Knowl Inf Syst (2016) 46:707–730
DOI 10.1007/s10115-015-0846-3

REGULAR PAPER

Classifying imbalanced data in distance-based feature space

Shin Ando[1]

Received: 26 June 2014 / Revised: 17 January 2015 / Accepted: 12 May 2015 /
Published online: 28 May 2015

Abstract Class imbalance is a significant issue in practical classification problems. Important countermeasures, such as re-sampling, instance-weighting, and cost-sensitive learning have been developed, but there are limitations as well as advantages to respective approaches. The synthetic re-sampling methods have wide applicability, but require a vector representation to generate additional instances. The instance-based methods can be applied to distance space data, but are not tractable with regard to a global objective. The cost-sensitive learning can minimize the expected cost given the costs of error, but generally does not extend to nonlinear measures, such as F-measure and area under the curve. In order to address the above shortcomings, this paper proposes a nearest neighbor classification model which employs a class-wise weighting scheme to counteract the class imbalance and a convex optimization technique to learn its weight parameters. As a result, the proposed model maintains the simple instance-based rule for prediction, yet retains a mathematical support for learning to maximize a nonlinear performance measure over the training set. An empirical study is conducted to evaluate the performance of the proposed algorithm on the imbalanced distance space data and make comparison with existing methods.

Keywords Class imbalance · Weighted nearest neighbor classifier · Structural classifier

1 Introduction

Class imbalance is a common issue in practical classification problems. Its disruptive effects over a wide range of algorithms have motivated an increasing amount of research in recent years [1,2]. A large imbalance in the number of instances causes classification algorithms to over-generalize for the class that accounts for the majority of the training set. Typically, an affected classifier achieves good overall accuracy but performs poorly with regard to

✉ Shin Ando
shin.ando@acm.org

1 School of Management, Tokyo University of Science, Tokyo, Japan

the minority class, e.g., in receiver operation characteristics (ROC) and F1-measure. It is especially problematic when the primary interest is on the minority class, e.g., credit card frauds, network attacks, hotspots, or diseases [3]. The effects of class imbalance in various algorithms have been elaborated in the past literature [1,2].

Previous studies on this topic have shown many useful countermeasures against class imbalance, and many are based on one of three major approaches re-sampling, instance-based learning, and cost-sensitive learning. Re-sampling is an intuitive approach to adjust the balance of class distributions, and synthetic over-sampling, in particular, has been shown to improve many classification algorithms [1,4]. Synthetic instances are typically generated by interpolation, i.e., linearly combining, existing minority instances. For interpolation, the input data need to be represented as a vector of attributes without nonlinear dependencies. In some applications, however, the input is given as pair-wise similarities or has a domain-specific structure. For example, the time series data consist of sequentially structured attributes, and the popular distance measure based on warping generates a pair-wise dissimilarities [5].

Another intuitive approach to the class imbalance problem is to emphasize the minority class with larger weights. In recent studies, the nearest neighbor algorithms with instance-weighting have been proposed for classifying imbalanced data [6,7]. The minority instances are given larger weights in the voting of the *k*-nearest neighbors to compensate for their sparsity. Many of the instance-weighting approach can exploit pair-wise dissimilarities and thus can be applied to metric or distance space data. Meanwhile, many instance-based learning employs a *bottom-up* approach, with which the global objective is difficult to keep track. Their parameter selection mostly relies on heuristics, e.g., choosing weights to maximize precision and recall over a local subset, but for nonlinear performance measures [6], optimizing over individual subsets does not assure improvement in the global performance. Furthermore, choosing the size or the range of the local subset adds to the problem of model selection.

In many applications where the class imbalance occurs, the cost of misclassifying the minority class is larger than that of others, which is a motivation for employing the cost-sensitive learning (CSL), such as MetaCost [8] and cost-sensitive boosting [9], on imbalanced data. However, the exact costs of errors are often unknown in practice, in which case general objective functions, such as F1 and area under the ROC curve (AUROC), are used to evaluate the performance. These nonlinear measures are not tractable in the CSL framework.

To summarize, the issues that limit current approaches against class imbalance are (1) addressing a non-vector representation, e.g., distance space data, (2) learning optimal model parameters, and (3) handling of nonlinear performance measures for the minority class. The motivation of this work is to develop a classification model that can address these three issues.

In this paper, an extension of the nearest neighbor classifier with a class-wise weighting scheme is proposed to counter the class imbalance. The mathematical part of this work shows the relation between the proposed model and a margin-based structural classifier [10,11] in a dissimilarity-based feature space. Finally, a training algorithm that can directly optimize a nonlinear performance measure with regard to the weight parameters of the proposed model is presented. The proposed model makes predictions by a simple weighted nearest neighbor rule, but gains effectiveness and efficiency from learning its parameters in a convex optimization problem, as opposed to using local heuristics or validation for a discrete grid search. Furthermore, an extension of the dissimilarity-based representation is proposed to exploit additional information on the class structure of the data, while maintaining the consistency with the optimization technique. An empirical study is presented to evaluate the proposed method on a collection of imbalanced datasets and to compare with existing models.

The rest of this paper is organized as follows. Section 2 discusses the related work on class imbalance, and Sect. 3 illustrates the motivation of this work. Sections 4 and 5 describe the proposed algorithm and the empirical results, respectively. The conclusion of this paper is presented in Sect. 6.

2 Related work

2.1 Re-sampling on imbalanced data

When a population is dominated by a majority class, it causes a strong bias for the statistical learning algorithms to over-generalize for that class. A biased model can reduce both the empirical risk over the training data and the complexity of the hypothesis, thus is difficult to avoid. The problem of class imbalance is mostly discussed in the two-class setting, and the minority and the majority classes are also referred to as the positive and the negative classes, respectively.

Various means to counteract the bias of imbalanced data have been proposed in the past literature [1,2]. In effect, they expand the decision boundaries for the minority class to reduce type-II errors, i.e., the misclassification of the minority class instances. Three major approaches: re-sampling, instance-weighting, and cost-sensitive learning, have been covered extensively in previous work.

Re-sampling adjusts the class distribution typically by over-sampling the minority class instances or under-sampling the majority class instances. Other types of re-sampling include: random, informed, and synthetic sampling [4]. Statistical re-sampling methods such as bootstrapping can also enhance margin-based classifiers in imbalanced settings [12,13]. The synthetic over-sampling is one of the most popular methods in this topic due to its affinity with many classification algorithms. There is a rich literature on integrating synthetic over-sampling into training of support vector machines and ensemble classification models [13–15].

Typically, synthetic instances are generated by linear combinations of the minority class samples to interpolate and make up for the sparsity of the class. In order to linearly combine the instances, their representation needs to be a vector of identical dimensionality and void of nonlinear dependencies between attributes. Subsequently, its applications to domains where input is represented in a distance space is limited.

In [5], a synthetic over-sampling method for metric and distance space data called SVM with ghost points (SVM-GP) has been proposed. In SVM-GP, each synthetic minority instance is embedded to the distance matrix as a pair of a row and a column that satisfies the triangle inequalities for every triplets of instances. The augmented distance matrix is used as a pre-computed distance kernel for training the SVM. The advantage of SVM-GP over the standard SVM was shown empirically in [5].

2.2 Instance-weighting and bottom-up approaches

Instance-based learning algorithms have drawn interest in the early studies on class imbalance [16], and recent studies have also reported the effectiveness of the extended nearest neighbor algorithms on imbalanced data [6,7]. In [7], a k-nearest neighbor algorithm with a weighted voting model called Class Confidence Weighted k-NN (CCWkNN) algorithm has been proposed. In order to counteract the imbalance, CCWkNN gives larger weights to the minority class instances based on the class conditional probability. While the assumption of a generative model is powerful, probabilistic modeling is usually difficult in cases where the instance-based techniques are employed. For example, useful transformations for time series classification, such as dynamic time warping and discrete Fourier transform, are difficult to use with a generative model.

The bottom-up approaches for building classification models are generally guided by the performance at a local scale. The Exemplar-based k-nearest neighbor algorithm (ENN) [6] employs an instance selection technique [17] to expand the decision boundary around *pivotal* positive class instances, which are selected based on the local evaluation of precision and recall. In [18], a bottom-up rule induction called BRACID is introduced for building a rule-based classifier. BRACID generates rules from sets of k-nearest neighbor examples and accumulates those with the best performance measures. In general, learning the model from local subsets is computationally cheaper, but does not accumulate to an optimal performance globally. The size or the extent of the local subsets significantly affects the performance in the bottom-up approach and usually requires validation for tuning.

In [19], a nearest neighbor classifier based on a weighted non-parametric density estimate has been proposed for classifying imbalanced data. Based on the relation between the non-parametric density model and a margin-based classifier, the training algorithm for the weight parameters were derived. However, its formulation only supported the binary classification problem where the number of neighbors k was 1. This formulation is extended in this paper to address general cases where the number of classes is more than 2 and $k > 1$.

