'$) one automatically obtains a similarity measure by defining $\mathcal{K}(x_j, x_k) = \langle \Phi(x_j), \Phi(x_k) \rangle$ by Mercer’s theorem.

By the representation theorem [8], any function can be expressed as a linear combination of the kernel function, and classification functions are no exception. Thus, the classifier $\text{class}(x_j)$ constructed using the similarity measurement is

$$\text{class}(x_*) = \text{argmax}_c \sum_{j=1}^J a_k \mathcal{K}(x_*, x_j).$$

The question then becomes: what kernel best captures the similarities (and dissimilarities) of the data’s classes?

3.1. Theoretical Framework—Preserving the Topology

Inspired by the work of dimensionality reduction techniques (DRT) such as UMAP [15], Laplacian Eigenmaps [16], and t-SNE [17], a similarity measure is designed to capture the local and global structure of the dataset, thereby preserving the topology of the data. This stands in contrast to much of the existing research, for example [13], which designs optimal weights for K -NN classifiers based on combinatorics, or [1,2], which select/design best-performing weights based on experimental success instead of using a mathematical framework for justifying their design choices. The similarity measures in UMAP [15] and Laplacian Eigenmaps [16] use a Laplacian distribution, in contrast to t-SNE [17], which uses a Gaussian distribution as a Laplacian distribution has a longer tail, which is necessary for capturing the distances between points in high-dimensional space [31]. The following describes a topological justification for the choice of similarity kernel $\mathcal{K}$, which preserves the topology of the dataset.

1. It is assumed the data points x_j are distributed uniformly on a Riemannian manifold $(\mathcal{M}, g)$ embedded in the vector space $\mathbb{R}^N$. This manifold represents the underlying natural structure of the data.

Then, naturally, the *geodesic* distance $g(x_j, x_m)$ along the manifold should be used to measure the closeness of neighbors, but g is not explicitly defined for real-world datasets. However, a Riemannian manifold is locally Euclidean. A lemma presented in [15] outlines a strategy for measuring geodesic distance on a ‘local neighborhood’ scale. The lemma states:

2. If g is locally constant in an open neighborhood U , then within a ball $\mathcal{B} \subseteq U$ with radius r of volume $\frac{\pi^{\frac{N}{2}} r^N}{\Gamma(\frac{N}{2}+1)}$ centered at point x_j the geodesic distance from x_j to any point $x_k \in \mathcal{B}_i$ is $\frac{1}{r} d(x_j, x_k)$ where d is the Euclidean distance in $\mathbb{R}^N$. This is because the geodesic distance in the neighborhood is bound by the Euclidean distance on the tangent plane at x_j .

Around each point x_j on the Riemannian manifold, there exists a tangent space that is spanned by the tangent vectors. The neighbors of x_j can be projected onto the tangent space. The geodesic distances between x_j and its neighbors are bound by the Euclidean metrics implied by the scalar product of the tangent vectors on the tangent space. If the data are contained on a Riemannian surface, the local neighborhood as defined by the geodesic distance is also the local neighborhood determined using Euclidean distance. Thus, for sufficiently small neighborhoods, the local metric can be captured by the Euclidean distance metric.

The question that then remains is how can this be used to compute similarity? The solution is to capture the topology of the data with a graph. The nodes of the graph represent the data points and the edges, weighted with neighborhood-local measurements measured using Lemma 1 from [15] and connecting neighbors together.

3. A simplex is a generalized triangle. From algebraic topology, it is known that a simplicial approximation of a manifold can be used to capture the topological aspects of that manifold [32].

We use the topology captured by the graph to make inferences about similarity based on the geodesic distance of the manifold. This follows the principle that the geodesic distance is the most intuitive metric to use when measuring similarity, allowing for the measure of similarity to be based on geodesic distance without explicitly knowing $g(x_j, x_m)$.

4. Similarity is a symmetric property.

The result is a non-directed graph that can be used as a tool to represent the topology of the dataset. A kernel that captures similarity based on the topology of the data can then be constructed from the graph, the technical details of which are outlined in Section 3.2.

3.2. The Locally Scaled Symmetric Laplacian Diffusion Kernel

Based on points 1, 2, and 3 above, a weight is first constructed. The weight is based on a Laplacian distribution that has been locally scaled between x_j and K of its nearest neighbors

$$w_{j|k}(x_j, x_k) = \begin{cases} e^{\frac{d_{j1}}{\sigma_j}} e^{-\frac{d_{jk}}{\sigma_j}} & \text{if } n_K(x_j, x_k) = 1. \\ 0 & \text{else} \end{cases} \quad (3)$$

where the local scaling factor $\exp(\frac{d_{j1}}{\sigma_j})$ is locally scaled to the neighborhood of x_j , using d_{j1} as the distance from x_j to its first nearest neighbor, which shifts the weights such that they all decay relative to the first-nearest-neighbor of the neighborhood, and σ_j is chosen such that $\sum_{k=1}^K w_{j|k} = \alpha$. Natural choices for the parameter α are 2, $\log_2(K)$, which is used by UMAP following combinatorial arguments, or $\sqrt{K}$. Table 7 in Section 6 quantifies the effects of α on classification accuracy. These weights are non-symmetric because the selection of K nearest neighbors is non-symmetric. The selection and order of the neighbors do not change by this local metric when compared to the Euclidean distance metric.

Furthermore, following point 4 above, the weights are symmetrized using

$$\hat{\mathcal{K}}_{jk} = w_{j|k} + w_{k|j} - w_{j|k}w_{k|j}.$$

Symmetrizing $w_{j|k}$ is necessary to obtain a proper similarity kernel $\hat{\mathcal{K}}_{jk}$. Note that this method of symmetrization follows fuzzy set intersection, as described in UMAP [15] and not the arithmetic mean like t-SNE [17]. Practically, this means that if $w_{i|j} = 1$ and $w_{j|i} = 0$, symmetrizing results in $\hat{\mathcal{K}}_{ij} = 1$. Without this, points in a tightly knit cluster might not be able to capture their location in the larger structure of the data, thereby losing important global structure information.

Finally, the weights are normalized with

$$\mathcal{K}_{jk} = \frac{\hat{\mathcal{K}}_{jk}}{\sum_{m=1}^K \hat{\mathcal{K}}_{jm}}.$$

A kernel matrix K is constructed where $K_{j,k} = \mathbb{K}_{jk}$ and $K_{j,j} = 1$. This is the locally scaled symmetric Laplacian diffusion (LSSLD) kernel $\mathbb{K}(x_j, x_k)$.

3.3. Kernel Computational Complexity

Let us point out that time complexity is not equal to the time required to execute code, but rather the number of steps needed to solve the problem. Regardless of whether one is computing the time complexity of a covariance matrix, distance matrix, or kernel, all combinatorial pairs need to be generated and all dimensions visited. This results in a complexity of $O(J^2N)$, where J is the number of data points and N is the dimension of the input space. Computing the LSSLD kernel requires a normalization for each data point, and for this, we use Newton's method, which is a constant time operation, to solve the non-linear equation.

