Statistics and Its Interface Volume 15 (2022) 247–260

A new k-nearest neighbors classifier for functional data*

Tianming Zhu and Jin-Ting Zhang†

For supervised classification of functional data, several classifiers have been proposed in the literature, including the well-known classic k-nearest neighbors (kNN) classifier. The classic kNN classifier selects k nearest neighbors around a new observation and determines its class-membership according to a majority vote. A difficulty arises when there are two classes having the same largest number of votes. To overcome this difficulty, we propose a new kNN classifier which selects k nearest neighbors around a new observation from each class. The class-membership of the new observation is determined by the minimum average distance or semi-distance between the k nearest neighbors and the new observation. Good performance of the new kNN classifier is demonstrated by three simulation studies and two real data examples.

AMS 2000 subject classifications: Primary 62H30; Secondary 62M99.
Keywords and phrases: Functional data analysis, Supervised classification, Functional dissimilarity measures, k-nearest neighbors classifier, Ties broken, Class imbalance problem.

1. INTRODUCTION

In recent years, more and more researchers concentrated their efforts on solving high dimension, low sample size problems. A particular case is that of random variables taking values into an infinite dimensional space, typically a space of functions defined on some continuous set $\mathcal{T}$. This kind of data is often referred to as functional data. In practice, functional data are obtained via observing some measures over time, and we assume the sample of functional observations was generated from a stochastic process. A stochastic function over time may be defined as a sequence of random variables or random vectors, which is usually known as a time series [23]. In the past decades, methods for statistical inferences for functional data have been paid much attention; see for example, see [7] and [23] and references therein. In this paper, we are interested in proposing a new k-nearest neighbors classifier for functional data.

A general G-class classification problem for function data can be described as follows. Suppose we have a training data set consists of G functional samples

$$x_{i1}(t), \ldots, x_{in_i}(t) \overset{i.i.d}{\sim} \mathrm{SP}(\eta_i, \gamma_i),\ i = 1, \ldots, G, \tag{1}$$

where $\eta_i(t), i = 1, \ldots, G$ are the unknown group mean functions, and $\gamma_i(s,t), i = 1, \ldots, G$ are the unknown group covariance functions. We write $x(t) \sim \mathrm{SP}(\eta, \gamma)$ to represent a stochastic process having mean function $\eta(t)$ and covariance function $\gamma(s,t)$ for simplicity. For a new coming observation $x(t)$, our aim is to determine the class membership of $x(t)$ based on the above training data set.

It is well-known that for the classic G-class classification problem for multivariate data, the classic kNN classifier which was proposed by [8] is a widely used approach due to its simplicity and efficiency. It consists of the following steps: given a training set with known class labels, classify a new observation into a class by examining its k nearest neighbors and applying the majority vote principle. The simplest form of kNN is when $k = 1$, that is, the unknown observation is simply assigned to the class of its nearest neighbor. Its asymptotic error rate has been found to approach the optimal Bayes error rate when $k \to \infty$ and $k/n \to 0$ as $n \to \infty$, where n is the training sample size. That is, the classic kNN is universally consistent [19]. Further, it is known that the error rate of the classic kNN is bounded above by twice the optimal Bayes error rate [3]. By using filtering to reduce the infinite dimension of the function space, [2] extended it to infinite-dimensional function spaces. They showed that the functional kNN classifier is also universally consistent in general separable Hilbert spaces.

The classic kNN classifier is of great interests to the statisticians since it was originally proposed. There are several important issues in the classic kNN classifier. A challenging problem is how to choose the number of nearest neighbors k properly. On the one-hand, increasing the value of k can reduce the influence of the noisy observations. On the other-hand, when the training sample size is large, large value of k can lead a long time to run the kNN algorithm, especially in the context of functional data. To find a good k, a usual method is via an M-fold cross-validation approach via trying several k and choosing the one with the minimum cross-validation score. However, it will be very time-consuming

*The work was financially supported by the National University of Singapore academic research grant R-155-000-212-114. The authors thank the co-Editor, the Guest Editor and two anonymous reviewers for their constructive comments and suggestions which help improve the article substantially.
†Corresponding author.

especially for multi-class classification problems for functional data since the amount of the data involved can be very huge. Even if we have selected a best k through an M-fold cross-validation approach, there is a high chance to have a tie among the largest number of nearest neighbors for more than two classes. This is due to the fact that the classic kNN classifier is only based on a simple majority voting principle. This tie problem may cause a difficulty to assign a class membership to a new coming observation. To overcome this problem, we often select an odd value of k for two-class classification problems to break ties, but there is not a general method for selecting the value of k for multi-class classification problems. It is fair to say that at least one of the reasons that the probability of misclassification given by the simple majority rule in the classic kNN classifier is due to the fact that the probability ties taking place is high [4]. In this paper, we propose and study a new kNN classifier to avoid ties. It works generally for multi-class classification problems.

There is a sizeable literature on techniques for overcoming the limitations of classic kNN classifier. For example, [4] proposed a distance-weighted kNN which weights the evidence of a neighbor close to an unclassified observation more heavily than the evidence of another neighbor which is at a greater distance from the unclassified observation. [12] proposed a so-called condensed nearest neighbor algorithm which stores the patterns one by one and eliminates the duplicate ones. [11] proposed a reduced nearest neighbor algorithm which is an improvement over the condensed nearest neighbor algorithm via including one more step that is an elimination of the patterns which are not affecting the training data set result. There are many other algorithms developed in the literature, such as model based kNN [13], rank nearest neighbors [1] and among others.

Although a lot of work has been done for overcoming the limitations of the classic kNN classifier, it still may fail for imbalanced data set. A medical industry application (Section 5.2) has partially motivated this paper. We want to classify a new corneal surface to one of the four groups of corneal surfaces: the normal cornea group, the unilateral suspect cornea group, the suspect map cornea group and the clinical keratoconus cornea group. As the unilateral suspect cornea group has few observations, it is difficult to detect the unilateral suspect cornea. Thus, the second contribution of this paper is to propose an adjusted new kNN classifier for imbalanced data set.

The rest of this paper is organized as follows. Section 2 introduces a new k-nearest neighbors classifier and applies it with several dissimilarity measures. Section 3 suggests an adjusted new kNN classifier for imbalanced classes. The good performance of the new kNN classifier with some functional dissimilarity measures is demonstrated by some simulation results presented in Section 4 and two real data examples presented in Section 5. Some concluding remarks are given in Section 6.

2. NEW K-NEAREST NEIGHBORS CLASSIFIER

In this section, the new kNN classifier for functional data is presented, with its classification rule introduced in Section 2.1, functional dissimilarity measures given in Section 2.2, and tuning parameters selection described in Section 2.3 respectively.

2.1 New kNN classification rule

The new kNN classification rule for functional data can be described as follows. Unlike the classic kNN classifier, instead of applying the majority vote principle, the new kNN classifier selects the same number of k nearest neighbors from each class based on some functional dissimilarity measures such as distance or semi-distance and computes the average of all the dissimilarity measures between the k nearest neighbors and the new observation in each class, respectively. By comparing these average dissimilarity measures, it assigns the new observation to the class with the smallest average dissimilarity measure. Suppose we have the G classes of functional observations in (1) and a new coming functional observation $x(t)$. Then for each class, we select k functional observations which are closest to $x(t)$ according to a functional dissimilarity measure $d(y,z)$ between two functional observations $y(t)$ and $z(t)$, say $x_{ij_1}(t),\ldots,x_{ij_k}(t), i=1,\ldots,G$ such that

$$d(x,x_{ij_1}) \le \cdots \le d(x,x_{ij_k}),\ i=1,\ldots,G. \tag{2}$$

The averages of the dissimilarity measures (2) of the k nearest neighbors for G classes are computed as

$$\tau_i = k^{-1}\sum_{\ell=1}^{k} d(x,x_{ij_\ell}),\ i=1,\ldots,G. \tag{3}$$

The class membership of $x(t)$ is assigned according to the smallest value of τ_i's. That is, we put $x(t)$ to the class g if $g=\operatorname{argmin}_{i=1,\ldots,G}\tau_i$.

It is interesting to compare the new kNN and the classic kNN classifiers. First of all, both the kNN classifiers are easy to implement provided that the number of nearest neighors k and the dissimilarity measure $d(\cdot,\cdot)$ are given. However, in the classic kNN classifier, the majority vote principle is applied. Since k is an integer, as mentioned earlier, there is a big chance that there is a tie among the largest number of nearest neighbors for more than two classes. For example, in a two-class classification problem, suppose we take $k=4$, then there is a big chance that each class has two observations among the four nearest neighbors. To overcome this difficulty, for a two-class classification problem, k has to be an odd number to break ties. Unfortunately, this strategy does not work for general multi-class classification problems. In our new kNN classifier, on the other hand, the class membership of the new observation is determined by the smallest

average dissimilarity measure among the k nearest neighbors from each class. Since the average dissimilarity measure such as distance or semi-distance is continuous, it is well known that the probability of the equality of two smallest average dissimilarity measures is zero almost surely. Secondly, the classic kNN classifier assigns the class membership of the new observation just based on the majority vote principle and the dissimilarity measures of the k nearest neighbors are not used or used indirectly. In some sense, the values of the dissimilarity measures of the k nearest neighbors are less related as long as the observations are selected as the k nearest neighbors. However, in the new kNN classifier, the values of the dissimilarity measures of the k nearest neighbors for each class are taken into account. The new kNN classifier provides a new look at the use of the nearest neighbors. It is expected that the new kNN classifier should outperform the classic kNN classifier generally. We shall conduct simulation studies to check if this is actually the case.