2.3 Performance optimizing learning

In the topic of information retrieval, optimization problems for nonlinear performance measures such as F1 and AUC have drawn strong interests in relation to search engines and recommendation systems [20–22]. However, convex optimization techniques are implemented for specific problems and do not easily transfer to others. For example in [22], a structural support vector machine (SVM) for optimizing the AUROC and F1 has been developed, but its optimization technique based on a cutting-plane algorithm does not extend to distance and pre-computed kernels. The structural classifier proposed in this paper addresses the distance space data based on the formulation of the weighted nearest neighbor density estimation.

3 Illustration of motivation

This section first reviews the basic concept of non-parametric density estimation, underlying the nearest neighbor classification. Secondly, the impact of class imbalance on the nearest neighbor algorithm and the motivation of this work are introduced using an illustrative example.

3.1 *k*-Nearest neighbor density model

The k-nearest neighbor density estimation is related to Kernel density estimation (KDE) using a uniform kernel, with automatically chosen parameters [23]. It is one of the nonparametric density estimation methods which is important in practice for building a classifier. The popular k-nearest neighbor algorithm is considered a variation of this model, where k-neighbors are selected from the mixture of classes, with identical behaviors in binary classification problems.

Let $\mathscr{X}$ denote a set of n points, each taking a class value $c \in \{c_1, \ldots, c_m\}$. Let us consider a sphere centering on a point $x \in \mathscr{X}$ and encloses k points. In the following, it is referred to as the k-radius sphere and its volume is denoted by $V_k(x)$. The k-nearest neighbor density estimate of $p(x)$ is

$$p(x) = \frac{k-1}{nV_k(x)} \tag{1}$$

It was shown in [23] that asymptotically, Eq. (1) gives an unbiased estimate of $p(x)$ for large k.

The conditional density of a point given a class is estimated in the same manner. Let n_i denote the total number of points of c_i, and $V_{i,k}(x)$ the volume of the sphere containing k points of c_i, respectively. The conditional density is

$$p(x|c_i) = \frac{k-1}{n_i V_{i,k}(x)} \tag{2}$$

Based on the probability density estimates, we can design a classifier that selects the class with the maximum posterior probability. The class probability given x can be compared by substituting the class prior $p(c_i) = \frac{n_i}{n}$ and (2) to the Bayes rule

$$p(c_i|x) \propto p(x|c_i)p(c_i) = \frac{1}{n}\frac{k-1}{V_{i,k}(x)} \tag{3}$$

The maximum posterior class $\hat{c}$ is written as

$$\hat{c} = \underset{c \in \{c_i\}_{i=1}^m}{\arg\max}\, p(c|x) \tag{4}$$

and from (3), $\hat{c}$ is equivalent to $c_{\hat{i}}$, where

$$\hat{i} = \underset{i}{\arg\min}\, V_{i,k}(x)2 - 10 \tag{5}$$

Note that $k-1$ is independent of class and thus removed from the maximization.

3.2 The nearest neighbor classification with class imbalance

Figure 1 illustrates an artificial dataset which has a pervasive majority class. The blue and red dots indicate the instances of the majority and the minority classes, respectively. The rectangular region, indicated by the red dashed line in Fig. 1, contains the minority class examples and is magnified in Fig. 2. Figure 2 shows the decision function of the nearest neighbor algorithm over the magnified region, which reduces to a Voronoi diagram. It shows that the presence of the majority class can induce type-II errors for a non-generalizing model such as the nearest neighbor algorithm. Note that type-II errors occur even with more precise means of density estimation. For example, KDE using a smooth kernel function produces a slightly smoother but similar decision boundaries as long as equal emphases are placed on the majority and the minority instances.

Reviewing the effect of imbalance with regard to the formulation in Sect. 3.1, it came to our attention that the conditional density estimate in (2) in particular is substantially susceptible. That is, the conditional density of the majority class $p(x|c_i)$ and $p(x)$ are estimated over regions of similar sizes, while the minority class uses a much larger k-radius. It can thus lead to an under-estimation of the density for the latter. Learning the adjustments for the above discrepancy in density estimation is the main approach of this paper.

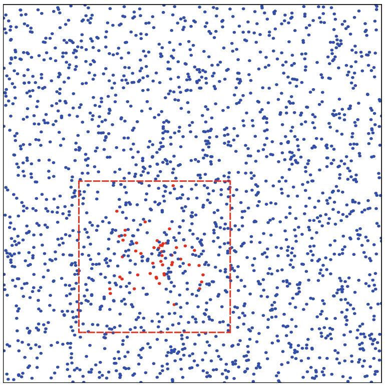


Fig. 1 2-Dimensional imbalanced data

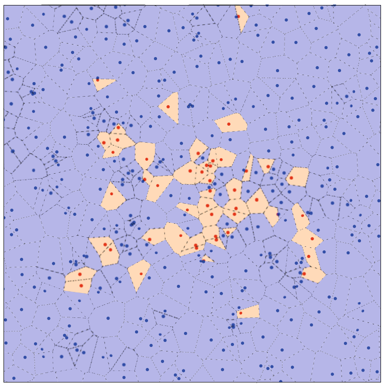


Fig. 2 The nearest neighbor rule on imbalanced data

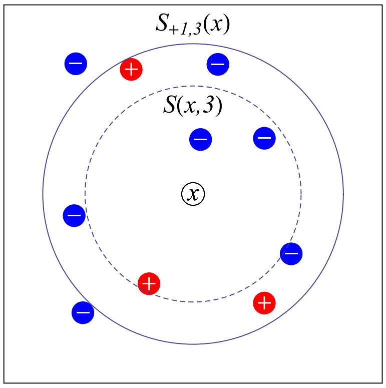
4 Class-wise nearest neighbor classification

This section describes the proposed method for minimizing the non-linear performance loss over the imbalanced data. First, a non-parametric density model with weighting, which emphasizes the importance of the minority instances, is introduced. Secondly, the problem of learning the weight parameters from the training data is formulated. Thirdly, the procedure for minimizing the non-linear performance loss is proposed. Finally, the implementation detail of the learning algorithm is presented.

4.1 Weighted density model

Let $\mathscr{X}$ denote a set of n points. Each point takes a class value from $\{c_1, \ldots, c_m\}$. Let us refer to the sphere, which centers on x and has the smallest radius containing at least k instances of class c_i, as the class-wise k-radius sphere of c_i. Such a sphere and its volume is denoted by $S_{i,k}(x)$ and $V_{i,k}(x)$, respectively. Figure 3 shows an illustration of the k-radius spheres for the imbalanced data. The $\oplus$'s and $\ominus$'s indicate the minority and the majority class instances, respectively. The solid and dashed circles indicate the class-wise k-radius and the original k-radius. The class-wise k-radius of the minority class is usually larger than that of the majority class as shown in this example.

Fig. 3 The class-wise k-radius spheres



The key intuition to compensate for the sparseness of the minority class is to employ an adjusted k-radius volume in place of $V_{i,k}$. Let us define the adjusted volume $\tilde{V}_{i,k}$ using a positive coefficient β_i, as follows:

$$\tilde{V}_{i,k}(x) = \frac{1}{\beta_i} V_{i,k}(x) \tag{6}$$

The value of β_i corresponds with the sparseness of the class, i.e., for the majority class, the weight is close to 1 as the k-radius is similar to its class-wise k-radius. Meanwhile, since the class-wise k-radius of the minority class is much larger than the k-radius, the β_i is set to a larger value.

The posterior class membership based on the adjusted volume is given as follows:

$$\hat{c} = \arg\max p(c_i|x) = \arg\max \frac{\beta_i}{V_{i,k}(x)} \tag{7}$$

In the density-based classification algorithms [24], the inverse of $V_{i,k}(x)$ is considered the *density* of a point x, and $V_{i,k}(x)$ is commonly approximated by the distance to the kth-nearest neighbor $\mathscr{D}_k(x)$. Based on this approximation, (7) can be seen as a weighted k-nearest neighbor classifier. We refer to this model as the weighted class-wise nearest neighbor (WCNN) classifier. The procedure of classifying a new instance x in the WCNN classifier is as follows:

- For each class c_i, compute the distance to its kth-nearest instance of class c_i in the training data $\mathscr{D}_{i,k}(x)$ as an approximation of $V_{i,k}(x)$
- Compute the maximum posterior class

$$\hat{c} = \arg\max_{c_i} \frac{\beta_i}{\mathscr{D}_{i,k}(x)} = \arg\min_{c_i} \frac{\mathscr{D}_{i,k}(x)}{\beta_i} \tag{8}$$

Compared to the nearest neighbor density model in Sect. 3.1, Eq. (8) expands the decision boundaries of the minority class based on the set of weights $\{\beta_i\}_{i=1}^m$.

4.2 Discriminative formulation of the WCNN classifier

Let us first define the training of the WCNN classifier in the two-class case. Let x denote an input, not necessarily of a vector form, whose approximation of the k-radius volume $\tilde{V}_{i,k}(x)$ is available. A vector representation of an instance x based on the class-wise k-radius volumes is defined as follows

$$\mathbf{v} = (v_1, \ldots, v_m) = \left(\tilde{V}_{1,k}(x), \ldots, \tilde{V}_{m,k}(x)\right) \tag{9}$$

where m denote the number of classes.