4. Applications of Kernels in Classification

Many, if not most, of the commonly used machine learning algorithms (SVM, regression, ridge regression, random forest, etc.) can be kernelized; the algorithm takes as an input the values of the kernel evaluated on all pairs or mathematically, a dual space. We use the LSSLD kernel we defined in the previous section, Section 3.2, in the following methods of classification. Specifically, this research considers two methods of classification, the first is a blended-model of weighed K -NN and confidence voting (of which we consider three ratios of 'blending'), and the second is kernel ridge regression.

The simplest approach to classification is a weighted K -NN classifier that utilizes the LSSLD kernel as weights

$$\text{class}(x_*) = \operatorname{argmax}_c \sum_{j=1}^J G_{j,c} \mathbb{K}(x_*, x_j). \quad (4)$$

4.1. Confidence Voting and the Blended-Model

Going one step further, the kernel could be used to measure the confidence of a point x_j regarding its own class. Perhaps x_j is an outlier, this can be assessed by measuring the similarity of x_j to points of the same class. Confidence voting is closely linked with outlier detection [33]; if x_j is similar to points that mostly belong to a different class, x_j could be an outlier. To prevent misclassification of x_* , outliers should not be weighted heavily. Thus, confidences $c_{j,c}$ are assigned to each sample x_j where $c_{j,c} = \sum_{k=1}^J G_{k,c} \mathbb{K}(x_j, x_k)$, such that the confidence reflects how certain x_j is of belonging to class c . From this confidence matrix C , again a $J \times C$ matrix, can be constructed. Confidence voting can be used to curtail the negative effects from outliers with

$$\text{class}(x_*) = \operatorname{argmax}_c \sum_{j=1}^J C_{j,c} \mathbb{K}(x_*, x_j). \quad (5)$$

Confidence voting introduces a validation for each of the training points to determine if the labels implied from the similarity kernel are equivalent to the labels given in the training set, and the decision-making is based on these implied labels. As can be seen in Table 8, datasets known to contain outliers [34] (Heart, Wine, Balance, Australian, Ionosphere, Rice, Haberman) perform better with confidence voting.

A parameter β represents the 'trade-off' between the proportion of the classification that is based on $G_{j,c}$ (the known labels of the class, or ground truth) and the proportion of the classification that is based on $C_{j,c}$ (the underlying structure of the data), leading to blended-model voting

$$\text{class}(x_*) = \operatorname{argmax}_c \sum_{j=1}^J ((1 - \beta) G_{j,c} + \beta C_{j,c}) \mathbb{K}(x_*, x_j). \quad (6)$$

Note that when $\beta = 0$, the blended-model is equivalent to K -NN using the LSSLD weights, and when $\beta = 1$, the blended-model is equivalent to confidence voting.

4.2. Regularization

This leads to the concept of regularization, a method that trades off, which effectively prevents over-fitting. The similarity kernel matrix K that was defined in Section 3.2 can nicely be used in a regularization framework. A proposed regularization approach used in [35,36] based on kernel ridge regression (KRR) uses a locally scaled Gaussian kernel. The regularization framework in [35] learns a global classification as

$$\text{class}(\mathbf{x}_*) = \underset{y_*}{\operatorname{argmin}} (1 - \gamma)(\mathbf{I} - \gamma\mathbf{K})^{-1}\mathbf{G} \quad (7)$$

where $\mathbf{I}$ is the identity matrix, and $0 < \gamma < 1$ is the regularization parameter, assigned a value of 0.99.

4.3. Classification Computational Complexity

For the classification complexity, we assume the weights or kernel have been pre-computed, and their complexity is outlined in Section 3.3. A weighted K -NN has a complexity of $O(J)$, independent of the dimension of the data N . Confidence voting requires the computation of the predicted class for each point resulting in a complexity of $O(J^2)$. Ridge regression is the most expensive method of classification as it requires the inversion of a matrix and has complexity $O(J^3)$.

5. Evaluation Methods

The following outlines our method of evaluating the LSSLD kernel-based classifiers. We could not directly compare the accuracy of the LSSLD kernel-based classifier to the accuracies of the classifiers in [1,18,19] as we found it challenging to reproduce their results. Some of the details surrounding their evaluation methods were ambiguous. Thus, we computed our own performance baseline of existing methods from [1,18,19] to compare the LSSLD kernel-based classifier against. Additionally, we describe our evaluations in detail below such that our results can be reproduced.

5.1. Datasets

This research uses 23 benchmark datasets from the UCI machine learning repository [11], as outlined in Table 3. These datasets are used as benchmarks in many areas of classification research [1,23–26,37,38] and vary in the number of points J , number of input dimensions N , and number of classes C . There are some notable remarks about the datasets that must be stated for the purpose of repeatability. To begin, for the Ionosphere dataset ($J = 351, N = 33$), either the 1st or 34th attribute can be interpreted as the label. This research uses the 34th attribute as the label, and the 1st attribute is included in the feature vector for $\mathbf{x}$. The smallest dataset, Vehicle, has $J = 94$, which prevents exploring values of $K > 94$ in Tables 4 and 6 in Section 6 as K cannot exceed J in these classification methods. Additionally, some datasets such as Glass suffer from extremely unbalanced classes. In the case of Glass, classes 0 and 1 have approximately 70 elements, while classes 2 and 4 have less than 20. For class 5 with 9 elements, after partitioning the train–test sets, the training set has an insufficient 5 elements. In these cases, specifically, the train–test partition must be carefully selected to ensure all classes are represented in both sets.

Table 3. A summary of the datasets used in this research.

Dataset Name	Size	Dimensions	Num. Classes
Banknote	1372	4	2
Bioddeg	1055	41	2
Diabetes	768	8	2
EEG	14,980	14	2
Glass	214	9	7
Heart	270	13	2
Ionosphere	351	33	2

Table 3. Cont.