2.2 Functional dissimilarity measures

In the classic and new kNN classifiers, a functional dissimilarity measure $d(\cdot,\cdot)$ is also needed. It can be distance or semi-distance between two functional observations. Here we review some useful functional dissimilarity measures. For this aim, we first define the L^p-norm of a function. Throughout this paper, let $\mathcal{T}$ be a finite interval, and we use $\|x\|_p$ to denote the L^p-norm of a function $x(t), t \in \mathcal{T}$: $\|x\|_p = \left(\int_{\mathcal{T}} |x(t)|^p dt\right)^{1/p}$, for $p = 1, 2, \ldots$. If $\|x\|_p < \infty$, we say $x(t), t \in \mathcal{T}$ is an L^p-integrable function. In this case, we write $x \in \mathcal{L}^p(\mathcal{T})$ where $\mathcal{L}^p(\mathcal{T})$ denotes the Hilbert space formed by all the L^p integrable functions over $\mathcal{T}$. In particular, $\mathcal{L}^2(\mathcal{T})$ denotes the Hilbert space formed by all the squared integrable functions over $\mathcal{T}$, which is an inner product space. The associated inner-product for any two functions is defined as $< x, y >= \int_{\mathcal{T}} x(t)y(t)dt, x(t), y(t) \in \mathcal{L}^2(\mathcal{T})$. Let $x(t)$ and $y(t)$ be two functional observations defined over a compact set $\mathcal{T}$ which are L^p integrable. Then the L^p-distances can be defined as

$$d_p(x, y) = \|x - y\|_p, \tag{4}$$

for $p = 1, 2, \ldots$. For simplicity, we often use L^1, L^2, and L^∞-distances where $d_\infty(x, y) = \|x - y\|_\infty = \sup_{t\in\mathcal{T}} |x(t) - y(t)|$. When $p = 2$, we may also use $\|\cdot\|$ to denote the L^2-norm for convenience.

To measure the shape similarity between $x(t)$ and $y(t)$, we can use the following functional cosine distance (FCD) proposed by [25]:

$$d_{FCD}(x, y) = 1 - \frac{< x, y >}{\|x\|\|y\|} = \frac{1}{2}\|\tilde{x} - \tilde{y}\|^2, \tag{5}$$

where $\tilde{x}(t) = x(t)/\|x\|$ and $\tilde{y}(t) = y(t)/\|y\|$ denote the normalised versions of $x(t), t \in \mathcal{T}$ and $y(t), t \in \mathcal{T}$, respectively. Note that $\tilde{x}(t)$ is also known as the spatial sign function of $x(t)$ [17], which may be interpreted as the direction of $x(t)$. Thus, the FCD between $x(t)$ and $y(t)$ measures the squared L^2-distance between the directions of $x(t)$ and $y(t)$.

The L^p-distances and the FCD can be implemented easily in the new kNN classifier but they do not take the correlation of a functional observation into account. To partially address this issue, [10] proposed the so-called functional Mahalanobis semi-distance which can also be employed in the new kNN classifier so that the correlation structure of functional observations can be taken into account partially. The functional Mahalanobis semi-distance is defined using a number of the largest eigenvalues and the associated eigenfunctions. Let $y(t) \sim \mathrm{SP}(\eta, \gamma)$. It is well known that when $\gamma(s, t)$ has a finite trace, i.e., $\mathrm{tr}(\gamma) = \int_{\mathcal{T}} \gamma(t, t)dt < \infty$, it has the following singular value decomposition [21, p. 3]: $\gamma(s, t) = \sum_{r=1}^{\infty} \lambda_r \phi_r(s)\phi_r(t)$, where $\lambda_r, r = 1, 2, \ldots$ are the decreasing-ordered eigenvalues of $\gamma(s, t)$, and $\phi_r(t), r = 1, 2, \ldots$ are the associated orthonormal eigenfunctions such that $\int_{\mathcal{T}} \phi_r^2(t)dt = 1$, and $\int_{\mathcal{T}} \phi_r(t)\phi_\ell(t)dt = 0,\ r \neq \ell$. Further, we have $y(t) = \sum_{r=1}^{\infty} \xi_r \phi_r(t)$, where $\xi_r =< y, \phi_r >$, $r = 1, 2, \ldots$ denote the associated principal scores of $y(t)$. Let $x(t)$ be another functional observation whose covariance function is also $\gamma(s, t)$. Then we can also expand $x(t)$ in terms of the eigenfunctions of $\gamma(s, t)$ as $x(t) = \sum_{r=1}^{\infty} \zeta_r \phi_r(t)$, where $\zeta_r =< x, \phi_r >, r = 1, 2, \ldots$ denote the associated principal scores of $x(t)$. Then, the functional Mahalanobis (FM) semi-distance between $x(t)$ and $y(t)$ is given by

$$d_{FM,q}(x, y) = \left[\sum_{r=1}^{q} \lambda_r^{-1}(\zeta_r - \xi_r)^2\right]^{1/2}. \tag{6}$$

Based on the principal scores of $x(t)$ and $y(t)$, [7] defined the following so-called functional principal components (FPC) based semi-distance:

$$d_{FPC,q}(x, y) = \left[\sum_{r=1}^{q} (\zeta_r - \xi_r)^2\right]^{1/2}. \tag{7}$$

In practice, the eigenvalues λ_r's and the principal scores ζ_r's and ξ_r's used in the FM semi-distance (6) and the FPC based semi-distance (7) have to be replaced with their estimates based on the data. For this purpose, we first need to estimate the mean functions $\eta_i(t)$ and the covariance functions $\gamma_i(s, t)$. Based on the training samples of G classes (1), the unbiased estimators of the group mean functions $\eta_i(t)$ are their usual group sample mean functions $\bar{x}_i(t)$. The group covariance functions $\gamma_i(s, t)$ can also be estimated using their usual sample covariance functions $\hat{\gamma}_i(s, t)$. The estimators are given by

$$\begin{aligned} &\hat{\eta}_i(t) = \bar{x}_i(t) = n_i^{-1} \sum_{j=1}^{n_i} x_{ij}(t),\ i = 1, \ldots, G, \quad \text{and} \\ &\hat{\gamma}_i(s, t) = (n_i - 1)^{-1} \sum_{j=1}^{n_i} [x_{ij}(s) - \bar{x}_i(s)][x_{ij}(t) - \bar{x}_i(t)], \\ &i = 1, \ldots, G. \end{aligned} \tag{8}$$

The associated pooled sample covariance function can be obtained as

$$\hat{\gamma}(s,t) = \sum_{i=1}^{G}(n_i - 1)\hat{\gamma}_i(s,t)/(n-G), \tag{9}$$

where $n = n_1 + \cdots + n_G$.

Let $\hat{\lambda}_r$'s and $\hat{\phi}_r(t)$'s denote the estimated decreasing-ordered eigenvalues and eigenfunctions of $\hat{\gamma}(s,t)$. Then the estimated FM semi-distance and the estimated FPC based semi-distance between a new functional observation $x(t)$ and any functional observations $x_{ij}(t)$ from the given G samples (1) can be defined as

$$\begin{aligned} d_{FM,q}(x, x_{ij}) &= \left[\sum_{r=1}^{q} \hat{\lambda}_r^{-1}(\hat{\zeta}_r - \hat{\xi}_{ij,r})^2\right]^{1/2}, \quad \text{and} \\ d_{FPC,q}(x, x_{ij}) &= \left[\sum_{r=1}^{q}(\hat{\zeta}_r - \hat{\xi}_{ij,r})^2\right]^{1/2}, \end{aligned}$$

where $\hat{\zeta}_r =< x, \hat{\phi}_r >$ and $\hat{\xi}_{ij,r} =< x_{ij}, \hat{\phi}_r >$ for $r = 1, \ldots, q$.

Note that the parameter q used in the estimated FM distance and the estimated FPC based distance should also be estimated based on the data. This issue will be handled in the next subsection.

Remark 1. *In this subsection and throughout, for simplicity, we assume that all the individual functions are fully observed without errors over a finite interval $\mathcal{T}$. This means that we implicitly adopt a "smoothing first, then inference" strategy that is widely used in functional data analysis; see [15] and [23] among others. In real data analysis, however, some smoothing techniques, e.g., smoothing splines [21] or local polynomial smoothing [5], have to be employed to reconstruct all the individual functions, especially when the design time points are not the same for all the individual functions. This is because in practice, the functional samples (1) are generally observed discretely on some design time points, often with noise. The reconstructed individual functions are then evaluated at a common grid of design time points so that the above dissimilarity measures can be computed numerically based on the resulting multivariate data. [24] shows that when the individual functions are densely observed, the reconstructed individual functions can be used to replace the underlying individual functions with asymptotically ignorable errors. [10] provided an example where a basis method was employed to implement their functional Mahalanobis semi-distance based classifier for functional data. The dissimilarity measures reviewed in this subsection may be computed directly if all the individual functional observations are observed at a common grid of design time points.*

Remark 2. *Note that the functional dissimilarity measures reviewed in this subsection do not take into account the smoothness of functions directly since as mentioned in Remark 1, we implicitly adopt the "smoothing first, then inference" strategy in this paper so that the smoothness of the individual functions may have already been taken into account in the reconstructing step of the individual functions. As suggested by an anonymous reviewer, it would be interesting to involve derivatives directly in a similarity measure, e.g., under the reproducing kernel Hilbert space if no reconstructing step of the individual functions has been made. This may be a very good research topic for a future study.*

2.3 Parameters selection

When the new kNN classifier is applied with one of the L^p-distances or FCD, the parameter we need to select is the number of nearest neighbors, k. When the new kNN classifier is applied with the FM semi-distance or the FPC based semi-distance, the extra parameter we need to select is the number of principal components, q. In this subsection, we discuss how to select them using the so-called cross-validation approach.