The majority and the minority classes, denoted by c_i and c_j, are assigned the class values -1 and $+1$, respectively. If the approximation is positive and monotonic, it can be shown that

Theorem 1 *For a binary classification problem, the maximum posterior class estimate of the new instance x, in (4), can be given by a discriminative function with a linear decision boundary in a following form.*

$$f(\mathbf{v}; \mathbf{w}) = sgn(\mathbf{wv}) \tag{10}$$

For the proof of Theorem 1, the following lemma is derived with regard to the posterior class probabilities introduced in Sect. 3.1,

Lemma 1 *The decision function based on the logarithmic ratio of the posterior class probabilities $p(c_i|x)$, $p(c_j|x)$ and one based on that of the class-wise k-radius volumes $V_{i,k}(x)$, $V_{j,k}(x)$ are equivalent, i.e.,*

$$\operatorname{sgn}\left(\log \frac{p(c_j|x)}{p(c_i|x)}\right) = \operatorname{sgn}\left(\log \frac{V_{i,k}(x)}{V_{j,k}(x)}\right) \tag{11}$$

The proof is straightforward and is omitted here.

Combining the result of Theorem 1 with the approximation of the k-radius volume using the distance to the kth-nearest neighbor

$$\tilde{V}_{i,k}(x) = \frac{\mathscr{D}_{i,k}(x)}{\beta_i} \tag{12}$$

with the weight β_i results in the following model

$$\arg\max_c p(c|x) = \operatorname{sgn}\left(\sum\nolimits_{c_q \in \{c_i, c_j\}} - c_q v_q\right) \tag{13}$$

$$= \arg\max_c \left(-c \sum_{c_q \in \{c_i, c_j\}} c_q \frac{\mathscr{D}_{q,k}(x)}{\beta_q}\right) \tag{14}$$

Equation (14) is a linear classifier based on the class-wise nearest neighbor distances $\mathscr{D}_{q,k}$. It is parametrized by β_q and returns an identical output as the nearest neighbor classifier when $\beta_q = 1$ for all q. Using a greedy algorithm with such an initial solution, the learned model can perform at least as well as the nearest neighbor classifier.

Let us now extend Theorem 1 to the general case where multiple majority and/or minority classes exist. It is assumed that the classes can be divided into sets of majority and minority classes. The distinction of the two sets is usually not a critical issue as the cardinality of each class is apparent from the training set. Additionally, since the target class is a minority in most cases, non-target, intermediate-sized classes can be included in the majority set without harm. Finally, the effect of class imbalance among minority classes is limited in practice as minority classes are commonly not pervasive in the data.

Under the above assumption, the class imbalance between multiple majority and minority classes can be accounted for by comparing the class memberships separately among the majority and the minority classes, and then comparing the highest conditional density estimates from each group. The key intuition is to implement such a decision function with a structural classifier [10,11].

In [11], the structural classifier is defined as a classification model with multiple input and output variables. The general form of the decision function of a structural classifier is comprised of an inner product of a *linear* discriminant function and an *optimizer* over the class variable Y, which replaces the sign function used in the binary case. In the following, a decision function f of above form, which returns a maximum posterior class based on the nearest neighbor density estimate, is derived.

Let $\mathscr{M} = \{i\}_{i=1}^{m}$ and $\mathscr{N} = \{-j\}_{j=1}^{n}$ denote the class values of majority and minority, respectively, and $\mathbf{d}(x)$ the distance-based representation of x

$$\mathbf{d}(x) = (\mathscr{D}_{1,k}(x), \ldots, \mathscr{D}_{m+n,k}(x))$$

Let $\hat{i}$, $\hat{j}$ denote the classes of highest posterior estimates from the majority and the minority, respectively. They relate to the log ratio of posteriors between a minority and a majority classes as

$$(\hat{i}, \hat{j}) = \arg\min_{j\in\mathscr{N}} \max_{i\in\mathscr{M}} \log \frac{p(i|x)}{p(j|x)} \tag{15}$$

The maximum posterior estimate $\hat{c}$ is either $\hat{i}$ or $\hat{j}$, thus rewritten as

$$\hat{c} = \arg\max_{c\in\mathscr{M}\cup\mathscr{N}} \log p(c|x) \tag{16}$$

$$= \begin{cases} \hat{i} & \text{if} \quad \operatorname{sgn}\left(\log \frac{p(\hat{i}|x)}{p(\hat{j}|x)}\right) = 1 \\ \hat{j} & \text{otherwise} \end{cases} \tag{17}$$

Since Lemma 1 stands for all combinations of $(i, j) \in \mathscr{M} \times \mathscr{N}$, the decision function based on the log posterior ratio is rewritten using the vector of adjusted volumes as

$$\operatorname{sgn}\left(\log \frac{p(i|x)}{p(j|x)}\right) = \operatorname{sgn}\left(\tilde{V}_{i,k}(x) - \tilde{V}_{j,k}(x)\right) \tag{18}$$

$$= \operatorname{sgn}\left((\mathbf{u}_i - \mathbf{u}_j)^\top \mathbf{v}\right) \tag{19}$$

where $\mathbf{u}_i$ is a unit vector whose ith element is 1.

Let G denote the log ratio of the posteriors $G(i, j) = \log \frac{p(i|x)}{p(j|x)}$ and ϕ an optimizer which selects one input of G.

$$\phi(i, j) = \begin{cases} i & \text{if} \quad \operatorname{sgn}(G(i, j)) = 1 \\ j & \text{otherwise} \end{cases} \tag{20}$$

From (19) and (20), (17) can be rewritten as

$$\arg\max_{c\in\mathscr{M}\cup\mathscr{N}} \log p(c|x) = \phi\left(\left(\mathbf{u}_{\hat{i}} - \mathbf{u}_{\hat{j}}\right)^\top \mathbf{v}\right) \tag{21}$$

Furthermore, the discriminant function F is written in a inner product form as

$$F(x; \mathbf{w}) = (\mathbf{u}_{\hat{i}} - \mathbf{u}_{\hat{j}})^\top \mathbf{v} \tag{22}$$

$$= \frac{\mathscr{D}_{\hat{i},k}(x)}{\beta_i} - \frac{\mathscr{D}_{\hat{j},k}(x)}{\beta_j}$$

$$= \langle \mathbf{w}, \Psi(\mathbf{d}(x)) \rangle \tag{23}$$

where $\mathbf{w} = (\beta_1^{-1}, \ldots, \beta_{m+n}^{-1})$ and Ψ is the feature function

$$\Psi(x) = \left(\mathbf{u}_{\hat{i}} - \mathbf{u}_{\hat{j}}\right)^{\top} \mathbf{d}(x)$$

Note that $\hat{i}$ and $\hat{j}$ are both functions of x.

From (21) and (23), it follows that

Theorem 2 *the maximum posterior estimate is given by a decision function f defined by a linear discriminant function and an optimizer ϕ, of the following form.*

$$f(\mathbf{d}; \mathbf{w}) = \phi\left(\langle \mathbf{w}, \Psi(\mathbf{d})\rangle\right) \tag{24}$$

Compared to the binary case in (14), the sign function is replaced by the optimizer ϕ and the nearest neighbor distance is replaced by the feature function Ψ in (24). Let us refer to the structural classifier of the above form as the Structural Nearest Neighbor (SNN) classifier.

The prediction of the SNN classifier is computed in a modified procedure of the nearest neighbor density estimation and replicates that of the WCNN classifier in the multiclass problem.

Given a new instance x,

- Select a majority class c_i, for which the distance between its kth-nearest instance and x is smallest, from the majority classes $\mathscr{M}$.
- Select a minority class c_j, for which the distance between its kth-nearest instance and x is smallest, from the minority classes $\mathscr{N}$.
- Based on the approximate density $\mathscr{D}_{i,k}(x)$ and $\mathscr{D}_{i,k}(x)$, select the maximum posterior estimate between c_i and c_j, such that $\hat{c} = \arg\min_{c\in\{i,j\}} \frac{\mathscr{D}_{c,k}}{\beta_c}$

The weight parameters of the WCNN classifier emphasizes a minority class relative to each majority class, i.e., the emphasis on a minority class i against a majority class j is quantified in the pair of weights (β_i, β_j). The output $\hat{c}$ is identical to that of (17) when $\beta_q = 1$ for all q. Learning $\mathbf{w}$ is a quadratic programming problem as described in the next section.

4.3 Learning SNN classifier with nonlinear performance loss

In [22], the structural SVM was proposed for problems where the *loss* is nonlinear, i.e., not decomposable to the predictions on individual instances. For reference, the typical formulation of learning the structural SVM is shown in Appendix 1. In this section, the training of the SNN classifier to optimize a nonlinear performance measure is formulated as a quadratic programming problem. Its approximate solution can be obtained by extending the algorithm for the structural SVM learning. For the sake of conciseness, this section focuses on the binary classification problem. The intuition for addressing more than one majority and minority classes is similar, but due to its length, the detail is deferred to Appendix 2.

Let $(x_1, y_1), \ldots, (x_\eta, y_\eta)$ denote the training data where y takes a class value from $\{-1, 1\}$. x itself does not need to be a feature vector, but its representation based on the distances to the nearest neighbor of respective classes, $\mathbf{d}(x)$, must be available. The input feature is represented as $\mathscr{X} = (\mathbf{d}(x_1), \ldots, \mathbf{d}(x_\eta))$ and the output as $\mathbf{y} = (y_1, \ldots, y_\eta)$ respectively.

