On novelty detection for multi-class classification using non-linear metric learning

Samuel Rocha Silva[a,*], Thales Vieira[b], Dimas Martínez[c], Afonso Paiva[a]

[a]*ICMC, Universidade de São Paulo, São Carlos, Brazil*
[b]*Instituto de Computação, Universidade Federal de Alagoas, Maceió, Brazil*
[c]*Departamento de Matemática, Universidade Federal do Amazonas, Manaus, Brazil*

Abstract

Novelty detection is a binary task aimed at identifying whether a test sample is novel or unusual compared to a previously observed training set. A typical approach is to consider distance as a criterion to detect such novelties. However, most previous work does not focus on finding an optimum distance for each particular problem. In this paper, we propose to detect novelties by exploiting non-linear distances learned from multi-class training data. For this purpose, we adopt a kernelization technique jointly with the Large Margin Nearest Neighbor (LMNN) metric learning algorithm. The optimum distance tries to keep each known class's instances together while pushing instances from different known classes to remain reasonably distant. We propose a variant of the K-Nearest Neighbors (KNN) classifier that employs the learned distance to detect novelties. Besides, we use the learned distance to perform multi-class classification. We show quantitative and qualitative experiments conducted on synthetic and real data sets, revealing that the learned metrics are effective in improving novelty detection compared to other metrics. Our method also outperforms previous work regularly used for novelty detection.

Keywords: novelty detection, outlier detection, anomaly detection, metric learning, multi-class classification

*Corresponding author. Tel: +55-16-3373-9700

Email addresses: `samuelrs@usp.br` (Samuel Rocha Silva), `thales@ic.ufal.br` (Thales Vieira), `dimas@ufam.edu.br` (Dimas Martínez), `apneto@icmc.usp.br` (Afonso Paiva)

URL: `http://www.ic.ufal.br/professor/thales` (Thales Vieira)

1. Introduction

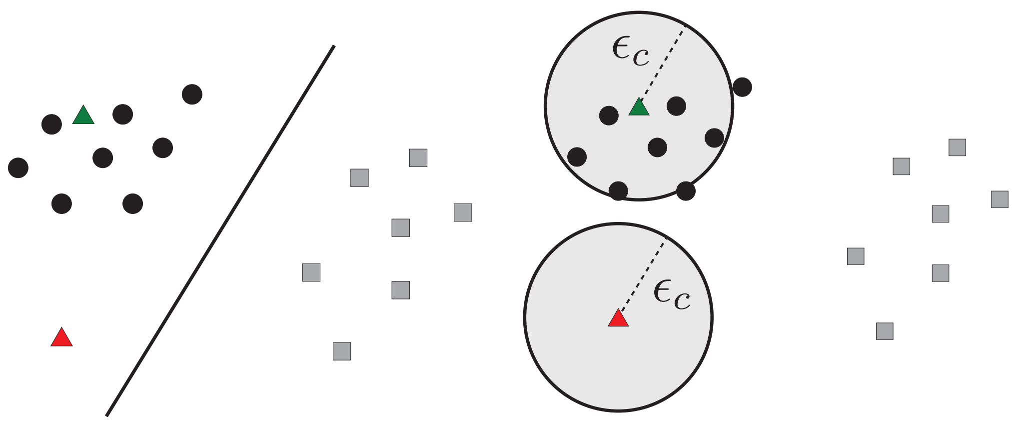
Novelty detection can be defined as the task of recognizing that test data differ in some respect from the data that are available during training (Pimentel et al., 2014). A large number of applications are covered by novelty detection methods including fraud detection (Jyothsna et al., 2011), medical diagnosis (Clifton et al., 2011), video surveillance (Diehl & Hampshire, 2002), mobile robotics (Sofman et al., 2011), among many others. It is worth mentioning that novelty detection is similar to anomaly and outlier detection and one-class classification, although it originated from different application domains. We refer the readers to the excellent surveys on novelty (Pimentel et al., 2014) and anomaly (Chandola et al., 2009) detection as well as one-class classification (Khan & Madden, 2010) for more information.

In general, novelty detection methods are applied when only data from a specific pattern is available, which is usually called the "normal" class, in contrast to novel data revealing a different pattern ("abnormal" class). In many real-world pattern recognition applications, however, many distinct patterns may be given during a training phase, and others may only appear over time (Faria et al., 2013).