As in the classic kNN classifier, it is necessary to select the number of nearest neighbors, k, properly in the new kNN classifier. If we consider a big training sample with noisy functional observations, we need a larger value of k to reduce the influence of the noisy functional observations. However, increasing the value of k can lead the classifier into taking a long time to run. To determine the best k, a straightforward method is through cross-validation by trying various possible values of k and choose the one with the best performance. Throughout this paper, we choose the M-fold cross-validation to select the optimal value of k. That is, we randomly partition the training sample into M equal sized subsamples, as in [9, chap. 7]. Of the M subsamples, a single subsample is retained as the validation data, and the remaining $M-1$ subsamples are used as training data. We run the classifier based on the training observations to predict the class membership for each observation in the validation data. If the predicted class membership matches its true class membership, we count a correct classification. Otherwise, it is a cross-validation error. Then the cross-validation process is repeated M times, with each of the M subsamples used exactly once as the validation data. We estimate the cross-validation error by taking an average of M cross-validation estimators. We determine the optimal value of k with the smallest cross-validation error rate.

Specially, let $\mathcal{X} = \{x_{ij}, j = 1, \ldots, n_i;\ i = 1, \ldots, G\}$ denote the whole training sample and we randomly partition it into M equal sized subsample, $\mathcal{X}_1, \ldots, \mathcal{X}_M$. For each $\mathcal{X}_m, m = 1, \ldots, M$, we let $\mathcal{X}\backslash\mathcal{X}_m$ denote the remaining training sample after choosing $\mathcal{X}_m$ as the validation data. Then the class membership of x_{ij} when $x_{ij} \in \mathcal{X}_m$ predicted by the new kNN classifier constructed based on $\mathcal{X}\backslash\mathcal{X}_m$ and with k nearest neighbors, i.e., the classification function, can be denoted as

$$\hat{g}_k\left(x_{ij}|\mathcal{X}\backslash\mathcal{X}_m\right). \tag{10}$$

If $\hat{g}_k\left(x_{ij}|\mathcal{X}\backslash\mathcal{X}_m\right) = i$, this is a correction classification; otherwise, this is a misclassification error. The average misclassification error rate based on this M-fold cross-validation

may be denoted as

$$\text{CV}_k = n^{-1} \sum_{m=1}^{M} \sum_{x_{ij} \in \mathcal{X}_m} I\{\hat{g}_k\left(x_{ij} | \mathcal{X} \backslash \mathcal{X}_m\right) \neq i\}, \tag{11}$$

where $n = \sum_{i=1}^{G} n_i$ denotes the total sample size of the training sample and $I\{A\}$ denotes the indicator function of A. Then the optimal value of k is given by $\text{argmin}_{1 \leq k \leq n_{\min}} \text{CV}_k$ where $n_{\min} = \min_{1 \leq i \leq G} n_i$ denotes the minimum group sample size. In practice, the parameter k can take values from 1 to $n_{\min}$.

When the FM semi-distance or FPC based semi-distance is used in the new kNN classifier, we also need to select the number of principal components, q, so that the resulting new kNN classifier has the best performance. In this case, the classification function (10) and the cross-validation average misclassification error rate (11) depend on both k and q and may be denoted as $\hat{g}_{k,q}\left(x_{ij} | \mathcal{X} \backslash \mathcal{X}_m\right)$ and $\text{CV}_{k,q}$ respectively instead. The optimal values of k and q are then given by $\text{argmin}_{k,q} \text{CV}_{k,q}$. Similarly, the parameter k can take values from 1 to $n_{\min}$ while the parameter q can take values from 1 to some q_0 which is specified by the user. A possible q_0 may be chosen such that the sum of the first q_0 eigenvalues $\sum_{r=1}^{q_0} \hat{\lambda}_r$ is about 85% to 95% of the total variation, i.e., $\text{tr}(\hat{\gamma})$.

3. ADJUSTED NEW KNN CLASSIFIER FOR IMBALANCED DATA

The new kNN classifier proposed in Section 2 works well for balanced data. However, it may fail when we have the class imbalance problem. In this section, we will study an adjusted new kNN classifier for imbalanced data.

Learning from imbalanced data has been identified as one of the 10 most challenging problems in data mining research [22]. A data set is imbalanced if the instances of one class are far less in number than the instances of another class. When a class has few observations, it is likely that these observations will be further apart than the observations of another class with more observations. At this time, the classic kNN classifier will fail for imbalanced classification and so will the new kNN classifier proposed in Section 2.

Figure 1 shows an artificial two-class imbalance classification problem, where the majority class is represented by blue circles and the minority class by red triangles. There are 300 observations in the majority class and 30 observations in the minority class. As can be seen from the figure, since the class instance ratio is 10 to 1, the majority class dominates the neighbors of the observations in the minority class. Thus, it is hard to classify the minority class correctly using the classic kNN classifier.

When we use the new kNN classifier proposed in the previous section, we select the same number of neighbors from each class. However, due to the imbalanced sample sizes, the average distance of the k nearest neighbors for the majority

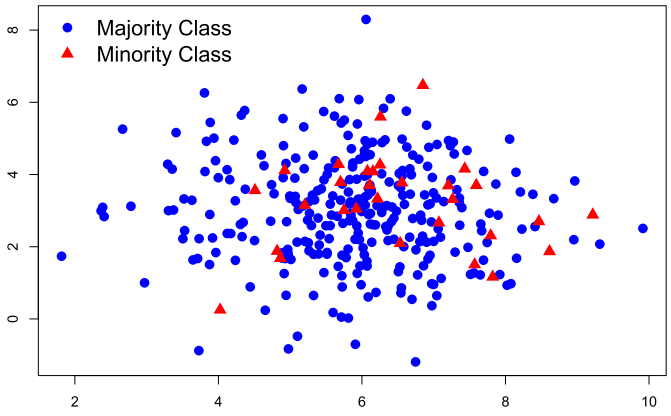

Figure 1. An imbalanced classification example.

class should be often smaller than the average distance for the minority class. Therefore, our new kNN classifier may fail for the imbalance classification problems too. To address this issue, we change the number of neighbors chosen from each class to reduce the imbalance between classes so that the number of neighbors chosen is proportional to the class size. Suppose we have the G functional samples in (1) and a new coming observation $x(t)$. Then for each class i, we select k_i functional observations which are closest to the new observation $x(t)$ according to a distance (or semi-distance) $d(x, y)$ between two functional observations $x(t)$ and $y(t)$, say $x_{ij_1}(t), \ldots, x_{ij_{k_i}}(t),\ i = 1, \ldots, G$ such that

$$d(x, x_{ij_1}) \leq \cdots \leq d(x, x_{ij_{k_i}}),\ i = 1, \ldots, G. \tag{12}$$

In order to reduce the imbalance between classes, we set $k_i = [\frac{n_i}{n_{\min}} k_{\min}], k_{\min} = 1, \ldots, n_{\min}, i = 1, \ldots, G$, where $[x]$ returns the nearest integer to x and $k_{\min}$ is the number of nearest neighbors from the class whose sample size is the smallest. If $n_1 = \cdots = n_G$, it reduces to the usual new kNN classifier for balanced classification problems. The averages of the distances (12) of the k_i neighbors for class i are computed as

$$\tau_i^* = k_i^{-1} \sum_{\ell=1}^{k_i} d(x, x_{ij_\ell}),\ i = 1, \ldots, G. \tag{13}$$

The class membership of the new functional observation $x(t)$ is assigned to class g if $g = \text{argmin}_{i=1,\ldots,G} \tau_i^*$.

For balanced classification problems, we use the classification accuracy or misclassification error rate to assess the classification performance. However, for imbalanced classification problems, classification accuracy is no longer a proper measure since the minority class has very little impact on the accuracy compared with the majority class. A typical example is the disease diagnostic problem. The disease cases are usually quite rare and there is a large number of patients who do not have that disease. The goal is to detect

those people with diseases. The classification accuracy may be close to 1 as the number of disease cases is small, but it cannot detect the diseases. A favorable classifier is one that provides a higher identification rate on the disease category. To consider the performance of majority class and minority class simultaneously, [14] suggested the G-mean for two-class scenario which is the geometric mean of the accuracies on two classes. [20] extended this measure to multi-class scenario. They defined the G-mean as the geometric mean of accuracies on all classes:

$$\text{G-mean} = \left(\prod_{i=1}^{G} \text{ACC}_i\right)^{\frac{1}{G}}, \tag{14}$$

where ACC_i is the accuracy on the class i, $i = 1, \ldots, G$. As each ACC_i representing the classification performance of class i is equally accounted, G-mean is capable to measure the balanced performance among classes of a classification output [20]. The larger the G-mean is, the better the classifier is.