In [11], the intuition for optimizing a nonlinear performance loss is to formulate the decision function such that the input and the output of all instances can be considered in each constraint of the quadratic programming problem. To this end, $\mathscr{X}$ is considered an instance of the multivariate input variable X and $\mathbf{y}$ an instance of the multivariate output variable Y. Further, the loss function Δ is associated with the nonlinear measures, e.g., AUC or F1, as the loss in the performance given an arbitrary prediction $\mathbf{y}'$ compared to the one given the *correct* output $\mathbf{y}$. Δ is thus written as a function of $\mathbf{y}$ and $\mathbf{y}'$, $\Delta(\mathbf{y}', \mathbf{y})$.

Let us define the feature function Ψ of the variables X and Y as

$$\Psi(X, Y) = \sum_{i=1}^{\eta} y_i \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \mathbf{d}(x_i) \tag{25}$$

The discriminant function can be written in a linear form as

$$F(X, Y; \mathbf{w}) = \langle \mathbf{w}, \Psi(X, Y) \rangle$$

where $\mathbf{w}$ is the weight vector $\mathbf{w} = (\beta_{+1}, \beta_{-1})$. The decision function f, which chooses the output $\mathbf{y}$ with the maximum margin, is written as follows:

$$\begin{aligned} f(X; \mathbf{w}) &= \arg\max_{Y} F(X, Y; \mathbf{w}) \\ &= \arg\max_{Y} \sum_{i=1}^{\eta} y_i (\mathbf{w}^\top \mathbf{d}(x_i)) \end{aligned} \tag{26}$$

The decision function for individual input, f', is obtained by decomposing the summation in (26) for each $\mathbf{d}$

$$f'(X; \mathbf{w}) = \arg\max_{y \in \{1, -1\}} y(\mathbf{w}^\top \mathbf{d}) \tag{27}$$

with the same form as (14).

In [10], the minimization of the loss Δ with regard to a *linear* decision function in the form of (26) is formulated as a quadratic programming problem.

Problem 1

$$\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2} \|\mathbf{w}\|^2 + C\xi \tag{28}$$

subject to $\forall \mathbf{z} \in \{-1, 1\}^\eta \backslash \{\mathbf{y}\}$,

$$\langle \mathbf{w}, \Psi(\mathbf{z}, X) \rangle - \langle \mathbf{w}, \Psi(Y, X) \rangle \geq \Delta(\mathbf{z}, Y) - \xi \tag{29}$$

where $\mathbf{z} = (z_1, \ldots, z_\eta)$ denotes the vector of predictions on the output variable and ξ is the slack variable that sets the upper-bound on the loss.

Equation (28) is the minimization of the loss Δ subject to the constraints on the margins in the feature space and is solved by a cutting-plane method [10]. The detail of the algorithm is described in Sect. 4.5.

Equation (1) reformulates the nearest neighbor classification of distance space data as a quadratic programming problem. WCNN gains a significant advantage from optimizing the weight parameters, as opposed to selecting a combination of parameter values by validation. Furthermore, the capacity to handle multiclass problems, as shown in the next section, allows us to exploit additional information on the class structure with an extended distance feature representation.

4.4 Supplementary component features

From its definition in Sect. 4.2, the SNN classifier addresses an m-class classification problem in an m-dimensional distance feature space. When respective classes are generated from single components, searching for an m-dimensional mapping that separates the class components well is a viable approach. In practice, however, one class may be comprised of more than one components. Particularly, in problems with class imbalance, the majority class is often a mixture of unannotated components. It can cause a high perplexity in the distance feature space and prevent effective training of the WCNN classifier.

An intuitive countermeasure for such a problem is to recognize distinct subcomponents as sub-classes and include them in the distance features. To this end, we introduce a clustering procedure that extracts the subcomponent structure from the training data. We design a supervised clustering, which exploits the label on the training instances to identify significant subcomponents comprised of instances from only one class.

Given a training data $\mathscr{X}$ and labels $\mathbf{y}$ taking a class value from $\mathscr{C} = \{c_i\}_{i=1}^{m}$,

1. Generate a hierarchical tree H of $\mathscr{X}$ by single-linkage clustering.
2. Extract all subtrees of H with λ or more leaves that are of the same class. Let us refer to these subtrees as *uniform* subtrees.
3. Eliminate all uniform subtrees that are a subtree of another uniform subtree.
4. Generate a distance representation of x,

$$\mathbf{d}'(x) = (\mathscr{D}_{1,k}(x), \ldots, \mathscr{D}_{m+|\mathscr{U}|,k}(x)),$$

where $\mathscr{D}_{m+j,k}$ denotes distance from x to its kth-nearest leaf in the jth LUS.

The aim in step 3 is to eliminate insignificant components, which is also essential for efficiency. Let us refer to the subtrees remaining after the elimination in Step 3 as the largest uniform subtrees (LUS) and denote the set of LUS's by $\mathscr{U}$. Each LUS represents a *uniform* subcomponent within a class, and λ is a parameter specifying the minimum number of leaves in LUS. Its value should be chosen such that the probability of the partition forming by chance is negligible. For practical problems, λ larger than at least 30 is suggested. Note that the purpose of clustering is to decompose each class into its subcomponents and not to estimate the mixture model or the number of components accurately.

Based on the input labels and partitioning, a high-dimensional feature space for SNN is defined such that the outputs of the SNN and the WCNN classifiers remain consistent. A new set of labels $\mathbf{y}'$ is generated by replacing the class value of each leaf of the jth LUS with c_{m+j}. Each new label takes a value from $\mathscr{C}' = \{c_1, \ldots, c_{m+|\mathscr{U}|}\}$. The new training data $\mathscr{X}' = \{\mathbf{d}'(x)\}$ and $\mathbf{y}'$ replace $\mathscr{X}$ and $\mathbf{y}$ as the input for the SNN classifier. Let u_j denote the jth LUS and $\mathbf{y}'(u_j)$, the vector of class values of u_j in $\mathbf{y}'$. The SNN classifier can replicate the output of the WCNN classifier using $(\mathscr{X}, \mathbf{y}) \cup (u_1, \mathbf{y}'(u_1)), \ldots, (u_{|\mathscr{U}|}, \mathbf{y}'(u_{|\mathscr{U}|}))$ as the training examples. The supplementary labels allow the SNN classifier to assign different relative importance to the subcomponents in the majority class. The predictions on the new labels can be associated with a class in the original labels without confusion.

The clustering procedure described above is related to data cleaning [25]. That is, the label is used *passively* for identifying agreements in the clustering structure and the class structure, which is referred to as the *class-clusters* [1]. There is a distinction from constrained clustering with cannot-link constraints [26,27], which actively exploit labels for discovering the cluster structure. Limiting the use of supervising information in the exploratory learning phase is justifiable, since the proposed method makes more sophisticated use of the labels in the subsequent learning and feature selection process.

Algorithm 1 WCNN Learning Algorithm with Supplementary Component Features

```
1: INPUT: Pair-wise dissimilarity matrix of training data $D = \left[d(x_i, x_j)\right]_{N\times N}$, labels $\mathscr{Y} = \{y_i\}_{i=1}^N$, $k$-radius, precision $\epsilon$
2: OUTPUT: LUS $\mathscr{U}$, weight vector $\mathbf{w} = (\beta_1^{-1}, \ldots, \beta_\upsilon^{-1})$
3: **function** ExtractLUS($\cdot$): extract largest uniform subtrees of $\lambda$ or more leaves from single-linkage clustering tree
4: METHOD:
5: Initialize $m \leftarrow$(# of class values)
6: $\mathscr{U} \leftarrow$ExtractLUS($\mathscr{D}$), $\upsilon = \#\mathscr{U}$
7: **for** $i = 1, \ldots, N$ **do**
8:     Generate $\mathbf{d}_i = (D_{1,k}(x_i), \ldots, D_{m,k}(x_i), \ldots, D_{m+\upsilon,k}(x_i))$ and compute the feature function $\Psi$ (25)
9: **end for**
10: $\mathscr{Z} = \emptyset$
11: **repeat**
12:     Find $\mathbf{z} \in \{1, -1\}^N$ which incurs the maximum violation when substituted to (29)
13:     **if** the violation exceeds the tolerable precision $\epsilon$ **then**
14:         Update $\mathscr{Z} \leftarrow \mathscr{Z} \cup \{\mathbf{z}\}$
15:         Compute $\mathbf{w}$ by minimizing (28)
16:     **end if**
17: **until** The violation is within the precision $\epsilon$
18: **return** $\mathscr{U}$, $\mathbf{w}$
```

4.5 Algorithm and discussions

The pseudocode of the WCNN learning algorithm is presented in Algorithm 1. Lines 7–9 describe the mapping of pair-wise dissimilarities to the supplementary component feature space. Lines 10–17 describe the iterative cutting-plane method for solving Problem 1.

To make predictions on the test instances with the supplementary labels, the class labels are replaced by them in the testing procedure. Given a predicted subcomponent, one can match the supplementary label to the class of the original label.