As remarked by Bodesheim et al. (2013), it is not always possible to define a complete set of classes and acquire training instances from each of them. According to that authors, in multi-class novelty detection, one wants to detect whether a test sample is a novelty or if it belongs to one of many available classes, no matter to which class. For instance, training visual recognition systems require the trainer to provide images displaying examples of objects. However, the potential number of classes of real-world objects visible in an image is virtually infinite, and thus it is not possible to enumerate and collect training samples from all types of objects. Another critical problem where this situation occurs is gesture recognition. Many methods represent gestures as sequences of trainable classes of key poses, where only a few key poses are selected for training (Miranda et al., 2014b; Lv & Nevatia, 2007). However, during a gesture execution, some unimportant poses (novelties) will also occur in-between consecutive key poses. As a consequence, a key pose classifier must be capable of robustly identifying both: poses that are instances of a trained (normal) class; and unimportant poses that are not instances of any of the trained key pose classes, *i.e.* instances of abnormal classes.

It is worth emphasizing that the multi-class novelty detection problem is not well addressed by classifiers such as multi-class Support Vector Machine (SVM) (Schölkopf & Smola, 2002), where the boundary of each class is estimated by only requiring that it properly separates the regions of the space representing the trained classes. Consequently, SVM may incorrectly classify an unimportant pose as an instance of a trained class, as shown in Fig. 1a. To overcome such issues, we aim to provide a multi-class classification solution that includes a novelty detection step to filter abnormal data.

According to Pimentel et al. (2014), novelty detection methods can be classified as: (i) probabilistic, such as the Gaussian Mixture Models proposed in (Ilonen et al., 2006); (ii) reconstruction-based, in which we emphasize the *Kernel PCA* (KPCA) approach (Hoffmann, 2007); (iii) domain-based, including the well-known one-class SVM (Schölkopf et al., 2001); (iv)



(a) SVM boundary: the hyperplane (here a line) that better separates both classes is chosen as the boundary between them. The green triangle is correctly classified as an instance of the class of circles. However, the red triangle is incorrectly classified as an instance of the same class of circles, although it is a novelty (or outlier).

(b) Distance-based novelty filter: by considering distances between an input instance and training instances, the red triangle can be recognized as an outlier and filtered, since none of the training instances are inside the circle of radius ϵ_c centered in the red triangle. In contrast, the green triangle can be recognized as an inlier of the black circles class.

Fig. 1: SVM classifier limitations in a simple binary classification problem and the advantages of employing a distance-based novelty filter: black circles and grey squares represent training instances of two classes, and triangles represent input instances to be classified.

information-theoretic (Keogh et al., 2007); and (v) distance-based. We focus on the latter, more specifically on KNN approaches. Although a few novelty detection methods are based on the distance to the K-nearest neighbors, as in (Zhang & Wang, 2006), not much attention has been given to learning more sophisticated metrics.

In the last decade, the multi-class novelty detection problem has received special attention from researchers. In (Faria et al., 2013), an evaluation approach for multi-class data streams novelty detection problems was proposed. Also focused on data streams, de Faria et al. (2016) proposed an algorithm for novelty detection that addresses the problem as a multi-class task.

More related to our work, Bodesheim et al. (2013) presented a kernel null space based discriminant analysis for novelty detection known as KNFST, which still achieves state-of-the-art performance. To improve the scalability of the KNFST method, an incremental version was later proposed by (Liu et al., 2017), reducing computing time with similar accuracy.

We propose a multi-class novelty detection method that employs a non-linear distance learned from data to both detect novelties and classify normal samples from multi-class datasets. To the best of our knowledge, non-linear distances have never been employed for novelty detection. The optimum distance is found using a kernelized extension of the *Large Margin Nearest Neighbor* (LMNN) algorithm (Weinberger et al., 2006), named *Kernel Large Margin Nearest Neighbors* (KLMNN) (Chatpatanasiri et al., 2010)). Such distance tries to keep instances of the same class nearby while pushing instances with different labels and novelties to remain reasonably distant. Consequently, it provides relevant information for novelty detection, and also for multi-class classification. Firstly, novelty detection is performed by a variation of the well-known KNN classifier, using the learned distance and distance thresholds also learned from data. Then, normal samples are classified by a different proposed variant of the KNN classifier that also employs the learned distance and thresholds. It is worth mentioning that our solution is also suitable for anomaly and outlier detection.