As in the new kNN classifier proposed in the last section, it is necessary to select the number of nearest neighbors in each class properly in this adjusted new kNN classifier as well. In fact, we only need to select the value of $k_{\min}$, that is, the number of nearest neighbors selected from the class whose sample size is the smallest. We still use the M-fold cross-validation approach to select the optimal value of $k_{\min}$. However, it is not proper to estimate the cross-validation error by taking an average of M cross-validation estimators. We determine the optimal value of $k_{\min}$ with the largest G-mean in (14). Based on the training sample $\mathcal{X}\backslash\mathcal{X}_m$, the class membership of x_{ij} when $x_{ij} \in \mathcal{X}_m$ predicted by the adjusted new kNN classifier with $k_i = [\frac{n_i}{n_{\min}} k_{\min}]$ nearest neighbors from class i, can be denoted as

$$\hat{g}^*_{k_{\min}}(x_{ij}|\mathcal{X}\backslash\mathcal{X}_m). \tag{15}$$

If $\hat{g}^*_{k_{\min}}(x_{ij}|\mathcal{X}\backslash\mathcal{X}_m) = i$, this is a correct classification; otherwise, this is a misclassification error. The G-mean based on the M-fold cross-validation approach may then be denoted as

$$\text{CV}^*_{k_{\min}} = \left[\prod_{i=1}^{G} \frac{1}{n_i} \sum_{m=1}^{M} \sum_{x_{ij} \in \mathcal{X}_m} I\{\hat{g}^*_{k_{\min}}(x_{ij}|\mathcal{X}\backslash\mathcal{X}_m) = i\}\right]^{\frac{1}{G}}. \tag{16}$$

Then the optimal value of $k_{\min}$ is given by $\text{argmax}_{1 \le k_{\min} \le n_{\min}} \text{CV}^*_{k_{\min}}$.

When the FM semi-distance or FPC based semi-distance is used in the adjusted new kNN classifier, we also need to select the number of principal components, q, so that the resulting adjusted new kNN classifier has the best performance. In this case, the classification function (15) and the cross-validation G-mean (16) depend on both $k_{\min}$ and q and may be denoted as $\hat{g}^*_{k_{\min},q}(x_{ij}|\mathcal{X}\backslash\mathcal{X}_m)$ and $\text{CV}^*_{k_{\min},q}$ respectively instead. The optimal values of $k_{\min}$ and q are then given by $\text{argmax}_{k_{\min},q} \text{CV}^*_{k_{\min},q}$. Similarly, the parameter $k_{\min}$ can take values from 1 to $n_{\min}$ while the parameter q can take values from 1 to some q_0 which is specified by the user.

It is interesting to compare the proposed new kNN and adjusted new kNN classifiers. First of all, the new kNN classifier works for balanced classification problems. It selects the same number of nearest neighbors from each class. However, if the number of instances of one class is much larger than that of another class, the proposed new kNN classifier is hard to classify the minority class correctly. Thus, we propose the adjusted new kNN classifier for imbalanced classification problems. The number of nearest neighbors from class i is proportional to the prior probability of this class. By taking the prior probability into account, the imbalance between different classes is reduced. Secondly, both the proposed new kNN and adjusted new kNN classifiers need to select the number of nearest neighbors. For the new kNN classifier, the number of nearest neighbors from each class is selected by the M-fold cross-validation approach described in Section 2.3. For the adjusted new kNN classifier, since it is not accurate to use accuracy as a measure of the classification performance, we still use the M-fold cross-validation approach to choose the number of nearest neighbors from each class, but the classification function and the cross-validation G-mean are given by (15) and (16), respectively. We shall conduct simulation studies to check the performance of this adjusted new kNN classifier for imbalanced classification problems.

4. SIMULATION STUDIES

It is of interest to check if the proposed new kNN classifier works well for balanced classification problems and the proposed adjusted new kNN classifier works well for imbalanced classification problems compared with the classic kNN classifier and some non-kNN classifiers. In this section, simulation studies are presented to compare the proposed new kNN and adjusted new kNN classifiers against the classic kNN classifier with various dissimilarity measures, including the L^p-distances (4) for $p = 1, 2, \infty$, functional cosine distance (5), functional Mahalanobis (FM) semi-distance assuming a common covariance operator (6) and functional principal components (FPC) based semi-distance assuming a common covariance operator (7), labeled by L^1, L^2, L^∞, FCD, FPC and FM. Two-class and three-class classification problems are considered respectively. Furthermore, an additional simulation study is presented to compare the proposed new kNN and adjusted new kNN classifiers against the classic kNN classifier with various dissimilarity measures and the support vector machine classifier as well.

4.1 Two-class classification

In this subsection, functional data for two-class classification problems are generated under four different scenarios. In the first scenario, two functional samples are generated

from two Gaussian processes defined over the unit interval $[0,1]$, with different group mean functions $\eta_1(t) = 25t^{1.1}(1-t)$ and $\eta_2(t) = 25t(1-t)^{1.1}$ but their covariance functions $\gamma_1(s,t)$ and $\gamma_2(s,t)$ are the same, denoted as $\gamma(s,t)$ whose eigenfunctions are given by $\phi_r(t) = \sqrt{2}\sin(r\pi t), r = 1,2,\ldots$ and the associated eigenvalues are given by $\lambda_r = 1/(r\pi)^2$, for $r = 1,2,\ldots$. The generated functions are evaluated at 1000 equidistant time points over $[0,1]$. In the second scenario, the functions are generated in a similar way except that the two covariance functions $\gamma_1(s,t)$ and $\gamma_2(s,t)$ are not the same. Although their eigenfunctions are the same as those defined in the first scenario, their eigenvalues are given by $\lambda_{1r} = 1/(r\pi)^2$ and $\lambda_{2r} = 2/(r\pi)^2$, for $r = 1,2,\ldots$ respectively. In the third and fourth scenarios, the functions are generated in a similar way as in the first two scenarios respectively except that the two Gaussian processes are replaced with two standardized exponential processes of rate 1, with the same group mean functions and the group covariance functions.

4.1.1 Balanced cases

For balanced classification problems, we only need to compare the performance of the proposed new kNN classifier and the classic kNN classifier since in this case, the new kNN and adjusted new kNN classifiers are equivalent. Under each scenario, two functional samples of equal sizes 100 are generated. The training sample is formed via selecting 50 functions from each sample so that the whole training sample consists of 100 functional observations. The remaining functional observations from the two functional samples form the test sample. The training sample is used to determine the tuning parameters. We use the 10-fold cross-validation approach described in Section 2.3 to select the tuning parameters involved in the classifiers, including the number of principal components q and/or the number of nearest neighbors k. We set the number of nearest neighbors k to range from 1 to 19 for simplicity, and set k to be odd numbers only to avoid the tie problem for the classic kNN classifier. At the same time, we set the number of principal components q to range from 1 to q_0 where q_0 may be chosen such that the sum of the first q_0 eigenvalues of the pooled sample covariance function $\hat{\gamma}(s,t)$ is about 95% of the total variation given by $\mathrm{tr}(\hat{\gamma})$. The test sample is used to compute the classification accuracy. The simulation process is repeated 1000 times so that 1000 classification accuracy values can be recorded.

Figure 2 displays the boxplots of the 1000 classification accuracy values of the classic and new kNN classifiers under the four scenarios and various dissimilarity measures. For each dissimilarity measure, the two boxes are associated with the classic and new kNN classifiers respectively. It is seen that in terms of classification accuracy, the proposed new kNN classifier generally performs better than the classic kNN classifier under each scenario and each dissimilarity measure, showing that the proposed new kNN classifier does work well in this simulation study.

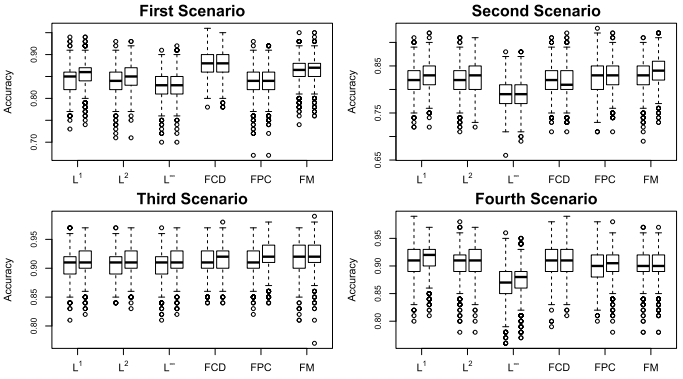

Figure 2. Boxplots of the classification accuracy values of the classic and new kNN classifiers under the four scenarios. For each dissimilarity measure, the two boxes are associated with the classic and new kNN classifiers respectively. The L^p-distances for $p = 1,2,\infty$, functional cosine semi-distance (FCD), functional Mahalanobis (FM) semi-distance, and functional principal components (FPC) are defined in (4), (5), (6), and (7) respectively.

4.1.2 Imbalanced cases

For imbalanced classification problems, we now compare the proposed new kNN classifier, the adjusted new kNN classifier against the classic kNN classifier. We consider the same simulation setup as in the previous subsection. Since in the previous subsection, the classic kNN and new kNN classifiers with the FM semi-distance perform well in all the four scenarios, for simplicity, in this subsection, we compare the classic kNN, new kNN and adjusted new kNN classifiers only with the FM semi-distance. In order to generate imbalanced classification problems, we generate $N_1 = cN_2$ functions from the first stochastic process mentioned in the functional data generation setting of the previous subsection and $N_2 = 50$ functions from the second process, where the sample size ratio constant $c = 1,2,4,10$ and 20, representing different degrees of imbalance of the sample sizes of the training and test samples. The training sample is formed via selecting $n_1 = N_1/2$ functions from the first sample and $n_2 = N_2/2$ functions from the second sample while the remaining functions form the test sample. It is of interest to evaluate the performance of the classic kNN, new kNN and adjusted new kNN classifiers for different c. Tuning parameters involved in the classic kNN and new kNN classifiers are selected by the same approaches as described in the previous subsection. In addition, we also use the 10-fold cross-validation approach described in Section 3 to choose the tuning parameters involved in the adjusted new kNN classifier. We also set the number of principal components q to range from 1 to q_0, and the number of nearest neighbors from the second process $k_{\min}$ to range from 1 to 20, for simplicity.