Dataset Name	Size	Dimensions	Num. Classes
Iris	150	4	3
Lympho	148	18	4
Musk	6598	166	2
Parkinson	1040	27	2
Phoneme	5404	5	2
Rice	3810	7	2
Segmen	2310	19	7
Sonar	208	60	2
Vowel	528	10	11
Wine	178	13	3
Yeast	1484	8	10
Australian	690	14	2
Balance	625	4	3
Haberman	306	3	2
Monks	432	6	2
Vehicle	94	18	4

5.2. Accuracy

Figure 1 is an example of a multi-class confusion matrix. Accuracy is a common metric of classifier success. Generally accuracy A can be described as

$$A = \frac{\text{Number of correct classifications}}{\text{Total number of test samples}}.$$

The question is, what is considered a *correct classification*? In the two-class case, the answer is obvious and well-agreed upon. The ‘number of correct classifications’ in the two-class case is the sum of the two diagonal entries in the confusion matrix (true positives and true negatives). This is independent of the class one considers as ‘positive’, and thus accuracy is also independent of class:

$$A = \frac{TP + TN}{TP + TN + FP + FN}. \quad (8)$$

However, if there are more than two classes, what is considered a ‘correct classification’ varies depending on the source. Following the work in [29,30], this work considers both true positives and true negatives as ‘correct’ in the multi-class case, thus the accuracy A_c depends on the class:

$$A_c = \frac{TP_c + TN_c}{TP_c + TN_c + FP_c + FN_c}.$$

In this research, to produce a single accuracy measurement the accuracies are averaged across all classes.

$$A = \frac{\sum_{c=1}^C \frac{TP_c + TN_c}{TP_c + TN_c + FP_c + FN_c}}{C}. \quad (9)$$

As stated, not all sources agree on this method of calculating multi-class accuracy, for example, ref. [28] does not include true negatives as ‘correct classifications’ and thus when measuring accuracy only considers entries on the diagonal of the confusion matrix as ‘correct’. This results in an entirely different measure of accuracy. In examples using UCI datasets where the method of calculating accuracy is ambiguous [1,24,25,37], it is not possible to compare the accuracy of a new method of classification to their published baseline. A consistent and clear definition of accuracy is essential for establishing a repeatable baseline. Therefore, the method of calculating accuracy in this research is made explicitly clear in Equation (9).

		Predicted Class					
		Class 1	Class 2	...	Class c	...	Class C
Actual Class	Class 1	$S_{1,1}$	$S_{1,2}$	...	$S_{1,c}$	...	$S_{1,C}$
	Class 2	$S_{2,1}$	$S_{2,2}$	...	$S_{2,c}$	...	$S_{2,C}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	Class c	$S_{c,1}$	$S_{c,2}$	...	$S_{c,c}$	...	$S_{c,C}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
	Class C	$S_{C,1}$	$S_{C,2}$	...	$S_{C,c}$	...	$S_{C,C}$

Figure 1. Example confusion matrix color-coded for measuring the success metrics pertaining to class c . True positives (TP_c) are in the green cell, false negatives (FN_c) are the sum of the red cells, false positives (FP_c) are the sum of the orange cells, and true negatives (TN_c) are the sum of the blue cells.

5.3. Train–Test Split

The question of how to partition the data into a training set and testing set is also an important consideration. In the literature, a common approach is to split the benchmark datasets with a 66:33 [1], 70:30 [2,3] or 80:20 [23] train–test partition. The success of the classifier is partially dependent on which data are in the training set and which data are in the testing set, especially in small datasets or datasets with under-represented classes. When the partitions are not published, it can be difficult to reproduce the results.

The histogram in Figure 2a illustrates classification accuracy on 100 random 60:40 partitions of the ‘smaller’ dataset Heart. From this histogram, it is clear that the choice of partition can significantly impact the accuracy. Here, the accuracy varies by 20%.

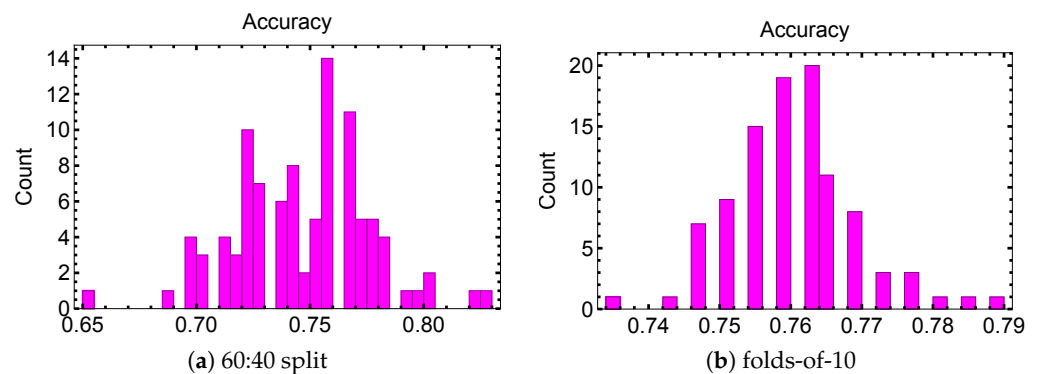


Figure 2. Two histograms illustrating the distributions of accuracy calculated from random train–test partitions of the dataset Heart, classified using Euclidean distance by a 1-nearest neighbor classifier. (b) is 100 trials from random folds-of-10, whereas (a) is 100 trials of 60:40 random train–test partitioning. Classes were represented proportionally in both the train and test sets.

To obtain more objective, repeatable, and accurate metrics of success, n -fold testing is often used, which divides the data into n sets and iterates through the train–test process n times, each time reserving one of the n sets for testing and using the others for training, before averaging the success metrics of all n iterations. Averaging across n folds improves the consistency of the success metrics but still relies on a random partitioning of the n sets that could lead to inconsistencies when the experiment is reproduced. Figure 2b illustrates the distribution of accuracy for 100 randomly selected fold partitions for folds-of-10 trials. The accuracy varies by 6% even with the 10-fold method. In many cases of comparing two or more classifiers, the reported gain in accuracy is only 1 or 2 percent, well within this range of variance.

Additionally, for both the train–test partitioning and the n -fold testing, the effect of the distribution of the classes between the train–test partition can have a very significant impact on accuracy, as picking 10 samples at random in a multi-class problem does not ensure each class is represented appropriately in the test and training sets. We illustrate that the problems we encounter with the train–test partition or n -fold method are not

related to the ratios of the partitions themselves but to the randomness in determining which data points belong to which set. It was observed during experimentation that it is very easy to overestimate the success of a classifier by selecting partitions that favor higher accuracy results.

To create an objective and repeatable baseline, this research uses leave-one-out testing, which means each sample is to be tested against all other samples, resulting in a 1: J train-test ratio. It is commonly agreed in the validation literature that leave-one-out testing, while computationally more expensive, has a consistent performance [39]. Leaving every sample in the dataset out exactly once results in a consistent metric of success. Leave-one-out testing is efficient to compute for blended-model classification as the kernel is only computed once for all data points, and the matrix G , the known labels, is updated in place for each test.

5.4. Pre-Processing

Finally, the method of preparing the data must be addressed in order to reproduce these results. All datasets were scaled dimension-wise between $[0, 1]$. No feature extraction or reduction (e.g., PCA) is performed on the data in this experiment.

6. Experimental Results

We implemented a classification system using C++ with the Eigen library [40], and an additional system was implemented in Python to verify consistency across languages. The system performs classification on datasets using methods including K -NN with distances from Table 1, weighted K -NN with weights from Table 2, and the LSSLD kernel techniques outlined in Section 4.