We performed experiments to show that non-linear learned distances indeed outperform the use of other distances such as linear learned distances (LMNN) and the Euclidean distance when adopted by the same classifier. We also show results of comparisons revealing that our method outperforms previous work on novelty detection.

2. Non-linear metric learning

In this section, we review metric learning concepts and techniques that we employ in the proposed method. First, we introduce the Mahalanobis distance and give some key insights on how it could be used for novelty detection (Section 2.1). Then, we describe how to learn linear metrics from training data (Section 2.2) using the LMNN algorithm (Weinberger et al., 2006; Weinberger & Saul, 2009), in such a way that each instance is nearest to instances of its class than to instances of other classes. Finally, we briefly describe the KLMNN algorithm (Chatpatanasiri et al., 2010): a non-linear version of the LMNN algorithm built over the KPCA (Section 2.3).

2.1. Transforming the space with the Mahalanobis distance

Given a set $\mathcal{X} = \{\mathbf{x}_i\} \subset \mathbb{R}^n$ of data points, the original Mahalanobis distance is defined using the covariance matrix $\mathbf{C}$ of $\mathcal{X}$. However, in the metric learning literature, many methods aim to compute a distance given in the form:

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}\,(\mathbf{x}_i - \mathbf{x}_j), \tag{1}$$

where $\mathbf{M}$ is some positive semi-definite matrix found by optimization, instead of $\mathbf{C}$. Distance functions from this class of functions can be adequately represented by its matrix $\mathbf{M}$, and are usually called generalized Mahalanobis distance.

A key insight for such distances is that matrix $\mathbf{M}$ can be factorized by using Cholesky decomposition as $\mathbf{M} = \mathbf{G}^\top\mathbf{G}$, where a linear transformation is $\mathbf{G}$. Consequently, we have

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{G}\mathbf{x}_i - \mathbf{G}\mathbf{x}_j\|_2^2\,, \tag{2}$$

which means that any generalized Mahalanobis distance is just the squared Euclidean distance applied to linearly transformed data. Thus, the problem of learning an optimal metric for a specific problem can be reduced to the problem of optimally deforming the feature space $\mathbb{R}^n$, before applying the squared Euclidean distance. It is worth mentioning that distinct features may be concatenated to form the data points $\mathbf{x}_i \in \mathbb{R}^n$, including continuous, binary, and categorical data as well. In particular, binary features may be represented by an integer with two possible values (for instance, 0 and 1), and categorical features may be one-hot encoded.

In this paper, we require such distances to be learned from data, satisfying specific constraints. In what follows, we describe such constraints and appropriate metric learning techniques we adopted to solve the constrained optimization problem.

2.2. Large Margin Nearest Neighbor (LMNN)

The LMNN algorithm was originally presented with the goal of improving k-NN classification accuracy (Weinberger et al., 2006). The algorithm receives as input a labeled dataset $\mathcal{P} = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_m, c_m)\}$ with $\mathbf{x}_i \in \mathcal{X}$ and labels $c_i \in \mathcal{C}$, where $\mathcal{C}$ is a set of classes.

The aim is to find a generalized Mahalanobis distance that tries to keep instances of the same class as nearest neighbors, while repelling instances from different classes (impostors). More specifically, the aim is to minimize

$$\mathcal{L}(\mathbf{M}) = \sum_{(i,j)\in\mathcal{S}} d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \lambda \sum_{(i,j,k)\in\mathcal{R}} [1 + d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k)]_+, \qquad (3)$$

where $[\cdot]_+ = \max(\cdot, 0)$ and $\lambda \in \mathbb{R}$ is a multiplier associated with the penalty term. $\mathcal{S}$ is the set of all pairs (i, j) where $\mathbf{x}_j$ is one of the K nearest neighbors in the same class as $\mathbf{x}_i$ and $\mathcal{R}$ is the set of all tuples (i, j, k) such that $(i, j) \in \mathcal{S}$ and $\mathbf{x}_k$ is an instance from a different class. For more details, see (Weinberger & Saul, 2009). Moreover, we highlight three key insights:

1. According to the authors, KNN classification is improved when a learned distance function is employed (Weinberger et al., 2006). By attracting nearest neighbors from instances of the same class while repelling nearest neighbors from distinct classes, the learned linear transformation $\mathbf{G}$ tends to better separate clusters of data points according to their classes. This behavior can be observed in Fig. 2;
2. To the best of our knowledge, this strategy has never been experimented for novelty, anomaly, or outlier detection;
3. As already mentioned, the learned Mahalanobis distance is just a squared Euclidean distance calculated over data previously transformed by $\mathbf{G}$. However, a linear transformation of the feature space may not be sufficient to evaluate distances in a high-dimensional space correctly. In order to overcome this limitation, Chatpatanasiri et al. (2010) proposed KLMNN as a non-linear extension for the LMNN.