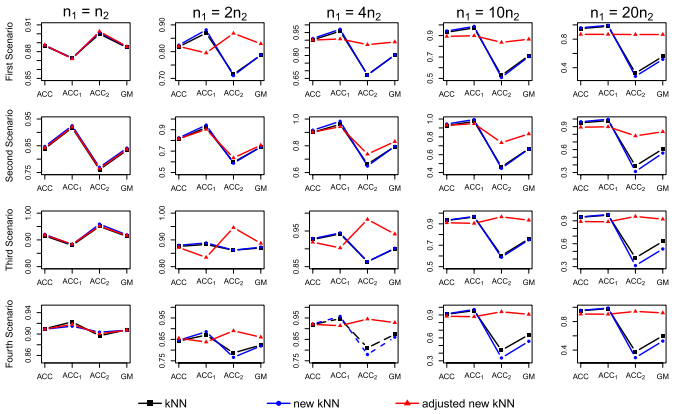

Figure 3. Performances of the classic kNN, new kNN and adjusted new kNN classifiers under the four scenarios and five different c values. For each classifier, the mean classification accuracy values for the whole test sample, the first class of the test sample, the second class of the test sample, and the average G-mean value of the whole test sample are labeled by ACC, ACC_1, ACC_2, and GM, respectively.

Figure 3 displays the mean classification accuracy values of the classic kNN, new kNN and adjusted new kNN classifiers under the four scenarios and five different c values with 1000 Monte Carlo simulation runs. For each classifier, the mean classification accuracy values for the whole test sample, the first class of the test sample, the second class of the test sample, and the average G-mean value of the whole test sample are calculated and labeled by ACC, ACC_1, ACC_2, and GM, respectively.

It is seen that when the two classes are balanced ($n_1 = n_2$), the performances of the three classifiers are generally comparable as expected. However, when the two classes are imbalanced ($n_1 > n_2$), if we use the classic and new kNN classifiers, the mean classification accuracy values on the first class, ACC_1, are always larger than those on the second class, ACC_2. As c increases, ACC_1 becomes larger while ACC_2 becomes smaller. In particular, when $n_1 = 10n_2$ and $n_1 = 20n_2$, ACC_1 is close to 1 while ACC_2 is less than 0.60. This is not surprise since when n_1 is much larger than n_2, the number of nearest neighbors from the first class is naturally much larger than that from the second class. It is then much easier to assign the new observation to the first class than to the second class. As a result, ACC_1 will be much larger than ACC_2 as expected. This explains why the performances of the classic and new kNN classifiers are often dominated by their performances on the majority class when the two classes are seriously imbalanced in sample sizes. This problem can be overcome by the adjusted new kNN classifier, as seen from Figure 3. The adjusted new kNN classifier chooses the number of nearest neighbors proportional to the respective sample size so that ACC_1 becomes smaller and ACC_2 becomes larger than those respective values of the classic and new kNN classifiers. Consequently, ACC_1 and ACC_2 become more balanced than in those cases of the classic and new kNN classifiers. Unfortunately, as seen from Figure 3, this advantage of the adjusted new kNN classifier is not reflected from the classification accuracy values of the whole test sample (ACC). Therefore, it is less appropriate to assess the performance of the adjusted new kNN classifier using ACC. Rather, we should use the G-mean defined in (14). In fact, as seen from Figure 3, in terms of G-mean, the adjusted new kNN classifier always performs better than the classic and new kNN classifiers when the two-classes are imbalanced in sample sizes.

4.2 Three-class classification

In this subsection, we investigate the performances of the classic kNN, new kNN and adjusted new kNN classifiers on three-class classification problems. For simplicity, we use only the first scenario as described in the previous subsection. That is, the functional samples are generated from Gaussian processes and the classes differ only in mean functions. For this purpose, we set $\eta_1(t) = 25t^{\alpha}(1-t)$ and $\eta_2(t) = 25t(1-t)^{\alpha}$ where $\alpha = 1.1, 1.2$ and 1.5, representing the shape differences of the first two mean functions. For convenience, we set the third mean function as $\eta_3(t) = [\eta_1(t) + \eta_2(t)]/2$, the average of the first two mean functions. As before, the generated functions are evaluated at 1000 equidistant time points over $[0, 1]$.

4.2.1 Balanced cases

As in the two-class classification, for balanced classification problems, we only need to compare the performance of the classic and new kNN classifiers. For simplicity, we set the training and test samples to have 50 observations in each class. The same approaches as in Section 4.1.1 are used to choose the tuning parameters, but we no longer set the number of nearest neighbors k to be odd since for three-class classification problems, this does not help solving the tie problem of the classic kNN classifier. The simulation process is repeated 1000 times and the boxplots of the 1000 classification accuracy values on the whole test samples with different values of α are shown in Figure 4. As before, for each dissimilarity measure, the two boxes are associated with the classic and new kNN classifiers respectively.

It is seen from Figure 4 that the proposed new kNN classifier generally outperforms the classic kNN classifier, especially when the three classes are difficult to be classified ($\alpha = 1.1$ and $\alpha = 1.2$). Note that with increasing the values of α, the three classes become easier to be classified due to the fact that the mean functions of the three classes become more different in shape.

4.2.2 Imbalanced cases

To generate three-class imbalanced classification problems, we set each of the training and test samples to have 50 observations from Class 1, 100 observations from Class

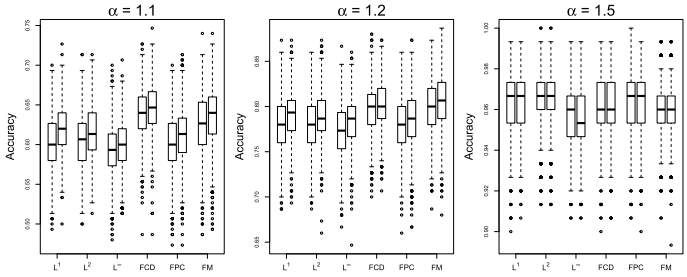

Figure 4. Boxplots of the classification accuracy values of the classic and new kNN classifiers with different values of α. For each dissimilarity measure, the two boxes are associated with the classic and new kNN classifiers respectively.

2, and 200 observations from Class 3. As in Section 4.1.2, we compare the performance of the classic kNN, new kNN and adjusted new kNN classifiers with different values of α. The same approaches as in the previous subsections are used to choose the tuning parameters used in the classic kNN and new kNN classifiers. For the tuning parameters used in the adjusted new kNN classifier, we use the 10-fold cross-validation approach described in Section 3. We set the number of nearest neighbors from Class 1 to range from 1 to 19 and the number of principal components q to range from 1 to q_0 as well where q_0 is chosen using the same method as described in Section 4.1.1. This process is repeated 1000 times. Figure 5 shows the boxplots of these 1000 classification accuracy values of the three classifiers for various dissimilarity measures and different values of α. For each dissimilarity measure, the three boxes are associated with the classic kNN, new kNN and adjusted new kNN classifiers, respectively.

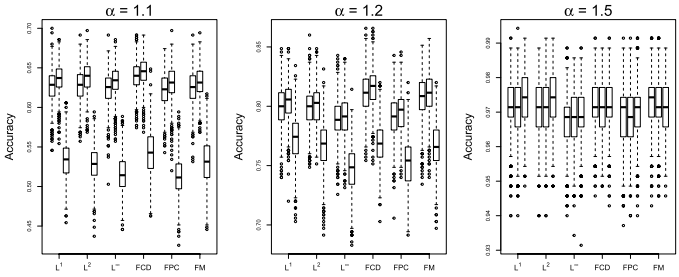

Figure 5. Boxplots of the classification accuracy values of the classic kNN, new kNN, and adjusted new kNN classifiers for various dissimilarity measures and different values of α. For each dissimilarity measure, the three boxes are associated with the classic kNN, new kNN and adjusted new kNN classifiers, respectively.

It is seen from Figure 5 that in terms of classification accuracy, when $\alpha = 1.1$ and $\alpha = 1.2$, the new kNN classifier generally outperforms the classic kNN classifier while the latter generally outperforms the adjusted new kNN classifier and when $\alpha = 1.5$, the three classifiers perform similarly.

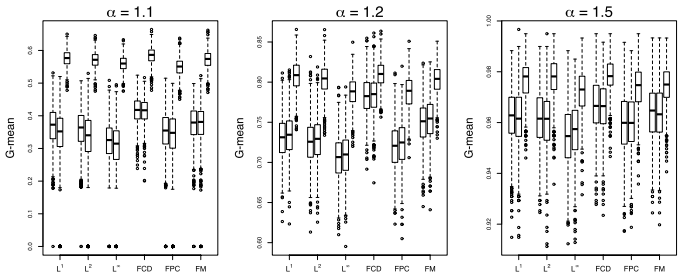

Figure 6. Boxplots of the G-mean values of the classic kNN, new kNN and adjusted new kNN classifiers with different values of α. For each dissimilarity measure, the three boxes are associated with the classic kNN, new kNN and adjusted new kNN classifiers, respectively.