In the traditional K -NN classifiers, distances can be computed on-demand during or prior to testing. However, using a LSSLD kernel requires the kernel to be computed prior to testing as it has a local scaling factor that must be precomputed. The local scaling factor computation requires solving a non-linear equation with Newton's method, which is efficient and converges quickly. Specifically, the solver computed for a maximum of 10 iterations or until a tolerance of ± 0.001 was reached. In addition to computing the nearest neighbors of each data point, confidence voting requires the normalization of each row and symmetrizing the kernel weights, both constant in the number of data points.

In place of using a boldface font to indicate the highest accuracy of each dataset, we have chosen to present the results with a fuchsia gradient. The highest accuracy for each dataset is highlighted with a 100% saturation, and correspondingly decreasing to a zero percent saturation for accuracies with a difference of 0.02 or less from the best.

6.1. Baselines

To establish a baseline, we ran K -NN classifier using four of the distance metrics from Table 1: the Euclidean distance, Hassanat distance (as it was presented to be the best option in general, as reported in [1]), cosine distance, and Manhattan distance. Their accuracy from leave-one-out testing is presented in Table 4. Note that the optimal K is presented in brackets for each metric, as searched for in the range of $[1, 99]$. This is computationally expensive and creates a tough baseline for the LSSLD kernel classifier to compare with. In a few cases (e.g., EEG), the classification accuracy is consistent across all metrics; however, for most datasets, there is at least one metric that outperforms the others. In some cases, choosing an ill-suited distance metric for a particular dataset can result in a greater than 20% loss of accuracy. Furthermore, only using 1-NN for any one distance metric (such as in [1]) could result in a 10% loss of accuracy or greater. It is evident from the last column that the accuracy of the classifier varies greatly with the choice of metric, and either significant computation effort should be made to determine the best operating parameters for each dataset or a method that achieves consistent accuracy regardless of operational parameters should be used.

Table 4. Accuracy of 4 popular distance metrics for a K -NN classifier from Equation (1). The metrics for both $K = 1$ and the best K (i.e., the K that results in the best accuracy) are presented, with the best K in brackets. The final column is the range that the accuracies from different distance metrics span, considering only the four best K columns presented here.

Dataset	Euclidean		Hassanat		Cosine		Manhattan		Range
	$K = 1$	Best K	$K = 1$	Best K	$K = 1$	Best K	$K = 1$	Best K	Best K
Banknote	0.999	0.999 (1)	0.999	0.999 (1)	0.977	0.980 (3)	0.999	0.999 (1)	0.019
Biodeg	0.841	0.860 (5)	0.834	0.864 (5)	0.830	0.860 (25)	0.830	0.867 (13)	0.007
Diabetes	0.706	0.759 (41)	0.677	0.749 (41)	0.652	0.694 (33)	0.698	0.758 (33)	0.065
EEG	0.982	0.982 (1)	0.979	0.979 (1)	0.980	0.980 (1)	0.979	0.979 (1)	0.003
Glass	0.915	0.915 (1)	0.921	0.921 (1)	0.916	0.916 (1)	0.923	0.923 (1)	0.008
Heart	0.759	0.811 (19)	0.748	0.819 (25)	0.741	0.826 (33)	0.778	0.826 (15)	0.015
Ionosphere	0.892	0.897 (5)	0.889	0.906 (5)	0.892	0.897 (5)	0.903	0.903 (1)	0.009
Iris	0.969	0.978 (7)	0.956	0.973 (23)	0.849	0.884 (15)	0.960	0.969 (3)	0.094
Lympho	0.872	0.909 (15)	0.895	0.929 (7)	0.875	0.916 (7)	0.885	0.919 (3)	0.020
Musk	0.953	0.968 (3)	0.968	0.975 (3)	0.952	0.966 (3)	0.959	0.972 (3)	0.009
Parkinson	0.910	0.917 (7)	0.671	0.682 (23)	0.914	0.915 (7)	0.889	0.931 (5)	0.249
Phoneme	0.908	0.908 (1)	0.913	0.913 (1)	0.886	0.886 (1)	0.909	0.909 (1)	0.027
Rice	0.887	0.928 (35)	0.883	0.929 (25)	0.893	0.926 (25)	0.882	0.929 (23)	0.003
Segman	0.993	0.993 (1)	0.993	0.993 (1)	0.993	0.993 (1)	0.993	0.993 (1)	0.000
Sonar	0.875	0.875 (1)	0.865	0.865 (1)	0.851	0.851 (1)	0.851	0.851 (1)	0.024
Vowel	0.999	0.999 (1)	0.994	0.994 (1)	0.999	0.999 (1)	0.998	0.998 (1)	0.005
Wine	0.966	0.985 (17)	0.899	0.906 (5)	0.966	0.978 (7)	0.974	0.989 (15)	0.083
Yeast	0.906	0.920 (27)	0.907	0.919 (17)	0.905	0.919 (23)	0.908	0.919 (15)	0.001
Australian	0.794	0.865 (25)	0.801	0.867 (17)	0.800	0.862 (27)	0.799	0.875 (37)	0.013
Balance	0.953	0.953 (1)	0.797	0.850 (27)	0.991	0.991 (1)	0.953	0.953 (1)	0.141
Haberman	0.680	0.765 (37)	0.641	0.755 (17)	0.660	0.775 (27)	0.650	0.765 (39)	0.020
Monks	0.826	0.991 (5)	0.646	0.701 (39)	0.812	0.975 (5)	0.938	0.998 (5)	0.297
Vehicle	0.803	0.803 (1)	0.761	0.761 (1)	0.798	0.798 (1)	0.819	0.819 (1)	0.058

Figure 3 shows, as is well known, that the accuracy of a K -NN classifier is very sensitive to the value of K and in most cases degenerates as K increases, varying substantially in the process. In contrast, the proposed LSSLD KRR and LSSLD kernel confidence vote classifiers have a nearly flat curve, the dependence on K is minimal, and the accuracy is consistent with increasing K .

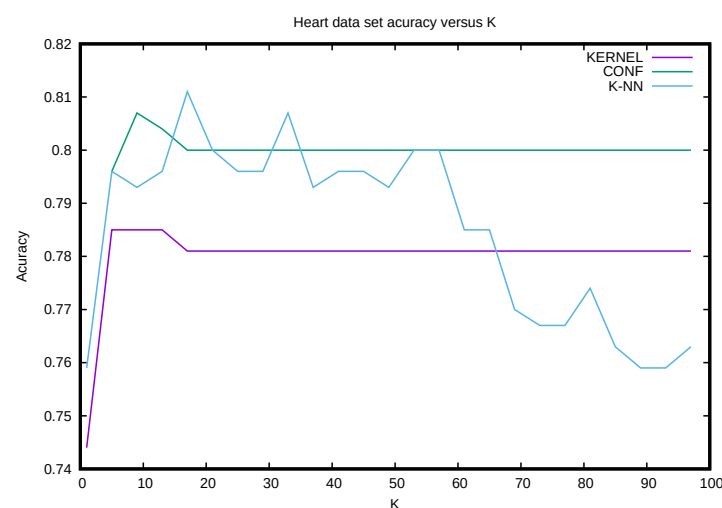


Figure 3. The accuracy of a traditional K -NN classifier compared with the accuracy of a confidence voting and ridge regression (KRR) classifier using the LSSLD kernel on the Heart dataset. The accuracy of the traditional K -NN classifier varies significantly depending on the value of K , degrading as K increases. In contrast, confidence voting and ridge regression using the LSSLD kernel report consistent accuracy across all values of K .