2.3. Kernel Large Margin Nearest Neighbor (KLMNN)

The KLMNN approach (Chatpatanasiri et al., 2010) relies on the *kernel trick* in KPCA derivation. The main advantage of this non-linear extension

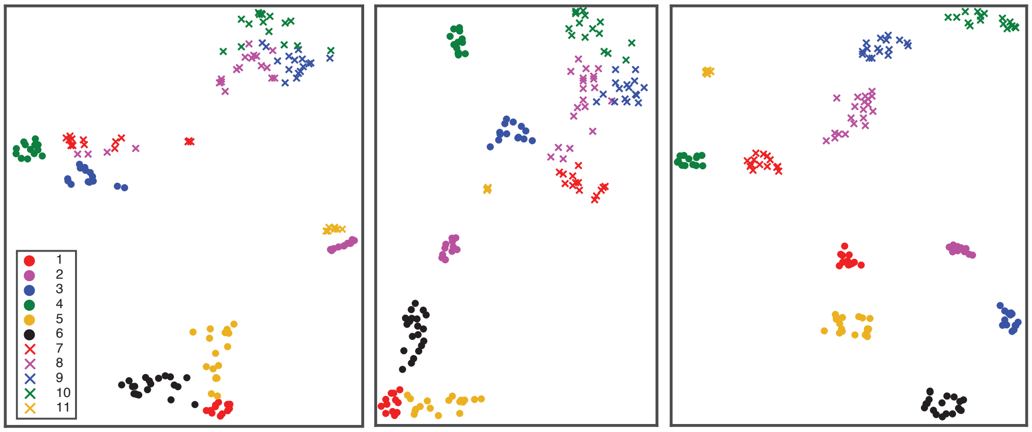


Fig. 2: Visualizing how LMNN and KLMNN improve data clusters: on the left, a t-SNE projection (Van Der Maaten, 2014) of a small training set composed of instances from 11 classes. After linearly transforming the data using LMNN (center), clusters are better separated from each other. Even better results, with more disjunct and spherical clusters, are achieved when the non-linear transformation computed using KLMNN is applied (right). Note that a multidimensional projection was applied because the dataset lies in $\mathbb{R}^{17}$.

is that it does not require the derivation of new mathematical formulas from the original LMNN formulation.

The first step of the framework is to (non-linearly) project the input dataset $\mathcal{X}$ using the KPCA (Schölkopf et al., 1997). It is known that there exists a non-linear, possibly very-high-dimensional mapping $\Phi\colon \mathbb{R}^n \mapsto \mathbb{R}^N$ capable of transforming $\mathcal{X}$ into a linearly separable set in a higher dimensional feature space $\mathbb{R}^N$ (J. Mercer, 1909). By applying kernelization, the KPCA implicitly uses the unknown map Φ to diagonalize the covariance matrix of the transformed data, given by

$$\widetilde{\mathbf{C}} = \frac{1}{m} \sum_{i=1}^{m} \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^\top .$$

Note that the number of eigenvectors of $\widetilde{\mathbf{C}}$ equals to $|\mathcal{X}|$, which is in general much higher than n. Thus, it is expected that many eigenvalues of $\widetilde{\mathbf{C}}$ will be very small, and consequently, a dimensionality reduction strategy may be applied. Following this idea, a transformed set $\widetilde{\mathcal{X}}$ can be computed by projecting the data points over the most relevant eigenvectors of $\widetilde{\mathcal{X}}$. From now on, we will denote the complete KPCA projection step by $\Pi : \mathcal{X} \to \widetilde{\mathcal{X}}$. In the following step, the original LMNN algorithm is applied to the transformed data to find the Mahalanobis matrix $\mathbf{M}$. Finally, from Equation (2), the non-linear learned distance is given by

$$\widetilde{d}_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{G}\Pi(\mathbf{x}_i) - \mathbf{G}\Pi(\mathbf{x}_j)\|_2^2 \,. \tag{4}$$

It is worth mentioning that the parameters of the KPCA projection must be kept to calculate distances between unprojected points. The superiority of using the non-linear composition $\mathbf{G}\circ\Pi$ to improve data clusters configuration, in comparison with the simple linear transformation from LMNN, can be seen in Fig. 2. It is also worth mentioning that, we compute KPCA using the Gaussian kernel with a σ hyperparameter tuned according to the procedure described in Section 4.1. For more details about the kernelization, we refer the reader to the papers (Schölkopf et al., 1997; Chatpatanasiri et al., 2010).