As mentioned in Section 4.1.2, for imbalanced classification problems, it is less appropriate to assess the performances of the three classifiers using the classification accuracy. Rather, we should use the G-mean defined in (14). Figure 6 displays the boxplots of the 1000 G-mean values of the three classifiers with different values of α. It is seen that in terms of the G-mean values, the adjusted new kNN classifier indeed outperforms the classic and new kNN classifiers especially when the mean functions of the three classes are more similar to each other (i.e., when $\alpha = 1.1$ and $\alpha = 1.2$).

4.3 A comparison with an non-kNN classifier

As suggested by an anonymous reviewer, it is of interest to compare the new kNN and adjusted new kNN classifiers with some non-kNN classifiers. In this subsection, we present such a simulation study using the simulation setup of the two-sample imbalanced cases as described in Section 4.1.2 to compare the proposed new kNN and adjusted new kNN classifiers against the classic kNN classifier with various dissimilarity measures and the support vector machine (SVM) classifier, one of the well-known and accurate non-kNN classifiers. For simplicity, we consider only the first scenario where the two functional samples are generated from Gaussian processes and the two classes differ only in mean functions. As before, the simulation process is repeated 1000 times. The boxplots of the 1000 classification accuracy values and 1000 G-mean values of the classifiers under consideration with different sample size ratios are displayed in Figure 7.

From the left panels of Figure 7, it is seen that in terms of the classification accuracy values, the new kNN classifier generally outperforms the classic kNN classifier under each setting. This conclusion is consistent with what we have drawn from the previous two simulation studies. It is also seen that the new kNN classifier is also comparable with the SVM classifier under each setting although

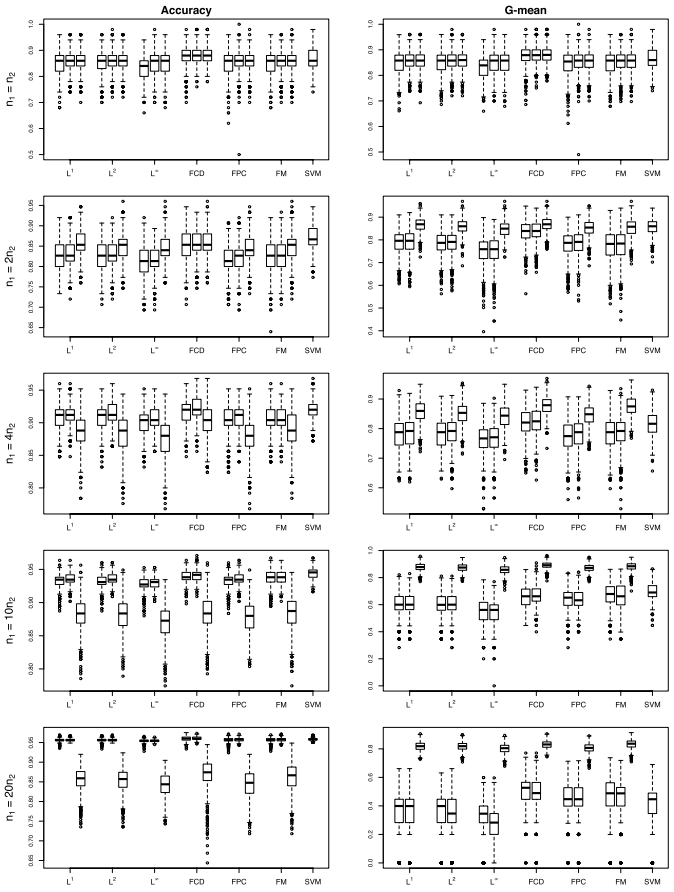

Figure 7. Boxplots of the classification accuracy values (left panels) and G-mean values (right panels) of the classic kNN, new kNN, adjusted new kNN classifiers for various dissimilarity measures and the SVM classifier under five different sample size ratios. For each dissimilarity measure, the three boxes are associated with the classic kNN, new kNN and adjusted new kNN classifiers, respectively and in each panel, the last box is associated with the SVM classifier.

the SVM classifier is much more sophisticated and time-consuming than the new kNN classifier. As expected, the adjusted kNN classifier is comparable with other classifiers when $n_1 = n_2$ and it seems outperforms other classifiers slightly when $n_1 = 2n_2$. However, its performance is getting worse with increasing the value of the sample size ratio. This indicates that when the two samples are imbalanced seriously, the classification accuracy value may be not an appropriate criterion for measuring the performance of a classifier.

From the right panels of Figure 7, it is seen that in terms of the G-mean values, the new kNN classifier generally outperforms the classic kNN classifier under each setting except when $n_1 = 20n_2$ and when the L^∞ dissimilarity measure is used. It is also seen that the new kNN classifier performs slightly worse than the SVM classifier when $n_1 \leq 4n_2$ but it also performs slightly better than the SVM classifier otherwise. As expected, the adjusted kNN classifier generally outperforms other classifiers under each setting. This indicates that when the two samples are imbalanced seriously, the G-mean value may be a better criterion for measuring the performance of a classifier. This is consistent with what we observed in the previous two simulation studies.

5. APPLICATIONS TO REAL-LIFE DATA

In this section, we present the applications of the proposed new kNN and adjusted new kNN classifiers, together with the classic kNN classifier, to two real-life datasets.

5.1 Tecator data

The Tecator dataset has been considered by [6] and [16] to illustrate their classification methodologies. It is available at http://lib.stat.cmu.edu/datasets/tecator. The dataset consists of 215 near-infrared absorbance spectra of meat samples. For each meat sample, the measurements consist of a 100-channel spectrum of absorbance spectra in the wave length range 850-1050 nm and the contents of moisture (water), fat and protein. Of interest is a two-class classification problem, aiming to classify a new meat sample into one of two classes with one class of meat samples with a high fat content ($\geq 20\%$) and the other class with a low fat content ($< 20\%$). Among $N = 215$ meat samples, $N_1 = 77$ meat samples from the first class and $N_2 = 138$ meat samples from the second class.

Figure 8 displays the raw spectrometric curves (1st row) of the Tecator data and their first (2nd row) and second derivative curves (3rd row) for the high fat content class (left panels) and the low fat content class (right panels). It is seen that the raw spectrometric curves of the two classes are similar in shape but their first and second derivative curves are less similar. This implies that it may be more effective to classify the Tecator data using their first and second derivative curves than using their raw spectrometric curves. Further, as the number of meat samples in the high fat content class is about half of that in the low fat content class, the two-class classification problem is somewhat unbalanced in sample sizes. It is then of interest to check the performance of the classic, new, and adjusted new kNN classifiers in this example.

We randomly split the 215 raw spectrometric curves of the Tecator data (and their first and second derivative curves respectively) into a training sample of size $n = 120$ and a test sample of size 95. For the three classifiers, we set the number of functional principal components to range up to 15 and the number of nearest neighbors to range up to 9. For the classic and new kNN classifiers, the number of functional principal components and the number of nearest neighbors are determined by the 10-fold cross-validation approach as described in Section 2.3 while for the adjusted new kNN classifier, they are selected by the 10-fold cross-validation approach as described in Section 3. We then com-

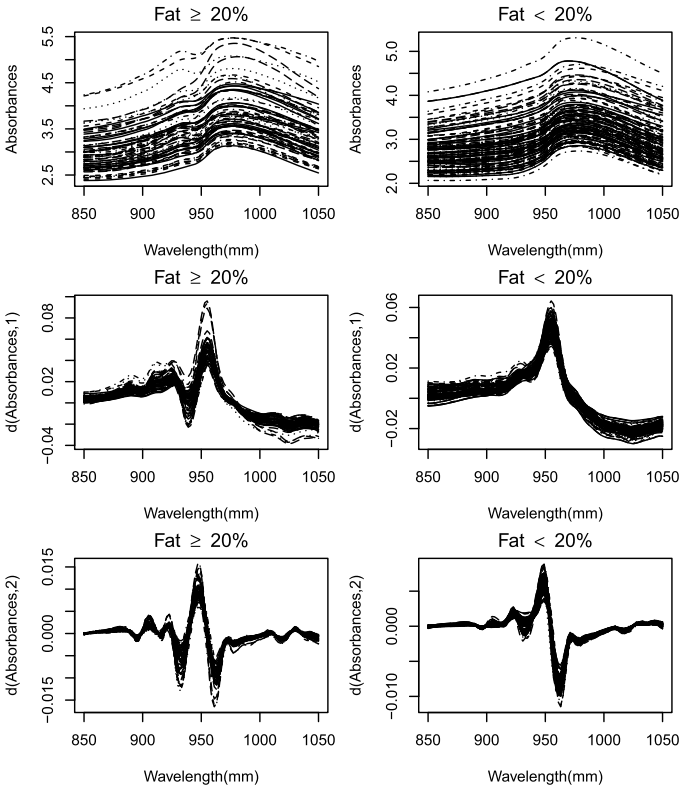

Figure 8. Raw spectrometric curves (1st row) of the Tecator data and their first (2nd row) and second derivative curves (3rd row) for the high fat content class (left panels) and the low fat content class (right panels).

pute the classification accuracy values of the three classifiers. We repeat this process 1000 times so that 1000 classification accuracy values for each classifier are obtained.