The complexity of the WCNN classifier is discussed separately for the training phase and the testing phase. In training, the complexity of solving for optimal **w** is a polynomial order [28]. With regard to the testing phase, the complexity of the WCNN algorithm is in the same order as the k-nearest neighbor algorithm. One may achieve $O(\log n)$ given an ideally balanced binary search tree.

5 Empirical results

Our empirical study focuses on the distance space data with class imbalance, where previous studies on the type of data have been limited due to the difficulty of the re-sampling methods. For mapping the time series data to the distance feature space, a domain-specific distance measure called dynamic time warping (DTW) is used. In the following experiment, all input data are given in the form of pair-wise DTW distances.

5.1 Data description

Table 1 shows the summary of the five time series benchmarks used in this experiment. Control Chart [29] and CBF functions [30] are synthetic datasets used as benchmarks in many studies. Lightning is an oft-used real-world dataset, which comprises lightning transients recorded by an orbital satellite.

Table 1 Summary of time series datasets

	Control chart	CBF	Person activity	Characters	Lightning
Series length	60	129	[21:147]	[124:171]	3181
Minority class	Increase/decrease	Bell	Falling	g/q	2/3/4
	Up-shift/down-shift	Funnel	Lying-down	y/z	
Ratio of minority	0.167	0.100	0.148/0.0905	0.250	0.140/0.116/0.157
n-Fold	2	5	10	5	10

Two other real-world time series data from UCI machine learning repository [31] were also included. The *Character Trajectories* dataset contains pen-tip trajectories of single *pen-down* characters captured at 200 Hz with a tablet PC. The trajectories of each character have been pre-formatted to the same length. In order to focus on substantially difficult tasks, four mutually similar characters were selected: g, q, y, and z, to set up the classification problems.

The *Localization Data for Person Activity* dataset consists of 3D coordinates recorded by sensor tags [32]. Four tags were worn on both ankles, the belt, and the chest of a person, respectively. Their coordinates were transmitted at 10 Hz, while the person performed multiple activities. Each coordinate is annotated with a time-stamp and a label indicating one of 11 activities performed during a session. Five people participated in the test, and five sessions were recorded for each person. This experiment focuses on the sequences of activities where *walking* is followed by *falling* and *lying-down*. They are important for the discriminative task in monitoring applications, as they indicate potential risk and a change of state, respectively.

5.2 Baseline methods and evaluation

Baselines were comprised of methods taking pair-wise dissimilarity input and those taking vector input. For the first group, the pair-wise DTW distances were computed as their input. For the second group, the time series in vector forms was given directly as their input.

The first group includes the ENN [6] algorithm, SVM-kNN [33], and SVM-GP [5]. The two SVM methods use the DTW distance matrix as pre-computed kernels. SVM-GP implements a synthetic over-sampling by local embedding in the pre-computed kernel matrix. SVM-kNN is a hybridization of the support vector machine and the k-nearest neighbor algorithm, which has a similar effect as under-sampling in an imbalanced setting. The nearest neighbors and the approximations of the k-radius volumes were computed using DTW.

The second group includes synthetic over-sampling methods: SMOTE [34] and SMOTE-Boost [35] and a cost-sensitive learning framework *MetaCost* [8]. Some of the nearest neighbor-based algorithms, e.g., CCWkNN [6], which builds probabilistic generative models and the large margin nearest neighbor algorithm [36] which uses an affine transform, also require vector representations and belong to this group.

The synthetic over-sampling methods were evaluated with standard classifiers, and the results with the best average performances, which were the RBF kernel SVM and Random Forest [37], respectively, are reported. For the MetaCost framework, the C4.5 decision tree was used as the weak learner following the setting in [6], and the relative cost of misclassifying the minority class instances was set to the inverse of the class ratio, shown in Table 1.

The k-nearest neighbor algorithms were run using $k = 1, 3, 5$, and SVMkNN were run using $k = 5, 10, 20$. In this paper, the best results of the respective baseline methods are reported. The implementation of LibSVM [38] was used for training SVM with pre-computed distance kernel. The SNN classifier was trained to minimize AUC using an implementation based on SVM-light.[1] $k = 3$ was chosen for the k-radius from a preparatory experiment using an independent dataset.

In previous studies, the area under the ROC Curve has been widely used as the evaluation measure for imbalanced classification problems. However, [39] has reported that AUROC yields an optimistic view of the expected loss in some cases and suggested the use of the area under the precision-recall curve (AUPRC). AUPRC is primarily used for comparisons in this study. For the Control Chart data, we follow previous work with the default split of train/test sets and use two-fold cross-validation. Depending on the total number of minority class examples, other datasets were split randomly into 5 or 10 folds for cross-validation, so that each class was divided evenly between the folds and there were at least 20 minority class examples in each fold. Maintaining the number of minority class examples in the test fold is important when evaluating F-measure and AUPRC, e.g., the recall reduces to a discrete value when there is only one positive example.

For each baseline, the signed-rank test is performed with an alternative hypothesis that the median AUC of WCNN is greater. Note that the purpose of each test is to compare the proposed algorithm with a baseline method individually and not to rank all methods in order. With regard to the statistical tests for comparing different classifiers, in [40], the signed-rank test and the Mann-Whitney test were compared in a controlled experiment, and the latter was reported to be more powerful. However, the use of a more conservative test is justified for this study as significant differences between the methods were detected.

5.3 Results

Table 2 summarizes the AUPRC values. The first column shows the minority classes of respective problems. The second column shows the rank of WCNN regarding AUC. The asterisk after the rank indicates that there is a tie with one or more columns. Rest of the columns show the AUC values of the baseline methods. The largest value on each row is indicated in bold.

In Table 2, the SVM-GP shows the highest averages among the baseline methods over the artificial datasets. No baseline methods exhibited a clear advantage over others in the real-world datasets, but the first group, using pair-wise distance input, shows a slight overall advantage to the second group. This advantage may be gained from the use of DTW, as reported in previous studies on time series classification. Meanwhile, the WCNN algorithm achieved the best AUPRC in 15 of 17 datasets with the worst rank of 3. The p values of the signed-rank tests are shown on the bottom row. For example, the p values of the hypothesis that the median AUPRC of WCNN is less than or equal to that of SVMkNN is 0.362 %. In all tests, the null hypotheses were rejected with 0.5 % confidence level.

5.4 Scale analysis and graphical examination

This section presents analyses on additional properties of the proposed method. First, a scale analysis, which evaluates the performances of the proposed method under larger class imbalance, is presented. Secondly, the distance feature spaces and the prediction model are graphically examined.

The scale analysis is conducted using an artificial dataset, CBF functions, with which the ratio of the minority class instances is controllable. The ratio of the minority class instances were changed from 0.0066, 0.016, 0.032, to 0.062. Figure 4 illustrates AUCs at respective

[1] http://svmlight.joachims.org.

Table 2 Summary of area under the precision-recall curve

Minority class	Rank	WCNN	SVM*k*NN	ENN	SVMGP	SMOTE+RF	SMOTEBoost	MetaCost
Increase	1	**0.999**	0.902	0.952	0.992	0.889	0.965	0.746
Decrease	1	**0.997**	0.917	0.961	0.984	0.903	0.953	0.678
Up-shift	1	**0.997**	0.906	0.927	0.972	0.841	0.956	0.548
Down-shift	1	**0.999**	0.872	0.928	0.952	0.734	0.951	0.601
Bell	1*	**1.00**	0.955	0.989	**1.00**	0.983	0.844	0.942
Funnel	1	**1.00**	0.910	0.990	0.947	0.823	0.553	0.567
Falling	1	**0.592**	0.470	0.371	0.470	0.506	0.574	0.461
Lying-down	1	**0.541**	0.488	0.480	0.479	0.343	0.474	0.528
HW-g	1	**0.984**	0.894	0.944	0.950	0.856	0.879	0.765
HW-q	1	**0.995**	0.984	0.990	0.990	0.952	0.978	0.940
HW-y	3	0.918	0.892	**0.943**	0.939	0.861	0.847	0.794
HW-z	1	**0.985**	0.963	0.981	0.981	0.922	0.942	0.872
L2	1	**0.805**	0.590	0.394	0.373	0.639	0.570	0.574
L3	1	**1.00**	0.787	0.975	0.850	0.838	0.557	0.746
L4	2	0.904	0.875	0.900	**0.912**	0.704	0.579	0.521
Mean rank	–	1.3	4.3	3.5	2.7	5.1	4.9	6.2
p-value ($\times 100$)	–	–	0.0362	0.143	0.419	0.0446	0.0666	0.0363

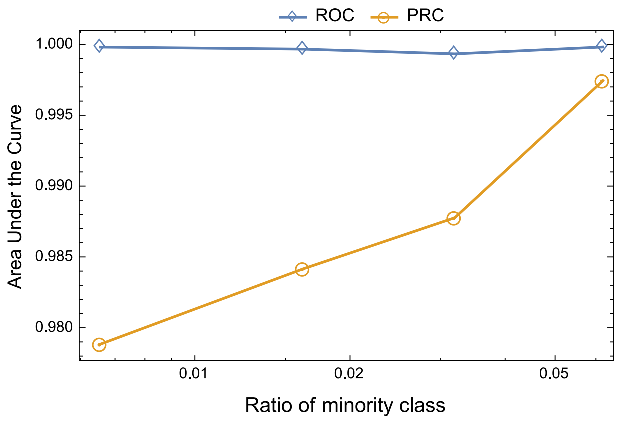


Fig. 4 AUC values versus minority class ratio

ratios. The x-axis indicate the ratio of the minority class instances in log scale, and the y-axes correspond to the AUC values. The AUROC and the AUPRC values are indicated by $\diamond$ and $\odot$, respectively. It is shown that AUPRC decreases with the minority class ratio, but is still very potent at the minority class ratio of 1:150. The rate of decrease is linear to the log ratio, suggesting that the proposed method can be robust for very small minority class ratios. Figure 4 also indicates that the value of AUROC may provide an optimistic view of the performances in imbalanced data.