Table 5 presents the baseline accuracy results of four popular weighted K-NN techniques. It is evident that most datasets tend to achieve good accuracy with the WKNN/WDKNN weights or the EXPKNN/NORKNN weights but usually not both. Interestingly, WDKNN [19], which was presented as an improvement on WKNN, reports identical accuracy to WKNN when comparing accuracy using leave-one-out testing. The authors interpret this as support for the use of leave-one-out testing to achieve consistent evaluation metrics that consistently capture the accuracy of the classifier.

Table 5. Accuracy of a weighted K-NN classifier using popular choices of weights from Table 2 and best K from $K = 5, 10, 20, 50$.

Dataset	WKNN	WDKNN	EXPKNN	NORKNN
Banknote	0.999 (5)	0.999 (5)	0.999 (5)	0.999 (5)
Biodeg	0.841 (5)	0.841 (5)	0.847 (5)	0.843 (5)
Diabetes	0.706 (5)	0.706 (5)	0.741 (50)	0.742 (50)
EEG	0.982 (5)	0.982 (5)	0.970 (5)	0.963 (5)
Glass	0.915 (5)	0.915 (5)	0.897 (5)	0.896 (5)
Heart	0.759 (5)	0.759 (5)	0.822 (10)	0.811 (10)
Ionosphere	0.892 (5)	0.892 (5)	0.897 (5)	0.897 (5)
Iris	0.969 (5)	0.969 (5)	0.973 (10)	0.973 (10)
Lympho	0.872 (5)	0.872 (5)	0.902 (20)	0.902 (20)
Musk	0.953 (5)	0.953 (5)	0.963 (5)	0.960 (5)
Parkinson	0.910 (5)	0.910 (5)	0.913 (5)	0.912 (5)
Phoneme	0.908 (5)	0.908 (5)	0.873 (5)	0.870 (5)
Rice	0.887 (5)	0.887 (5)	0.927 (20)	0.926 (50)
Segman	0.993 (5)	0.993 (5)	0.986 (5)	0.985 (5)
Sonar	0.875 (5)	0.875 (5)	0.803 (5)	0.779 (5)
Vowel	0.999 (5)	0.999 (5)	0.985 (5)	0.983 (5)
Wine	0.966 (5)	0.966 (5)	0.981 (20)	0.981 (20)
Yeast	0.906 (5)	0.906 (5)	0.918 (20)	0.917 (20)
Australian	0.794 (5)	0.794 (5)	0.861 (20)	0.862 (20)
Balance	0.966 (10)	0.966 (10)	0.939 (50)	0.933 (50)
Haberman	0.706 (5)	0.706 (5)	0.755 (50)	0.745 (20)
Monks	0.875 (50)	0.875 (50)	0.988 (5)	0.975 (5)
Vehicle	0.803 (5)	0.803 (5)	0.739 (5)	0.745 (10)

6.2. One Tuning for All

One key objective was to design a kernel-based classifier for which one value for each operating parameter (in this case K , α , and β) would work well for all datasets. Without this insensitivity, the operating parameters would have to be selected using trial and error, which is a computationally expensive task and not practical in application.

Table 6 presents the accuracy of three LSSLD classifiers: K-NN weighted with LSSLD weights (i.e., blended-model with $\beta = 0$), confidence voting (i.e., blended-model with $\beta = 1$), and KRR. As in Table 5, the accuracies of the kernel-based classifiers are quite stable with respect to K but do vary across the three methods of classification. Notably, the most computationally expensive method, KRR, does not significantly outperform the other less computationally expensive methods; for most datasets, it does not outperform the other methods at all.

Table 6. Accuracy of different LSSLD kernel-based classifiers (LSSLD K-NN, confidence voting, kernel ridge regression) with $\alpha = 2$ and varying K .

Dataset	LSSLD K-NN ($\beta = 0$) (4)					LSSLD Confidence Vote ($\beta = 1$) (5)			LSSLD KRR (7)	
	$K = 5$	$K = 10$	$K = 20$	$K = 50$	$K = 100$	$K = 5$	$K = 10$	$K = 20$	$K = 20$	$K = 50$
Banknote	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Biodeg	0.863	0.863	0.864	0.864	0.864	0.846	0.851	0.856	0.866	0.865
Diabetes	0.757	0.760	0.751	0.750	0.751	0.757	0.753	0.754	0.759	0.759
EEG	0.985	0.985	0.985	0.985	0.985	0.970	0.970	0.969	0.984	0.985
Glass	0.925	0.924	0.924	0.924	0.923	0.908	0.904	0.907	0.923	0.923
Heart	0.785	0.789	0.781	0.781	0.781	0.796	0.804	0.800	0.796	0.793
Ionosphere	0.877	0.875	0.875	0.877	0.877	0.897	0.897	0.895	0.883	0.880
Iris	0.964	0.964	0.964	0.964	0.964	0.973	0.969	0.969	0.969	0.969
Lympho	0.905	0.905	0.905	0.899	0.899	0.892	0.895	0.899	0.905	0.902
Musk	0.973	0.972	0.970	0.970	0.970	0.963	0.960	0.961	0.970	0.970
Parkinson	0.922	0.926	0.923	0.923	0.923	0.890	0.901	0.898	0.928	0.929
Phoneme	0.901	0.900	0.901	0.901	0.901	0.884	0.882	0.881	0.902	0.902
Rice	0.910	0.909	0.909	0.909	0.909	0.925	0.927	0.927	0.912	0.912
Segman	0.990	0.991	0.991	0.991	0.991	0.988	0.988	0.988	0.991	0.991
Sonar	0.856	0.865	0.865	0.865	0.865	0.832	0.837	0.827	0.856	0.856
Vowel	0.999	0.999	0.999	0.999	0.999	0.995	0.994	0.995	0.998	0.998
Wine	0.978	0.974	0.974	0.970	0.970	0.981	0.981	0.981	0.974	0.974
Yeast	0.909	0.910	0.911	0.911	0.911	0.920	0.921	0.920	0.912	0.912
Australian	0.830	0.832	0.832	0.832	0.835	0.841	0.851	0.846	0.835	0.838
Balance	0.867	0.881	0.921	0.931	0.931	0.947	0.958	0.949	0.924	0.931
Haberman	0.703	0.706	0.709	0.709	0.709	0.729	0.735	0.732	0.712	0.712
Monks	0.956	0.933	0.912	0.910	0.910	0.810	0.792	0.780	0.907	0.900
Vehicle	0.798	0.793	0.787	0.787	0.000	0.745	0.755	0.750	0.798	0.793

Table 7 demonstrates the effect of α on the kernel-based classifier's accuracy. From this, it is clear that the best choice for α varies across datasets, but in most cases, the choice of α has little impact on the classification accuracy.