3. Multi-class novelty detection and classification

In this section, we present a multi-class novelty classifier that may also be employed to provide multi-class classification of normal data. Here, we mainly consider novelties (or abnormal data) or instances from classes not included in the training set. If a test instance is classified as a novelty, it is filtered. Otherwise, it follows for multi-class classification.

3.1. Novelty classifier training

Following the supervised learning approach, the proposed novelty classifier is trained from a labeled dataset $\mathcal{P}$ (defined in Section 2.2). The training phase is comprised of two steps: non-linear metric learning and distance threshold estimation.

3.1.1. Non-linear metric learning

In the first step, the KLMNN framework (Section 2.3) is employed to learn a non-linear distance function $\widetilde{d}_{\mathbf{M}}$ from $\mathcal{P}$. From another perspective, a non-linear transformation of the feature space $\mathbb{R}^n$ is learned better to reposition inliers (normal data) in well-defined clusters and isolate outliers (abnormal data).

Consequently, we expect inliers to be near the K nearest neighbors of their corresponding classes, and far from instances of other normal classes, as illustrated in Fig. 2 (right). Besides, our central hypothesis is that inliers

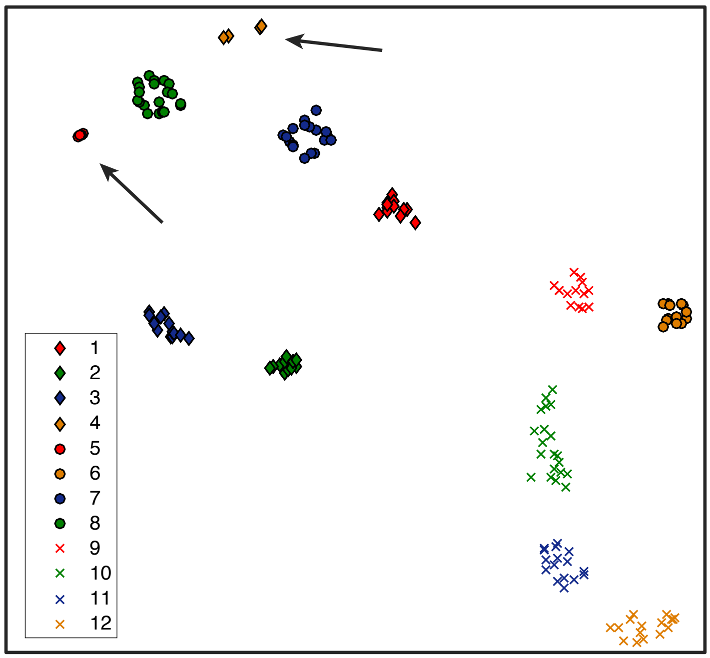


Fig. 3: Visualizing the behavior of abnormal classes after non-linearly transforming the space using instances from 10 trained classes, through t-SNE projections: instances from each inlier class are attracted to its representative cluster, while instances from two untrained classes (4 and 5) are isolated, as indicated by the arrows. In particular, instances of untrained class 4 are the only ones that are not well clustered, since classes 4 and 5 instances are not considered in the distance learning optimization (Equation (3)).

should become far from instances of abnormal classes (outliers) because it is expected that such outliers would lie outside the attracting region of each class. This idea is visually validated in the projection shown in Fig. 3, where outliers from two untrained (abnormal) classes are far from the regions of ten trained classes.

3.1.2. Distance threshold estimation

We propose a distance-based filter to detect and filter novelties composed of data from untrained classes and/or noise. To evaluate if a point $\mathbf{x}$ is an inlier of a specific class c, we take into account the distance from $\mathbf{x}$ to its K-nearest neighbors, considering only training instances from c.