Table 1 displays the means and standard deviations (in parentheses) of the 1000 classification accuracy values (in percent) of the three classifiers using the raw spectrometric curves of the Tecator data and their first and second derivative curves respectively. With the same dissimilarity measure, the largest mean classification accuracy value of the classifiers is in boldface. It is seen that in terms of mean classification accuracy, the new kNN and adjusted new kNN classifiers generally perform better than or are largely comparable with the classic kNN classifier, regardless if we use the raw spectrometric curves or their first and second derivative curves. It is also seen that with the same dissimilarity measure, the three classifiers generally have much larger mean classification accuracy values for the first and second derivative curves than those for the raw spectrometric curves of the Tecator data. This shows that the three classifiers indeed perform better when the first and second derivative curves are used than when the raw spectrometric curves of the Tecator data are used. This is consistent with what we observed from Figure 8, as we mentioned earlier.

It is sometimes interesting to compare the best performance of a classifier with various dissimilarity measures. The largest mean classification accuracy values of the classic kNN classifier with the raw spectrometric curves and their first and second derivative curves are 94.79%, 98.09%, and 97.88% respectively. For the new kNN classifier, these three numbers are 95.20%, 98.05%, and 98.05% respectively, while for the adjusted new kNN classifier, they are 95.21%, 97.95%, and 98.40%, respectively. These results are comparable with or even better than the mean classification accuracy values of the support vector machine (SVM) classifier of [16] although the SVM classifier is generally more sophisticated than the classic, new, and adjusted new kNN classifiers. In fact, under an experiment with a setup similar to ours, [16] reported that the mean classification accuracy values of the SVM classifier with the raw spectrometric curves and their second derivative curves are 96.62% and 96.72% respectively.

As mentioned before, when the classification problem is unbalanced in sample sizes, it is more appropriate to compare the performances of the three classifiers using G-mean values. Table 2 displays the means and standard deviations (in parentheses) of the 1000 G-mean values (in percent) of the classic, new, and adjusted new kNN classifiers with the raw spectrometric curves of the Tecator data and their first and second derivative curves respectively. Again, for the same dissimilarity measure, the largest G-mean value of the three classifiers is in boldface. It is seen that with the same dissimilarity measure, the adjusted new kNN classifier has larger average G-mean values than the new kNN classifier most of time while the latter has larger average G-mean values than the classic kNN classifier most of time. This again shows that the proposed new kNN and adjusted kNN classifiers outperform the classic kNN classifier generally.

5.2 Corneal surface data

The corneal surface data set was from the keratoconus study, a consulting project with Ms. Nancy Tripoli and Dr. Kenneth L. Cohen of Department of Ophthalmology, University of North Carolina at Chapel Hill. In this corneal surface data set, 150 corneal surfaces were divided into four groups. The normal cornea group consists of 43 healthy corneas while the unilateral suspect, suspect map and clinical keratoconus cornea groups respectively consist of 14, 21, and 72 disease corneas which are misshaped in some degree.

Figure 9 displays a corneal surface from each of the four cornea groups. To apply the classic, new, and adjusted new kNN classifiers to the above corneal data, we fitted the corneal surfaces using the method described in [18] so that each of the corneal surfaces is represented by a feature vector of length 2000. The feature vectors of the whole corneal data set are displayed in Figure 10. It is seen that there are two outliers in the unilateral suspect cornea group (right upper panel), thus we delete them before classification. Now, we have 43 feature vectors in the normal cornea group, 12

Table 1. Means and standard deviations (in parentheses) of the 1000 *classification accuracy values (in percent) of the classic kNN, new kNN and adjusted new kNN classifiers using the raw spectrometric curves of the Tecator data and their first and second derivative curves, respectively*

Curve	Method	L^1		L^2		L^∞		FCD		FPC		FM	
Raw	kNN	76.83	(4.79)	**79.20**	(4.55)	84.63	(4.32)	92.89	(2.49)	78.95	(4.82)	94.79	(2.37)
	new kNN	**77.26**	(4.57)	78.90	(4.61)	**84.64**	(3.98)	**93.61**	(2.37)	**79.44**	(4.30)	95.20	(2.14)
	adjusted new kNN	74.15	(4.61)	76.51	(4.70)	83.02	(4.55)	93.16	(2.40)	76.50	(4.59)	**95.21**	(2.19)
1st Derivative	kNN	97.18	(1.55)	97.06	(1.58)	96.68	(1.62)	**98.09**	(1.33)	97.25	(1.45)	96.92	(1.93)
	new kNN	97.46	(1.43)	97.33	(1.44)	96.95	(1.48)	98.05	(1.27)	97.44	(1.51)	**96.98**	(1.85)
	adjusted new kNN	**97.67**	(1.41)	**97.42**	(1.44)	**97.06**	(1.39)	97.95	(1.42)	**97.52**	(1.47)	96.91	(1.91)
2nd Derivative	kNN	97.88	(1.51)	97.49	(1.62)	97.30	(1.53)	**97.84**	(1.38)	97.84	(1.66)	96.99	(1.99)
	new kNN	97.86	(1.43)	97.58	(1.60)	**97.31**	(1.53)	97.79	(1.36)	98.05	(1.55)	**97.33**	(1.95)
	adjusted new kNN	**98.40**	(1.30)	**97.96**	(1.47)	97.28	(1.47)	97.82	(1.33)	**98.37**	(1.54)	96.98	(2.05)

Table 2. Means and standard deviations (in parentheses) of the 1000 *G-mean values (in percent) of the classic, new, and adjusted new kNN classifiers with the raw spectrometric curves of the Tecator data and their first and second derivative curves respectively*

Curves	Method	L^1		L^2		L^∞		FCD		FPC		FM	
Raw	kNN	72.76	(7.60)	76.23	(6.77)	82.16	(5.70)	91.12	(3.34)	75.82	(7.03)	93.29	(3.25)
	new kNN	74.39	(5.60)	76.43	(5.83)	82.47	(4.84)	92.06	(2.96)	**77.09**	(5.37)	93.80	(2.93)
	adjusted new kNN	**74.46**	(4.46)	**76.76**	(4.60)	**83.24**	(4.38)	**92.25**	(2.85)	76.86	(4.46)	**94.20**	(2.80)
1st Derivative	kNN	96.60	(2.01)	96.40	(2.08)	95.86	(2.13)	97.43	(1.79)	96.59	(1.87)	95.90	(2.48)
	new kNN	96.98	(1.76)	96.78	(1.78)	96.24	(1.86)	97.39	(1.67)	96.94	(1.78)	96.01	(2.36)
	adjusted new kNN	**97.47**	(1.56)	**97.10**	(1.67)	**96.61**	(1.67)	**97.50**	(1.77)	**97.26**	(1.64)	**96.16**	(2.33)
2nd Derivative	kNN	97.23	(1.96)	96.67	(2.08)	96.46	(1.97)	97.15	(1.83)	97.28	(2.11)	96.21	(2.53)
	new kNN	97.31	(1.82)	96.86	(1.99)	96.59	(1.85)	97.09	(1.75)	97.59	(1.92)	**96.70**	(2.38)
	adjusted new kNN	**98.09**	(1.58)	**97.49**	(1.77)	**96.84**	(1.66)	**97.40**	(1.68)	**98.10**	(1.76)	96.43	(2.50)

feature vectors in the unilateral suspect cornea group, 21 feature vectors in the suspect map cornea group and 72 features in the clinical keratoconus cornea group. It is seen from Figure 9 that the corneal surfaces from the unilateral suspect and suspect map groups are quite similar in shape. A preliminary study also indicated that almost all the unilateral suspect corneas are classified into the suspect map cornea group. To overcome this difficulty, we combined the unilateral and suspect map cornea groups into one group so that we now have 43 feature vectors in the normal cornea group, 33 feature vectors in the combined suspect cornea group, and 72 feature vectors in the clinical keratoconus cornea group.

To compute the classification accuracy values of the classic, new, and adjusted new kNN classifiers for the corneal data, we randomly split the 148 feature vectors into a training sample and a test sample. The training sample contains 26 feature vectors from the normal cornea group, 20 feature vectors from the combined suspect cornea group, and 43 feature vectors from the clinical keratoconus cornea group. The remaining feature vectors form the test sample. As before, for the classic kNN classifier, we set the number of nearest neighbors to range up to 7 and the number of principal components to range up to 9. For the new kNN classifier, we set the number of nearest neighbors from each class to range up to 5 and the number of principal components to range up to 9. The number of nearest neighbors and the number of principal components are selected by the 10-fold cross-validation approach described in Section 2.3. For the adjusted new kNN classifier, we set the number of nearest neighbors and the number of principal components to range up to 4 and 9 respectively. The number of nearest neighbors and the number of principal components are selected by the 10-fold cross-validation approach described in Section 3. The process is repeated 1000 times.

Table 3 displays the mean classification accuracy values and their standard deviations (in parentheses) of the three classifiers for the corneal surface data. With the same dissimilarity measure, the largest mean classification accuracy value of the classifiers is in boldface. It is seen that in terms of mean classification accuracy, the new kNN classifier outperforms the other two classifiers. Since the sample sizes of the three classes of the corneal surface data are quite different, it may be more appropriate to compare the performances of the three classifiers using their average G-mean values. Table 4 displays the means and standard deviations (in parentheses) of the 1000 G-mean values of the three classifiers with various dissimilarity measures. It is seen that in terms of average G-means, the adjusted new kNN classifier outperforms the classic and new kNN classifiers.