Table 7. Accuracy of a LSSKD K-NN classifier (4) using LSSLD weights and varying parameter α , here $K = 20$.

Dataset	$\alpha = 2$	$\alpha = 3$	$\alpha = 5$	$\alpha = 7$	$\alpha = 10$	$\alpha = \log(K)$
Banknote	0.999	0.999	0.999	0.999	0.999	0.999
Biodeg	0.864	0.878	0.875	0.869	0.866	0.876
Diabetes	0.751	0.763	0.757	0.747	0.746	0.758
EEG	0.985	0.982	0.976	0.970	0.964	0.978
Glass	0.924	0.919	0.909	0.905	0.897	0.912
Heart	0.781	0.804	0.800	0.815	0.815	0.804
Ionosphere	0.875	0.883	0.892	0.889	0.892	0.895
Iris	0.964	0.969	0.964	0.964	0.969	0.969
Lympho	0.905	0.912	0.909	0.916	0.909	0.905
Musk	0.970	0.967	0.964	0.962	0.960	0.965
Parkinson	0.923	0.926	0.926	0.920	0.913	0.928
Phoneme	0.901	0.897	0.889	0.879	0.868	0.893
Rice	0.909	0.920	0.925	0.928	0.927	0.925
Segman	0.991	0.990	0.989	0.988	0.987	0.989
Sonar	0.865	0.841	0.812	0.769	0.731	0.817
Vowel	0.999	0.997	0.993	0.982	0.959	0.994
Wine	0.974	0.978	0.985	0.989	0.985	0.985
Yeast	0.911	0.915	0.918	0.919	0.919	0.917
Australian	0.832	0.848	0.851	0.854	0.858	0.851
Balance	0.921	0.921	0.921	0.930	0.930	0.921
Haberman	0.709	0.719	0.739	0.745	0.739	0.729
Monks	0.912	0.963	0.905	0.887	0.861	0.921
Vehicle	0.787	0.793	0.771	0.761	0.755	0.771

Finally, Table 8 reports the accuracy of the blended-model classifier. The trend seems to indicate that the kernel-based classifier has higher accuracy on some datasets with $\beta = 0$, indicating the given labels of the data are preferred to make successful classifications, whereas the kernel-based classifier has higher accuracy on other datasets when $\beta = 1$, indicating the labels inferred by the kernel are preferred to make successful classifications. One interpretation of this phenomenon is that datasets that report a higher accuracy with $\beta = 1$ might have more outliers or mislabeled points. From Table 8 β of 0.5 produces the most consistently high accuracy.

Table 8. Accuracy of a blended-model classifier (6) using LSSLD weights and varying parameter β , here $K = 20$ and $\alpha = 2$.

Dataset	$\beta = 0$	$\beta = 0.25$	$\beta = 0.5$	$\beta = 0.75$	$\beta = 1$
Banknote	0.999	0.999	0.999	0.999	0.999
Biodeg	0.864	0.870	0.873	0.866	0.856
Diabetes	0.751	0.760	0.767	0.758	0.754
EEG	0.985	0.984	0.981	0.977	0.969
Glass	0.924	0.920	0.923	0.917	0.907
Heart	0.781	0.793	0.800	0.796	0.800
Ionosphere	0.875	0.892	0.895	0.886	0.895
Iris	0.964	0.969	0.969	0.969	0.969
Lympho	0.905	0.909	0.909	0.905	0.899
Musk	0.970	0.969	0.967	0.964	0.961
Parkinson	0.923	0.927	0.921	0.913	0.898
Phoneme	0.901	0.901	0.900	0.895	0.881
Rice	0.909	0.914	0.920	0.924	0.927
Segman	0.991	0.991	0.990	0.988	0.988
Sonar	0.865	0.856	0.861	0.856	0.827
Vowel	0.999	0.998	0.998	0.997	0.995
Wine	0.974	0.978	0.978	0.978	0.981
Yeast	0.911	0.913	0.916	0.919	0.920
Australian	0.832	0.838	0.843	0.846	0.846
Balance	0.921	0.931	0.939	0.939	0.949
Haberman	0.709	0.716	0.722	0.729	0.732
Monks	0.912	0.907	0.891	0.829	0.780
Vehicle	0.787	0.798	0.782	0.766	0.750

6.3. LSSLD Kernel Comparison

Table 9 compares the accuracies of the LSSLD kernel-based classifier methods to the accuracy of the K -NN classifier from Table 4 with the highest accuracy (selected across all distance metrics and values of K). As the accuracies of the three LSSLD classifiers do not vary much with respect to their operating parameters, the parameters have been set to $K = 10$, $\alpha = 2$, and $\beta = 0.5$ where applicable. Additionally, running the LSSLD classifier with three different values of β ($\beta = 0, 0.5, 1$) only requires one computation of the kernel matrix as the kernel matrix does not depend on β . Table 9 demonstrates that the LSSLD classifier techniques have comparable, if not better, accuracies than the best K -NN metric- K combination, with the additional benefit of requiring significantly less computational effort, as the specific operating parameters do not have a major influence on the accuracy.

The LSSLD methods are less accurate on datasets that violate the manifold hypothesis [21] (Vehicle, Australian, Balance) and especially on data that are artificial (Haberman) or combinatorial (Monks). For that reason, those datasets have been listed at the bottom of Tables 9 and 10.

Table 10 compares the accuracies of the LSSLD kernel-based classifier methods to the accuracy of the weighted K -NN classifier from Table 5 with the highest accuracy (selected across all weights and values of K). Again, the LSSLD methods have comparable, if not better, accuracies than the weighted K -NN techniques but require significantly less computation effort thanks to their consistency in accuracy over the operating parameters.

Table 9. Comparing three LSSLD classifiers ($\alpha = 2$, $K = 10$) to the most accurate K -NN distance metric with the most accurate K .