Since classes may be represented by point sets with different density, we compute distance thresholds individually for each class. Considering the set

$\mathcal{X}^c = \{\mathbf{x}_i \mid (\mathbf{x}_i, c) \in \mathcal{P}\}$, *i.e.* the instances of class c in the training set $\mathcal{P}$. For each $\mathbf{x}_i^c \in \mathcal{X}^c$, we use $\widetilde{d}_{\mathbf{M}}$ to find the distance d_i^c from $\mathbf{x}_i^c$ to its K-nearest neighbors in $\mathcal{X}^c$. The distance threshold for class c is set to

$$\epsilon_c = \tau \left[\mathrm{mean}(\mathcal{D}^c) + \mathrm{std}(\mathcal{D}^c)\right] \quad \text{with} \quad \mathcal{D}^c = \left\{d_i^c \mid i = 1, \ldots, |\mathcal{X}^c|\right\},$$

where $\mathrm{mean}(\mathcal{D}^c)$ and $\mathrm{std}(\mathcal{D}^c)$ are respectively the arithmetic mean and standard deviation of $\mathcal{D}^c$, and $\tau \in \mathbb{R}$ is a tolerance to strengthen or soften the novelty classifier. We empirically found that $\tau = 1.2$ showed good results, although a cross-validation procedure could be applied for better tuning, as discussed in Section 4.1.

3.2. Novelty filtering

Let $\mathbf{x} \in \mathbb{R}^n$ be an input instance (from an unseen example) and its neighborhood $\mathcal{N}^c(\mathbf{x}) = \left\{\mathbf{x}_i \in \mathcal{X}^c \mid \widetilde{d}_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_i) < \epsilon_c\right\}$. An input instance $\mathbf{x}$ is considered to be an outlier of class c if $|\mathcal{N}^c(\mathbf{x})| < \kappa$, where $\kappa \leq K$ is an integer hyperparameter. Consequently, $\mathbf{x}$ is filtered as a novelty if $\mathbf{x}$ is an outlier for all trained classes. Only instances that pass this step will be considered for classification.

Fig. 1b depicts an example where a red instance would be filtered as a novelty since no training instance is inside its circle.

3.3. Inlier classification

We propose a modified KNN method to classify inlier instances, making use of the distance thresholds estimated in the training phase. We aim to take into account the density of each set $\mathcal{X}^c$ for classification. For instance, we consider that compact clusters, such as the cluster representing class 11 of Fig. 2 (right), should only influence classification if an instance is very near the cluster. On the opposite, the larger spread of class 8 data implies a larger influence region.

Given an input instance $\mathbf{x}$, let $\mathcal{K}(\mathbf{x}) \subset \mathcal{P}$ be the set of the K nearest *valid* neighbors of $\mathbf{x}$, defined as

$$\mathcal{K}(\mathbf{x}) = \left\{(\mathbf{x}_i, c_i) \in \mathcal{P} \;\middle|\; \widetilde{d}_{\mathbf{M}}(\mathbf{x}, \mathbf{x}_i) < \epsilon_c\right\}.$$

To find $\mathcal{K}(\mathbf{x})$, we recursively search for the K_i nearest neighbors of $\mathbf{x}$ using a k-d tree data structure built over $\mathcal{P}$. We initiate the search with $K_1 = K$, and stop if at least K nearest neighbors are valid. Otherwise, the search continues for the K_{i+1} nearest neighbors with $K_{i+1} = 2K_i$, until the condition is satisfied. Thus, $\mathbf{x}$ is classified as the mode of the labels of $\mathcal{K}(\mathbf{x})$.

4. Experiments

In this section, we describe experiments thoroughly performed on synthetic and real datasets to validate, evaluate, and compare the proposed methodology to well-established methods available in the literature. We provide experiments setup details in Section 4.1. Simulation studies on synthetic data are described in Section 4.2, including computational performance evaluations. Then, the results of a comparison between our method and other novelty detection approaches on real datasets are revealed in Section 4.3. Finally, in Section 4.5, we described a simple proof-of-concept visual experiment focused on gesture recognition applications.

4.1. Experiments Setup

We intended to answer the following research questions in our experiments:

Q.1 How does the proposed novelty detection method perform on linearly and non-linearly separable data?

Q.2 How does the proposed method accuracy behave when the number of training instances increases?

Q.3 Is the proposed method robust to the curse of dimensionality?

Q.4 How does the proposed method compares to well established and state-of-the-art novelty methods?

Q.5 How does the proposed method training time increase when the number of training instances and/or the number of features increase?

Q.6 How does the proposed method perform on real noisy datasets?