For graphical analyses, we select two datasets, on which the proposed method achieved the highest and the lowest AUCs in Sect. 5.3: CBF function and Person Activity. The distance feature space for the two datasets is illustrated in Figs. 5 and 6, respectively.

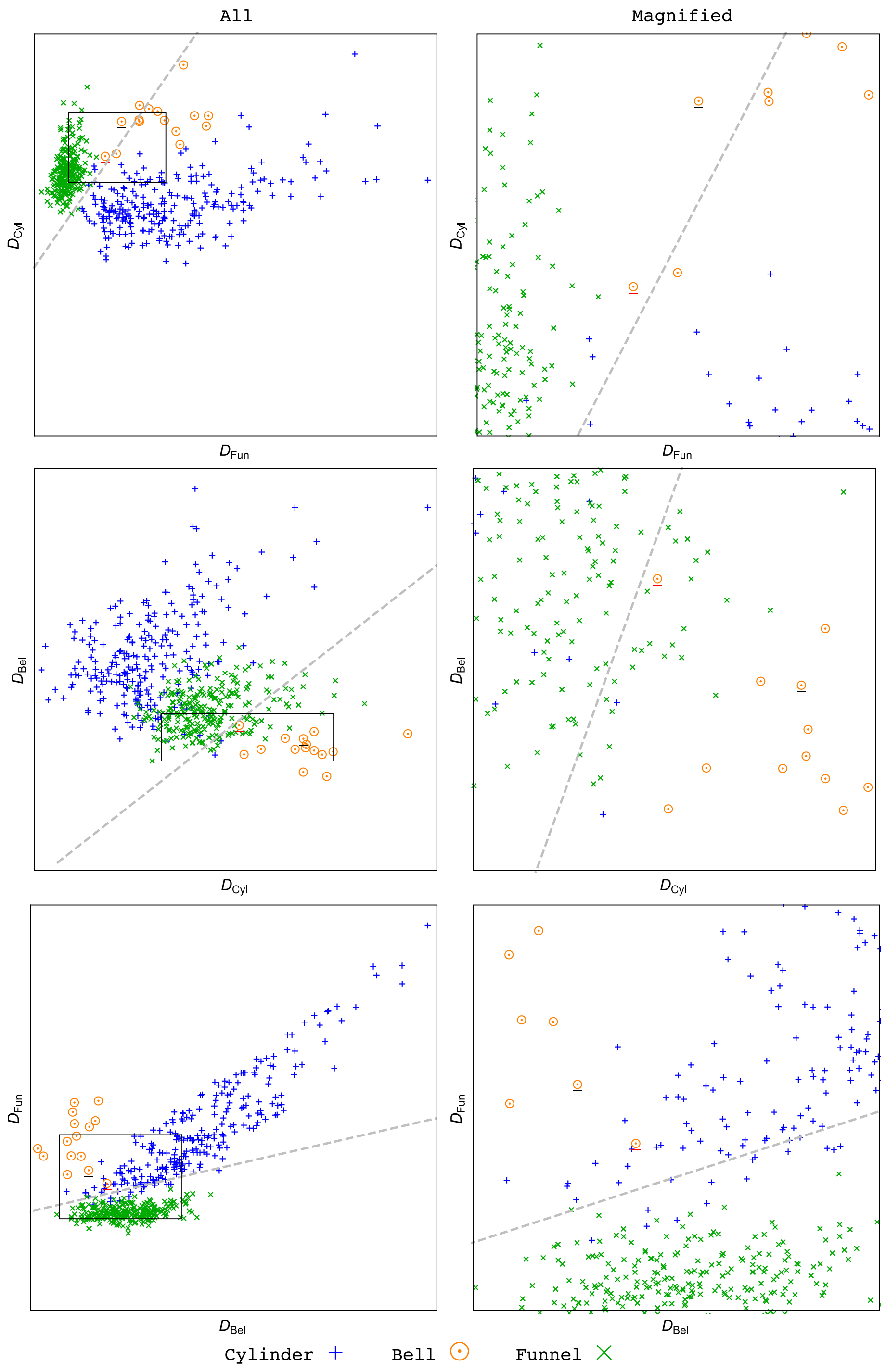


Fig. 5 2-D projections of the distance feature spaces (CBF)

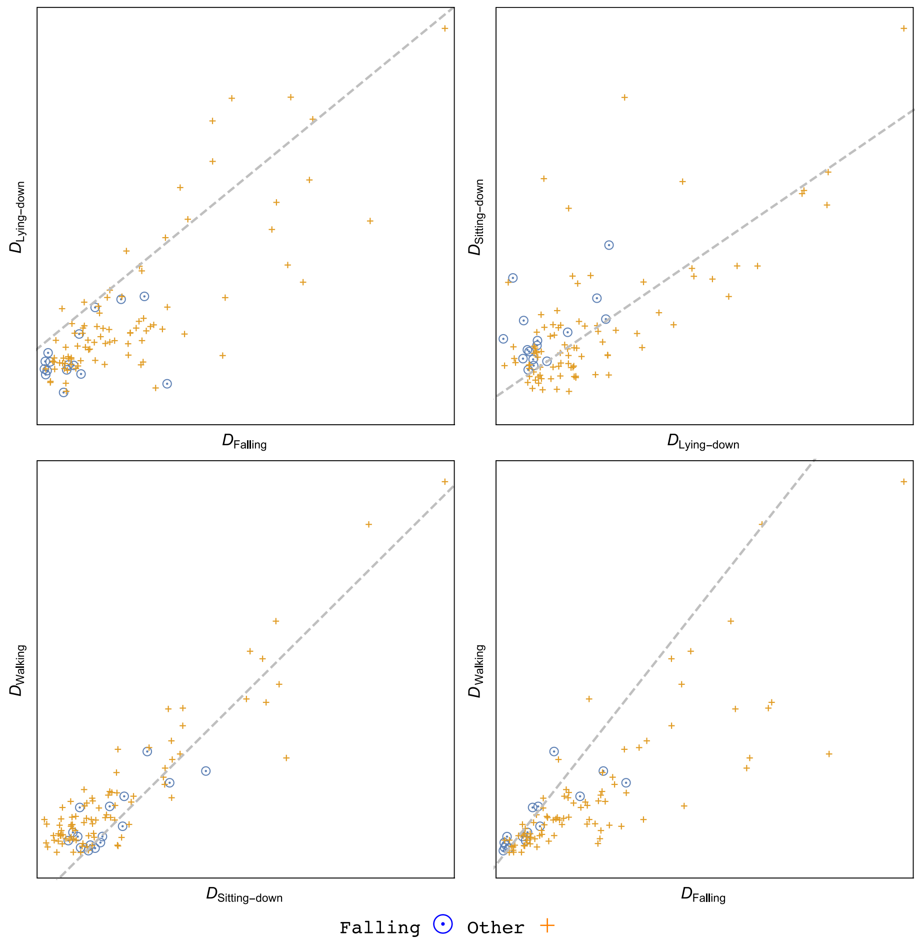


Fig. 6 2-D projections of the distance feature space (person activity)

In Fig. 5, three plots on the left column illustrate the same set of test instances. The instances of Cylinder, Bell, and Funnel classes are shown by $+$, $\odot$, and $\times$. There are three distance features, which represent the weighted distance from the nearest Cylinder, Bell, or Funnel class instances, indicated as: D_{Cylinder}, D_{Bell}, or D_{Funnel}, respectively. The x- and y-axes on each plot correspond to a different combination of features. The rectangular regions in the left column is magnified in the plot to the right. The dashed line in each plot shows the boundary where the two weighted distances are equivalent, i.e., for the instances below the boundary, the class corresponding to the y-axis is closer and for the instances the boundary line, the class corresponding to the x-axis is closer.

In Fig. 5, the class distributions in the distance feature space are well-separated overall, making the baseline performances are high as shown in Sect. 5.3. However, there exist some perplexities or overlaps in Fig. 5, from the generative noise. The proposed method can improve on the baseline performances by learning the weights, which adjusts the dashed lines, to optimize AUPRC.