	LSSLD K-NN ($\beta = 0$)	Blended ($\beta = 0.5$)	Confidence Vote ($\beta = 1$)	Best K-NN	Difference to Best
Banknote	0.999	0.999	0.999	0.999	0.000
Biodeg	0.863	0.871	0.851	0.867	−0.004
Diabetes	0.760	0.764	0.753	0.759	−0.005
EEG	0.985	0.981	0.970	0.982	−0.003
Glass	0.924	0.925	0.904	0.923	−0.002
Heart	0.789	0.807	0.804	0.826	0.019
Ionosphere	0.875	0.889	0.897	0.906	0.009
Iris	0.964	0.969	0.969	0.978	0.009
Lympho	0.905	0.912	0.895	0.929	0.017
Musk	0.972	0.968	0.960	0.975	0.003
Parkinson	0.926	0.921	0.901	0.931	0.005
Phoneme	0.900	0.899	0.882	0.913	0.013
Rice	0.909	0.918	0.927	0.929	0.002
Segman	0.991	0.990	0.988	0.993	0.002
Sonar	0.865	0.856	0.837	0.875	0.010
Vowel	0.999	0.998	0.994	0.999	0.000
Wine	0.974	0.978	0.981	0.989	0.008
Yeast	0.910	0.915	0.921	0.920	−0.001
Australian	0.832	0.838	0.851	0.875	0.024
Balance	0.881	0.906	0.958	0.991	0.033
Haberman	0.706	0.706	0.735	0.775	0.040
Monks	0.933	0.884	0.792	0.998	0.065
Vehicle	0.793	0.771	0.755	0.819	0.026

Table 10. Comparing three LSSLD classifiers ($\alpha = 2$, $K = 10$) to the most accurate weighted K -NN classifier.

	LSSLD K-NN ($\beta = 0$)	Blended ($\beta = 0.5$)	Confidence Vote ($\beta = 1$)	Best Weighted K-NN	Difference to Best
Banknote	0.999	0.999	0.999	0.999	0.000
Biodeg	0.863	0.871	0.851	0.847	−0.016
Diabetes	0.760	0.764	0.753	0.742	−0.002
EEG	0.985	0.981	0.970	0.982	−0.003
Glass	0.924	0.925	0.904	0.915	−0.010
Heart	0.789	0.807	0.804	0.822	0.015
Ionosphere	0.875	0.889	0.897	0.897	0.000
Iris	0.964	0.969	0.969	0.973	0.004
Lympho	0.905	0.912	0.895	0.902	−0.010
Musk	0.972	0.968	0.960	0.963	−0.009
Parkinson	0.926	0.921	0.901	0.913	−0.013
Phoneme	0.900	0.899	0.882	0.908	0.008
Rice	0.909	0.918	0.927	0.927	0.000
Segman	0.991	0.990	0.988	0.993	0.002
Sonar	0.865	0.856	0.837	0.875	0.010
Vowel	0.999	0.998	0.994	0.999	0.000
Wine	0.974	0.978	0.981	0.981	0.000
Yeast	0.910	0.915	0.921	0.918	−0.003
Australian	0.832	0.838	0.851	0.862	0.011
Balance	0.881	0.906	0.958	0.966	0.008
Haberman	0.706	0.706	0.735	0.755	0.020
Monks	0.933	0.884	0.792	0.988	0.055
Vehicle	0.793	0.771	0.755	0.803	0.010

7. Conclusions

Determining the best choice of operating parameters, the first step in many methods of classification, is critical for the accuracy of the classifier used. The dimensionality reduction community developed methods aimed at capturing/learning the topological structure of the training data. We present a kernel approach, using a Laplacian kernel with local scaling.

Experiments using benchmark data sets demonstrate that the LSSLD kernel achieves comparable leave-one-out prediction accuracy scores using one single tuning/set of parameters for all data sets to weighted and non-weighted K -NN classifiers tuned over a variety of metrics or weights and neighborhood sizes tuned for each data set individually. Using confidence voting, representing a trade-off between the given labels and the labels implied by the kernel further confirms that the presented kernel is a promising step towards a ‘one tuning for all’ classifier.

8. Future Work

Inspired by the success of the LSSLD kernel, we want to further research the methods of kernel construction that capture similarity. The accuracy of the LSSLD kernel was satisfactory, but we believe accuracy could be improved further, especially regarding datasets that did not satisfy the manifold hypothesis (Vehicle, Australian, Balance, Haberman, Monks), as mentioned in Section 6.3. One way the classification accuracy might be improved is by utilizing the label information of the dataset in the construction of a kernel. As kernels can be linearly combined to construct ‘new’ kernels, the LSSLD kernel could be enhanced with ‘supervised’ kernels constructed with the information provided by the labels to create ‘semi-supervised’ kernels. We also could continue to explore the relationship between functions of classification and kernels, and how the classifier decisions could be encapsulated in the kernel.

Author Contributions: Conceptualization, E.H. and M.v.M.; methodology, E.H. and M.v.M.; software, E.H. and M.v.M.; validation, E.H. and M.v.M.; formal analysis, E.H. and M.v.M.; investigation, E.H. and M.v.M.; resources, E.H. and M.v.M.; data curation, M.v.M.; writing—original draft preparation, E.H. and M.v.M.; writing—review and editing, E.H. and M.v.M.; visualization, E.H. and M.v.M.; supervision, M.v.M.; project administration, M.v.M.; funding acquisition, E.H. and M.v.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by MITACS grant number IT17587.

Data Availability Statement: The data presented in this study are available in UCI Machine Learning Repository at <https://archive.ics.uci.edu>.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DRT	Dimensionality Reduction Techniques
DWKNN	Distance Weighted K -Nearest Neighbors
EXPKNN	Exponentially weighted K -Nearest Neighbors
KRR	Kernel Ridge Regression
LSSLD	Locally Scaled Symmetric Laplacian Diffusion
K -NN	K -Nearest Neighbors
NORKNN	Normally weighted K -Nearest Neighbors
SVM	Support Vector Machine
t-SNE	t-distributed Stochastic Neighbor Embedding
UCI	University of California Irvine
UMAP	Uniform Manifold Approximation and Projection