Table 3. Means and standard deviations (in parentheses) of the 1000 classification accuracy values (in percent) of the classic, new, and adjusted new kNN classifiers for corneal surface data

Method	L^1		L^2		L^∞		FCD		FPC		FM	
kNN	57.80	(6.19)	55.54	(6.27)	48.93	(5.55)	50.46	(5.82)	53.05	(6.63)	66.50	(5.88)
new kNN	**58.57**	(5.66)	**56.15**	(5.77)	**50.41**	(5.30)	**52.02**	(5.78)	**54.96**	(6.25)	**68.33**	(5.68)
adjusted new kNN	55.98	(5.67)	54.62	(5.78)	47.63	(5.74)	48.07	(6.13)	53.25	(6.58)	65.58	(6.24)

Table 4. Means and standard deviations (in parentheses) of the 1000 G-mean values (in percent) of the classic, new, and adjusted new kNN classifiers for the corneal surface data

Method	L^1		L^2		L^∞		FCD		FPC		FM	
kNN	53.27	(7.47)	51.46	(7.47)	44.43	(7.15)	43.13	(8.37)	47.63	(8.47)	61.67	(7.13)
new kNN	53.91	(7.15)	51.70	(7.30)	45.93	(7.08)	43.73	(8.18)	49.23	(8.46)	63.62	(6.84)
adjusted new kNN	**55.28**	(6.29)	**54.01**	(6.22)	**47.39**	(6.17)	**45.71**	(6.85)	**52.30**	(7.37)	**64.05**	(6.68)

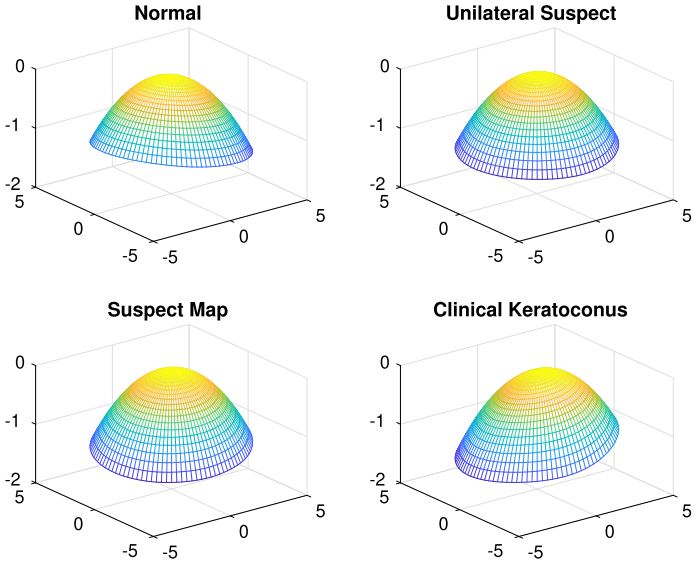

Figure 9. Corneal surfaces from each of the normal, unilateral suspect, suspect map, and clinical keratoconus cornea groups.

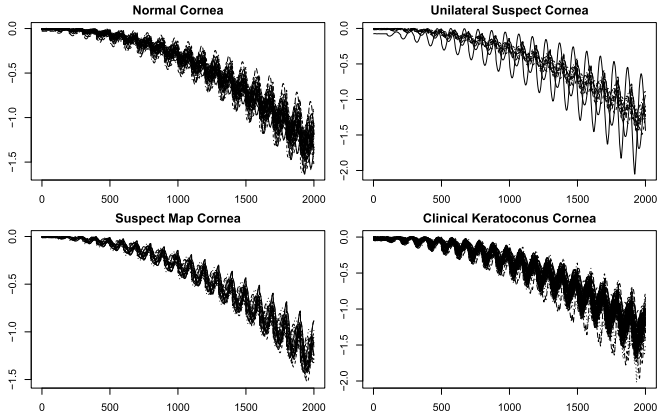

Figure 10. Feature vectors of the four cornea groups.

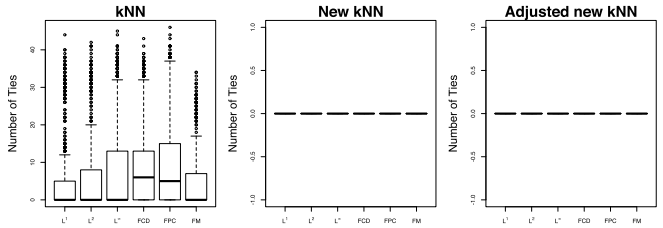

Figure 11. Boxplots of the number of ties happened in the 1000 Monte Carlo runs by the classic, new, and adjusted new kNN classifiers, respectively.

As mentioned in the introduction section, the classic kNN classifier has a tie problem while the new and adjusted new kNN classifiers do not have such a problem. We now demonstrate this via a small scale simulation study. Figure 11 displays the boxplots of the number of ties happened in the 1000 Monte Carlo runs by the classic, new, and adjusted new kNN classifiers, respectively. As expected, there is a serious tie problem for the classic kNN classier while this is not the case for the new and adjusted new kNN classifiers.

6. CONCLUDING REMARKS

The classic kNN classifier works well for supervised classification of functional data but it has a tie-voting problem. To overcome this difficulty, in this paper, we proposed and studied a new k-nearest neighbors classifier. Its key ideas are as follows. For a new unclassified observation, we take k nearest neighbors from each class and compute their average dissimilarity measures from the new observation respectively. This new observation will be assigned to the class with the minimum average dissimilarity measure. Since the dissimilarity measure is continuous, the new kNN classifier does not have a tie-voting problem. When the number of observations in one class is very different from those in other classes, like the classic kNN classier, the new kNN classifier may not

work well. In this case, we propose the adjusted new kNN classifier via selecting the number of nearest neighbors proportional to the sample size of a class. Good performances of the new and adjusted new kNN classifiers are demonstrated by three simulation studies and two real data examples. We believe that the proposed new kNN and adjusted new kNN classifiers may be adopted for other types of data, including the irregular and sparse functional data or high-dimensional data. A further study in this direction is interesting and warranted.

Received 28 February 2020

REFERENCES

[1] Bagui, S. C., Bagui, S., Pal, K. and Pal, N. R. (2003). Breast cancer detection using rank nearest neighbor classification rules. *Pattern recognition* **36** 25–34.

[2] Biau, G., Bunea, F. and Wegkamp, M. H. (2005). Functional classification in Hilbert spaces. *IEEE Transactions on Information Theory* **51** 2163–2172. MR2235289

[3] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory* **13** 21–27.

[4] Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics* **4** 325–327.

[5] Fan, J. and Gijbels, I. (1996). *Local polynomial modelling and its applications*. CRC Press. MR1383587

[6] Ferraty, F. and Vieu, P. (2003). Curves discrimination: a nonparametric functional approach. *Computational Statistics & Data Analysis* **44** 161–173. MR2020144

[7] Ferraty, F. and Vieu, P. (2006). *Nonparametric functional data analysis: theory and practice*. Springer Science & Business Media. MR2229687

[8] Fix, E. and Hodges Jr, J. L. (1951). Discriminatory analysis-nonparametric discrimination: consistency properties. Technical Report, DTIC Document.

[9] Friedman, J., Hastie, T. and Tibshirani, R. (2001). *The elements of statistical learning* **1**. Springer series in statistics Springer, Berlin. MR2722294

[10] Galeano, P., Joseph, E. and Lillo, R. E. (2015). The Mahalanobis distance for functional data with applications to classification. *Technometrics* **57** 281–291. MR3369683

[11] Gates, G. (1972). The reduced nearest neighbor rule (Corresp.). *IEEE transactions on information theory* **18** 431–433.

[12] Gowda, K. and Krishna, G. (1979). The condensed nearest neighbor rule using the concept of mutual nearest neighborhood (Corresp.). *IEEE Transactions on Information Theory* **25** 488–490.

[13] Guo, G., Wang, H., Bell, D., Bi, Y. and Greer, K. (2003). KNN model-based approach in classification. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"* 986–996. Springer.

[14] Kubat, M., Holte, R. C. and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine learning* **30** 195–215.

[15] Ramsay, J. O. and Silverman, B. W. (2005). *Functional Data Analysis*. Spring, New York. MR2168993

[16] Rossi, F. and Villa, N. (2006). Support vector machine for functional data classification. *Neurocomputing* **69** 730–742.

[17] Sguera, C., Galeano, P. and Lillo, R. (2014). Spatial depth-based classification for functional data. *Test* **23** 725–750. MR3274472

[18] Smaga, Ł. and Zhang, J.-T. (2019). Linear hypothesis testing with functional data. *Technometrics* **61** 99–110. MR3933662

[19] Stone, C. J. (1977). Consistent nonparametric regression. *The annals of statistics* 595–620. MR0443204

[20] Sun, Y., Kamel, M. S. and Wang, Y. (2006). Boosting for learning multiple classes with imbalanced class distribution. In *Data Mining, 2006. ICDM'06. Sixth International Conference on* 592–602. IEEE.

[21] Wahba, G. (1990). *Spline models for observational data* **59**. Siam. MR1045442

[22] Yang, Q. and Wu, X. (2006). 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making* **5** 597–604.

[23] Zhang, J.-T. (2013). *Analysis of variance for functional data*. CRC Press. MR3185072

[24] Zhang, J.-T. and Chen, J. (2007). Statistical inferences for functional data. *The Annals of Statistics* **35** 1052–1079. MR2341698

[25] Zhu, T. and Zhang, J.-T. (2020). Cosine similarity-based classifiers for functional data. In *Contemporary Experimental Design, Multivariate Analysis and Data Mining* 277–292. Springer.

Tianming Zhu
Department of Statistics and Applied Probability
National University of Singapore
Singapore
E-mail address: tianming_zhu@nus.edu.sg

Jin-Ting Zhang
Department of Statistics and Applied Probability
National University of Singapore
Singapore
E-mail address: stazjt2020@nus.edu.sg