The two instances of $\odot$ with underscores are highlighted for further examination. These target class instances are selected in particular because the k-NN algorithm, which is very

Table 3 Comparison of AUC with/without SCF (control chart)

ID	Setup		WCNN with SCF		Without SCF	
	Positive (+1)	Negative (−1)	AUROC	AUPRC	AUROC	AUPRC
i	Cyc, Inc, Dnw	Nrm, Dec, Upw	**0.977**	**0.977**	0.666	0.722
ii	Nrm, Inc, Dec	Cyc, Upw, Dnw	**0.994**	**0.987**	0.874	0.798
iii	Nrm, Dnw	Cyc, Inc, Dec, Upw	**1.00**	**0.998**	0.605	0.659
iv	Cyc, Inc	Nrm, Dec, Upw, Dnw	**0.980**	**0.972**	0.697	0.740

robust for CBF data as described in [41], err on them. The errors occur because their nearest Funnel class training instance is closer to them than the nearest Bell class training instance.

In the WCNN model, meanwhile, the nearest majority class between the Cylinder and the Funnel classes is first chosen. The plots on the top row show that these instances are closer to the funnel class. Its distance is then compared to the distance to the minority class, i.e., Bell. The plots on the bottom row shows that the two instances are closer to a Bell class instance than Funnel; therefore, these instances are correctly classified by WCNN.

Figure 6 illustrates the two-dimensional projections of the person activity dataset to the distance feature space. Four combinations of features are selected for the sake of conciseness, and the distinction is limited to Falling and all other classes for comprehension. The class distribution is highly perplexed for this dataset as shown in Fig. 6, and the baseline performance is much lower as a result. The training of WCNN was still able to improve the performance on such data as shown in Sect. 5.3.

5.5 Analysis on supplementary component features

In this section, we attempt to demonstrate the merit of the supplementary component features (SCF) for WCNN using artificial and real-world datasets. The performance of the WCNN classifier without SCF was used as the baseline for comparison. In order to show the effect of SCF explicitly, classes with distinct subclasses, which are the primary motivation of SCF, were generated in the artificially dataset. From the control chart dataset, the class values normal (Nrm), cyclic (Cyc), increasing (Inc), decreasing (Dec), upward-shift (Upw), and downward-shift (Dnw) were reassigned to either positive (+1) or negative (−1) classes. The re-assignment setups are shown in the first three columns of Table 3. The setups were designed such that each class comprises distinctly different patterns. No modifications were made on the time series of the Control Chart data. The two real-world datasets, Person Activity and Character Trajectories, were used without modification on the label or the time series data. The settings for cross-validation and the distance function were kept from Sect. 5.3.

The parameter of SCF, the minimum size of the largest uniform subtrees, λ, was set to 30 as suggested in Sect. 4.4. That is, clusters with more than λ leaves contributed to the distance features used by WCNN. The rest of the procedure is the same with and without SCF.

The results on the artificial datasets are shown in Table 3. The fourth and fifth columns show the AUC values of WCNN with SCF and the last two columns show the baseline performances. The higher values of AUPRC and AUROC are indicated in bold. Table 4 summarizes the performances on the real-world datasets. The baseline performances are indicated as ‘w/o SCF’. The higher values are indicated in bold.

In Table 3, the AUC values of the baseline show significant drop from the empirical results in Sect. 5.3. A graphical analysis in the distance feature space showed that the distributions of

Table 4 Comparison of AUPRC with/without SCF (real-world datasets)

	Falling	Lying-down	HW-g	HW-q	HW-y	HW-z
WCNN with SCF	**0.593**	**0.629**	0.984	0.994	**0.920**	0.985
WCNN w/o SCF	0.592	0.541	0.984	**0.995**	0.918	0.985

the positive and the negative classes largely overlap, which can be attributed to the perplexity of similarities between subclass patterns of majority and minority classes. For example in setup (i), Increasing and Upward-shift classes, which are respectively re-assigned to positive or negative classes, are similar to each other than to the other subclasses in the same class. Meanwhile with SCF, the WCNN classifiers maintained competitive AUC. Examining the clustering output, LUS corresponding to respective subclasses were identified, and the perplexity of the distance feature space were contained as a result.

In Table 4, a substantial improvement was made on the Lying-down class of the Person Activity dataset, for which the AUC was originally low. There were also small differences in two classes of the Character Trajectory dataset. No substantial decline from the baseline were observed.

In a well-prepared dataset, there is unlikely to be distinct subclasses of a minority class. However, majority classes often are unannotated in practice, and identifying substantial subcomponents in them invariably help reduce the loss of information when the pair-wise distance input is transferred to a distance feature space. Given that no detriments from using SCF were observed in the empirical results on the real-world data, we believe that the use of SCF with WCNN can be beneficial while carefully managing the possible demerits such as overfitting from generating trivial and too many distance features.

6 Conclusion

This paper proposed an extension of the nearest neighbor density estimation with a class-wise weighting to address the issues of class imbalance. Further, an optimization problem for learning the weights with respect to a nonlinear performance measure over the training set was presented. The resulting model has a strong mathematical support for performance optimization based on the formulation of a structural classifier proposed in this paper and maintains as much applicability to distance-based data as the instance-based model.

In the empirical study, the proposed method outperformed other methods for imbalanced distance space data consistently. The results indicate that the introduction of the optimization technique can contribute to a significant advantage for problems with this type of representation.

An important principle in the proposed approach is addressing the main problem, optimizing the performance measure over the minority class, directly rather than dividing the problem in two parts: balancing the class distribution of the data, then learning the classifier. Note that it is not our claim that the instance-weighting approach has an inherent advantage over other approaches for class imbalance. In fact, the intuitions for avoiding the over-generalization to the majority class are similar to the re-sampling approach. The key difference is the convex optimization technique for training its parameter vectors in relation to the global objective. In our view, the advantage of avoiding the computationally and knowledge intensive alternative of a grid of search by validation is substantial in practice.

Acknowledgments The authors would like to thank the handling editor and the anonymous reviewers for their valuable and insightful comments. This study is supported by the Grant-in-Aid for Young Scientists (B) 25730127 by the Japanese Ministry of Education, Culture, Sports, Science and Technology.

Appendix 1: Structural SVM learning

A structural classifier addresses a problem with multivariate input or output variables that are structured or dependent [10,11]. The dependency is captured in the max-margin learning formulation using a *feature* function and an *optimizer*. The *feature function* $\varPsi$ generates an arbitrary representation from a pair of input/output values $(\mathbf{x}, y)$. A discriminant function $F : X \times Y \rightarrow \mathbb{R}$ is then defined as an inner product of a weight parameter $\mathbf{w}$ and $\varPsi$

$$F(\mathbf{x}, y; \mathbf{w}) = \langle \mathbf{w}, \varPsi(\mathbf{x}, y)\rangle \tag{30}$$

From F and the optimizer that selects a class over Y, the decision function $f(\mathbf{x}; \mathbf{w})$ is defined as

$$f(\mathbf{x}; \mathbf{w}) = \arg\max_{y} F(\mathbf{x}, y; \mathbf{w}) \tag{31}$$

A typical example of the structural classifier is the multiclass SVM described in [11]. Let $\{c_j\}_{j=1}^{n}$ denote the class values and $\varLambda(c_j) = \mathbf{e}_j$ the *binary encoding* of c_j, i.e., a unit vector whose jth element is 1. The feature function $\varPsi$ is defined as

$$\varPsi(\mathbf{x}, y) = \mathbf{x} \otimes \varLambda(y) \tag{32}$$

where $\otimes$ denotes the tensor product.

Substituting the feature function (32) and a stack of vectors $\mathbf{w} = [\mathbf{w}'_1, \ldots, \mathbf{w}'_p]$ as parameters into (30), the discriminant function produces a set of margins for respective classes. Subsequently, the class predicted by (31) is the class with the largest margin.

Training of the structural classifier is formulated in a quadratic programming problem [10]. Given the training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{\eta}$,

Problem 2

$$\arg\min_{\mathbf{w}, \xi \geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + C\xi$$

subject to $\forall i, \forall y \neq y_i$,

$$\langle \mathbf{w}, \varPsi(\mathbf{x}_i, y_i)\rangle - \langle \mathbf{w}, \varPsi(\mathbf{x}_i, y)\rangle \geq 1 - \xi \tag{33}$$

An approximated solution of Problem 2 can be obtained by a cutting-plane algorithm [22].

Appendix 2: Multiclass formulation of SNN classifier

This section describes the formulation of SNN classifier learning with multiple majority and minority classes. Similar to the binary classification problem described in Sect. 4.2, the main intuition is to maximize the margin between the correct input and all other cases through the constrained minimization of the $\ell 2$-norm.

Let $\mathscr{M} = \{i\}_{i=1}^{m}$ and $\mathscr{N} = \{m+j\}_{j=1}^{n}$, denote the class values of the majority and minority classes, respectively. Note that the class values of $\mathscr{N}$ differ from Sect. 4.2. For mathematical convenience, a matrix notation W is introduced to represent the weights considered in the selection of the nearest neighbors among different classes. The relation between W and the weight vector $\mathbf{w}$ is defined as follows:

$$W = \begin{bmatrix} \begin{matrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{matrix} & M \\ M' & \begin{matrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{matrix} \end{bmatrix}_{(m+n)\times(m+n)} \tag{34}$$

where

$$M = \left[\frac{w_i}{w_j}\right]_{m\times n}, \quad M' = \left[\frac{w_j}{w_i}\right]_{n\times m}$$

W can be computed automatically from each $\mathbf{w}$ and is not shown in the main content to avoid the overuse of notations.