References

1. Abu Alfeilat, H.A.; Hassanat, A.B.; Lasassmeh, O.; Tarawneh, A.S.; Alhasanat, M.B.; Eyal Salman, H.S.; Prasath, V.S. Effects of distance measure choice on k-nearest neighbor classifier performance: A review. *Big Data* **2019**, *7*, 221–248. [\[CrossRef\]](#)
2. Alkasassbeh, M.; Altarawneh, G.A.; Hassanat, A. On enhancing the performance of nearest neighbour classifiers using hassanat distance metric. *arXiv* **2015**, arXiv:1501.00687.
3. Nayak, S.; Bhat, M.; Reddy, N.S.; Rao, B.A. Study of distance metrics on k-nearest neighbor algorithm for star categorization. *J. Phys. Conf. Ser.* **2022**, *2161*, 012004. [\[CrossRef\]](#)
4. Zhang, C.; Zhong, P.; Liu, M.; Song, Q.; Liang, Z.; Wang, X. Hybrid metric k-nearest neighbor algorithm and applications. *Math. Probl. Eng.* **2022**, *2022*, 8212546. [\[CrossRef\]](#)
5. Yean, C.W.; Khairunizam, W.; Omar, M.I.; Murugappan, M.; Zheng, B.S.; Bakar, S.A.; Razlan, Z.M.; Ibrahim, Z. Analysis of the distance metrics of KNN classifier for EEG signal in stroke patients. In Proceedings of the 2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), Kuching, Malaysia, 15–17 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–4.
6. Ratnasari, D. Comparison of Performance of Four Distance Metric Algorithms in K-Nearest Neighbor Method on Diabetes Patient Data. *Indones. J. Data Sci.* **2023**, *4*, 97–108. [\[CrossRef\]](#)
7. Hofer, E.; v. Mohrenschildt, M. Model-Free Data Mining of Families of Rotating Machinery. *Appl. Sci.* **2022**, *12*, 3178. [\[CrossRef\]](#)
8. Ghojogh, B.; Ghodsi, A.; Karray, F.; Crowley, M. Reproducing Kernel Hilbert Space, Mercer’s Theorem, Eigenfunctions, Nyström Method, and Use of Kernels in Machine Learning: Tutorial and Survey. *arXiv* **2021**, arXiv:2106.08443.
9. Kang, Z.; Peng, C.; Cheng, Q. Kernel-driven similarity learning. *Neurocomputing* **2017**, *267*, 210–219. [\[CrossRef\]](#)
10. Rousseeuw, P.J.; Hubert, M. Robust statistics for outlier detection. In *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*; Wiley & Sons: Hoboken, NJ, USA, 2011; Volume 1, pp. 73–79.
11. Dua, D.; Graff, C. *UCI Machine Learning Repository*; University of California, Irvine, School of Information and Computer Sciences: Irvine, CA, USA, 2017.
12. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [\[CrossRef\]](#)
13. Samworth, R.J. Optimal weighted nearest neighbour classifiers. *Ann. Statist.* **2012**, *40*, 2733–2763. [\[CrossRef\]](#)
14. Al Daoud, E.; Turabieh, H. New empirical nonparametric kernels for support vector machine classification. *Appl. Soft Comput.* **2013**, *13*, 1759–1765. [\[CrossRef\]](#)
15. McInnes, L.; Healy, J.; Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv* **2018**, arXiv:1802.03426.
16. Belkin, M.; Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **2003**, *15*, 1373–1396. [\[CrossRef\]](#)
17. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
18. Dudani, S. The distance-weighted k-nearest neighbor rule. *IEEE Trans. Syst. Man Cybern.* **1978**, *8*, 311–313. [\[CrossRef\]](#)
19. Gou, J.; Du, L.; Zhang, Y.; Xiong, T. A new distance-weighted k-nearest neighbor classifier. *J. Inf. Comput. Sci.* **2012**, *9*, 1429–1436.
20. Hong, P.; Luo, L.; Lin, C. The Parameter Optimization of Gaussian Function via the Similarity Comparison within Class and between Classes. In Proceedings of the 2011 Third Pacific-Asia Conference on Circuits, Communications and System (PACCS), Wuhan, China, 17–18 July 2011; pp. 1–4. [\[CrossRef\]](#)
21. Fefferman, C.; Mitter, S.; Narayanan, H. Testing the manifold hypothesis. *J. Am. Math. Soc.* **2016**, *29*, 983–1049. [\[CrossRef\]](#)
22. Tenenbaum, J.B.; Silva, V.d.; Langford, J.C. A global geometric framework for nonlinear dimensionality reduction. *Science* **2000**, *290*, 2319–2323. [\[CrossRef\]](#)
23. Ali, N.; Neagu, D.; Trundle, P. Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. *SN Appl. Sci.* **2019**, *1*, 1559. [\[CrossRef\]](#)
24. Nasiri, J.A.; Charkari, N.M.; Jalili, S. Least squares twin multi-class classification support vector machine. *Pattern Recognit.* **2015**, *48*, 984–992. [\[CrossRef\]](#)
25. Farid, D.M.; Zhang, L.; Rahman, C.M.; Hossain, M.A.; Strachan, R. Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. *Expert Syst. Appl.* **2014**, *41*, 1937–1946. [\[CrossRef\]](#)
26. Singh, D.; Singh, B. Investigating the impact of data normalization on classification performance. *Appl. Soft Comput.* **2020**, *97*, 105524. [\[CrossRef\]](#)
27. Kapoor, S.; Narayanan, A. Leakage and the reproducibility crisis in machine-learning-based science. *Patterns* **2023**, *4*, 100804. [\[CrossRef\]](#)
28. Grandini, M.; Bagli, E.; Visani, G. Metrics for multi-class classification: An overview. *arXiv* **2020**, arXiv:2008.05756.
29. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [\[CrossRef\]](#)
30. Tharwat, A. Classification assessment methods. *Appl. Comput. Inform.* **2020**, *17*, 168–192. [\[CrossRef\]](#)
31. Kundu, D. Discriminating between normal and Laplace distributions. In *Advances in Ranking and Selection, Multiple Comparisons, and Reliability: Methodology and Applications*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 65–79.
32. Hajij, M.; Zamzmi, G.; Papamarkou, T.; Maroulas, V.; Cai, X. Simplicial complex representation learning. *arXiv* **2021**, arXiv:2103.04046.

33. Ramirez-Padron, R.; Foregger, D.; Manuel, J.; Georgiopoulos, M.; Mederos, B. Similarity kernels for nearest neighbor-based outlier detection. In Proceedings of the Advances in Intelligent Data Analysis IX: 9th International Symposium, IDA 2010, Tucson, AZ, USA, May 19–21 2010; Proceedings 9; Springer: Berlin/Heidelberg, Germany, 2010; pp. 159–170.
34. Dik, A.; Jebari, K.; Bouroumi, A.; Ettouhami, A. Similarity-based approach for outlier detection. *arXiv* **2014**, arXiv:abs/1411.6850.
35. Zhou, D.; Bousquet, O.; Lal, T.; Weston, J.; Schölkopf, B. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA 2003; Volume 16.
36. Liu, W.; Qian, B.; Cui, J.; Liu, J. Spectral kernel learning for semi-supervised classification. In Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, Pasadena, CA, USA, 11–17 July 2009.
37. Song, Q.; Jiang, H.; Liu, J. Feature selection based on FDA and F-score for multi-class classification. *Expert Syst. Appl.* **2017**, *81*, 22–27. [[CrossRef](#)]
38. Khan, M.M.R.; Arif, R.B.; Siddique, M.A.B.; Oishe, M.R. Study and observation of the variation of accuracies of KNN, SVM, LMNN, ENN algorithms on eleven different datasets from UCI machine learning repository. In Proceedings of the 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (ICEEICT), Dhaka, Bangladesh, 13–15 September 2018; pp. 124–129.
39. Vehtari, A.; Gelman, A.; Gabry, J. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Stat. Comput.* **2016**, *27*, 1413–1432. [[CrossRef](#)]
40. Guennebaud, G.; Jacob, B.; Avery, P.; Bachrach, A.; Barthelemy, S.; Becker, C.; Benjamin, D.; Berger, C.; Berres, A.; Luis Blanco, J.; et al. *Eigen*, Version v3; 2010. Available online: <http://eigen.tuxfamily.org> (accessed on 25 September 2019).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.