Let us define the decision function of an individual input as follows:

$$f'(X; W) = \underset{y\in\mathscr{M}\cup\mathscr{N}}{\arg\max}\ W^\top(\mathbf{u}(y) \odot \mathbf{d}) \tag{35}$$

where $\odot$ denotes the Hadamard product and $\mathbf{u}(y)$ the unit vector which has 1 at yth position.

In (35), the yth row of W is considered in choosing the nearest neighbor. From the definition of W, the weights are equivalent among the minority classes if $y \in \mathscr{M}$. Otherwise, i.e., $y \in \mathscr{N}$, the weights are equivalent among the majority classes.

The decision functions for the structural input is written as a summation of f',

$$f(X; W) = \arg\max_{Y} \sum_h W^\top (\mathbf{u}(y_h) \odot \mathbf{d}(x_h)) \tag{36}$$

Based on (36), let us define the feature function Ψ as follows

$$\Psi(X, Y) = \sum_{h=1}^{\eta} \mathbf{u}(y_h) \odot \mathbf{d}(x_h)$$

and the discriminative function F as a matrix product of W and Ψ.

$$F(X, Y; W) = W^\top \Psi(X, Y)$$

Using Ψ and F, (36) is rewritten in the same form as (26)

$$f(X; W) = \arg\max_{Y} F(X, Y; \mathbf{w}) \tag{37}$$

From (37), the constrained $\ell2$-norm minimization problem is formulated as follows.

Problem 3

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + C\xi$$

subject to $\forall \mathbf{z} \in (\mathscr{M} \cup \mathscr{N})^\eta \backslash \{\mathbf{y}\}$

$$W^\top\Psi(\mathbf{z}, X) - W^\top\Psi(Y, X) \geq \Delta(\mathbf{z}, Y) - \xi \tag{38}$$

Problem 3 can be solved by the cutting-plane method described in Sect. 4.

References

1. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
2. Chawla N, Japkowicz N, Kotcz A (2004) Editorial: special issue on learning from imbalanced data sets. SIGKDD Explor Newsl 6(1):1–6
3. Chan P, Stolfo S (1998) Toward scalable learning with non-uniform cost and class distributions: a case study in credit card fraud detection. In: Proceedings of the fourth international conference on knowledge discovery and data mining, pp 164–168
4. Chawla NV, Cieslak DA, Hall LO, Joshi A (2008) Automatically countering imbalance and its empirical relationship to costs. Data Min Knowl Discov 17(2):225–252
5. Köknar-Tezel S, Latecki LJ (2011) Improving SVM classification on imbalanced time series data sets with ghost points. Knowl Inf Syst 28(1):1–23
6. Li Y, Zhang X (2011) Improving k nearest neighbor with Exemplar generalization for imbalanced classification. In: Proceedings of the 15th Pacific-Asia conference on advances in knowledge discovery and data mining, vol 2, PAKDD'11, pp 321–332
7. Liu W, Chawla S (2011) Class confidence weighted kNN algorithms for imbalanced data sets. In: Proceedings of the 15th Pacific-Asia conference on advances in knowledge discovery and data mining, vol 2, PAKDD'11, pp 345–356
8. Domingos P (1999) MetaCost: a general method for making classifiers cost-sensitive. In: Proceedings of the 5th SIGKDD international conference on knowledge discovery and data mining, pp 155–164
9. Sun Y, Kamel MS, Wong AKC, Wang Y (2007) Cost-sensitive boosting for classification of imbalanced data. Pattern Recogn 40(12):3358–3378
10. Joachims T, Finley T, Yu CNJ (2009) Cutting-plane training of structural SVMs. Mach Learn 77:27–59
11. Tsochantaridis I, Joachims T, Hofmann T, Altun Y (2005) Large margin methods for structured and interdependent output variables. J Mach Learn Res 6:1453–1484
12. Hido S, Kashima H (2008) Roughly balanced bagging for imbalanced data. In: Proceedings of the SIAM international conference on data mining. SDM 2008, pp 143–152
13. Wallace BC, Small K, Brodley CE, Trikalinos TA (2011) Class imbalance, redux. In: Proceedings of the 2011 IEEE 11th international conference on data mining. ICDM'11, pp 754–763
14. Chen S, He H, Garcia EA (2010) Ramoboost: ranked minority oversampling in boosting. Trans Neural Netw 21(10):1624–1642
15. Masnadi-Shirazi H, Vasconcelos N (2010) Risk minimization, probability elicitation, and cost-sensitive SVMs. In: Proceedings of the 27th international conference on machine learning, pp 759–766
16. Holte RC, Acker LE, Porter BW (1989) Concept learning and the problem of small disjuncts. In: Proceedings of the 11th international joint conference on artificial intelligence, pp 813–818
17. Batista G, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. SIGKDD Explor Newsl 6(1):20–29
18. Napierala K, Stefanowski J (2012) BRACID: a comprehensive approach to learning rules from imbalanced data. J Intell Inf Syst 39(2):335–373
19. Ando S (2012) Performance-optimizing classification of time-series based on nearest neighbor density approximation. In: 2012 IEEE 12th international conference on data mining workshops (ICDMW), pp 759–764
20. Calders T, Jaroszewicz S (2007) Efficient AUC optimization for classification. In: Proceedings of the European conference on principles and practice of knowledge discovery in databases, pp 42–53
21. Yue Y, Finley T, Radlinski F, Joachims T (2007) A support vector method for optimizing average precision. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval. SIGIR '07, pp 271–278
22. Joachims T (2006) Training linear SVMs in linear time. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '06, pp 217–226
23. Fukunaga K (1990) Introduction to statistical pattern recognition, computer science and scientific computing, 2nd edn. Elsevier science, Amsterdam
24. Angiulli F, Basta S, Pizzuti C (2006) Distance-based detection and prediction of outliers. IEEE Trans Knowl Data Eng 18(2):145–160
25. Tomek I (1976) Two modifications of CNN. IEEE Trans Syst Man Cybern SMC–6(11):769–772
26. Covões TF, Hruschka ER, Ghosh J (2013) A study of k-means-based algorithms for constrained clustering. Intell Data Anal 17(3):485–505
27. Zeng H, Cheung Ym (2012) Semi-supervised maximum margin clustering with pairwise constraints. IEEE Trans Knowl Data Eng 24(5):926–939
28. Joachims T (2005) A support vector method for multivariate performance measures. In: Proceedings of the 22nd international conference on machine learning. ICML '05, pp 377–384

29. Pham DT, Chan AB (1998) Control chart pattern recognition using a new type of self-organizing neural network. Proc Inst Mech Eng Part I J Syst Control Eng 212(2):115–127
30. Saito N (1994) Local feature extraction and its applications using a library of bases. Ph.D. thesis, Yale University, New Haven, CT, USA
31. Bache K, Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml
32. Kaluža B, Mirchevska V, Dovgan E, Luštrek M, Gams M (2010) An agent-based approach to care in independent living. In: Ambient intelligence, vol 6439 of lecture notes in computer science. Springer, Berlin, pp 177–186
33. Zhang H, Berg AC, Maire M, Malik J (2006) SVM-KNN: discriminative nearest neighbor classification for visual category recognition. In: Proceedings of the 2006 IEEE computer society conference on computer vision and pattern recognition, vol 2, pp 2126–2136
34. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Int Res 16(1):321–357
35. Chawla NV, Lazarevic A, Hall LO, Bowyer K (2003) Smoteboost: improving prediction of the minority class in boosting. Knowledge discovery in databases: PKDD 2003, lecture notes in computer science. Springer, Berlin, vol 2838, pp 107–119
36. Weinberger KQ, Saul LK (2009) Distance metric learning for large margin nearest neighbor classification. J Mach Learn Res 10:207–244
37. Verikas A, Gelzinis A, Bacauskiene M (2011) Mining data with random forests: a survey and results of new tests. Pattern Recogn 44(2):330–349
38. Chang C, Lin C (2001) LIBSVM: a library for support vector machines
39. Flach PA, Hernández-Orallo J, Ramirez CF (2011) A coherent interpretation of AUC as a measure of aggregated classification performance. In: Proceedings of the 28th international conference on machine learning, ICML 2011, pp 657–664
40. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
41. Xi X, Keogh E, Shelton C, Wei L, Ratanamahatana CA (2006) Fast time series classification using numerosity reduction. In: ICML '06: Proceedings of the 23rd international conference on machine learning, pp 1033–1040

Shin Ando was born in July of 1976. He has been a Junior Associate Professor at the School of Management, Tokyo University of Science since April of 2014. He was an assistant professor at the Department Computer Science in Gunma University from April 2008 to March 2014 and an assistant researcher at Yokohama National University from April 2005 to March 2008. He was also an postdoctoral researcher at the Interdisciplinary School of Science and Engineering at Tokyo Institute of Technology from April 1993 to March 1996. He received his bachelor, masters, and doctoral degree from University of Tokyo in 1999, 2001, and 2004, respectively. His research fields include data mining and machine learning.