To answer such questions, we adopted two specific protocol for cross-validation and hyperparameters tuning, and two appropriate error metrics, which we detail next. Then, we briefly describe the compared methods.

Cross-validation protocols. We aim to evaluate how novelty classifiers, including ours and other compared methods, perform to discriminate data from trained (normal) classes and data from classes that are not included in the training set (abnormal). For this purpose, we propose two cross-validation protocols: the first for simulation studies in synthetic datasets and the second for experiments on real datasets.

P.1 To conduct simulation studies on each synthetic dataset, we build a supervised training set $\mathcal{P}_t$, representing the negative class (no novelties), comprised of synthetically generated instances of each normal class and including their corresponding labels. We define negative classes using analytical models, which we detail in Section 4.2. It is worth emphasizing that data from at least two classes is required to learn the non-linear metrics. To tune the hyperparameters, random samples from an abnormal/positive class are synthetically generated to compose $\mathcal{P}_u$. We then perform a uniform grid search with ten validation experiments per hyperparameter configuration using data from $\mathcal{P}_t$ and $\mathcal{P}_u$. The hyperparameter configuration with the best average performance in the ten experiments is selected for the evaluation. Finally, a different test set is composed of synthetic data uniformly sampled from the analytical model (negative class) and the feature space (positive class). More details on the error metrics and hyperparameters tuning procedure follow at the end of this section. The number of samples employed for each experiment is detailed in Section 4.2.

P.2 For real datasets, in each experiment, we exclude all instances of several classes randomly selected from the set of classes. More specifically, let $\mathcal{C}$ be the set of classes of a dataset $\mathcal{P}$. We randomly sample a few classes from $\mathcal{C}$ to compose the trained class set $\mathcal{C}_t$, and the remaining are included in the untrained class set $\mathcal{C}_u$. We denote by $\mathcal{P}_u$ and $\mathcal{P}_t$ subsets of instances of $\mathcal{P}$ belonging to $\mathcal{C}_u$ and $\mathcal{C}_t$, respectively. For cross-validation purposes, the training set is composed of a random split of 80% of the instances in $\mathcal{P}_t$. The remaining instances of $\mathcal{P}_t$ are randomly included in the test set (20%). Besides, the test set receives the same ratios of instances (20%) from $\mathcal{P}_u$. Each random selection of classes is repeated ten times so that different subsets of classes are considered normal/abnormal, and the random split for training/test is performed. For instance, in a skeleton body poses dataset (described in Section 4.3), we randomly exclude all instances of 9 key pose classes, using only data from the remaining nine key poses for training. In this manner, we can verify if poses from untrained classes are correctly filtered as novelties. In practice, if an individual performs a body pose which was not trained before (such as a pose similar to the instances of one of the nine excluded key pose classes), the classifier should succeed in filtering such pose as a novelty, instead of classifying it as one of the trained key poses. To tune the hyperparameters, we performed a uniform grid search with ten validation experiments per hyperparameter configuration. The hyperparameter configuration with the best average performance in the ten experiments is selected, and the respective average score is used as a result of the final evaluation of the model.

Error metrics. To evaluate the accuracy of the novelty detection classifiers, we adopted two appropriate metrics for binary classification problems: the F_1 score and the Matthews correlation coefficient (MCC). As we describe next, both can be written in terms of TP (true positives); TN (true negatives), FP (false positive), and FN (false negatives). We consider abnormal data (novelties) to be a positive class and normal data to be the negative class. In both metrics, higher scores are better.

- F_1 *score*: a measure of accuracy of the classifier, defined as the harmonic mean of precision and recall, as follows

$$F_1 = \frac{TP}{TP + 0.5(FP + FN)} \, .$$

- *Matthews correlation coefficient* (MCC): a more informative measure of accuracy for binary classifier, which is generally better than F_1 score for novelty detection problems, since it takes into account the balance ratios of the four confusion matrix categories:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \, .$$

In the following experiments, we consider the MCC as the reference error metric, although the F_1 score is also shown.

Hyperparameters tuning details. In all experiments, we tune hyperparameters of our approach, and also each compared methods to perform fair comparisons. We applied uniform grid searches over the hyperparameters space of each method, considering the resulting MCC metric over validation data from the cross-validation approach. Our experiments revealed that such calibration was crucial to achieve great results, as exemplified in Fig. 4: the non-linear classifier outperformed the other metrics only when the Gaussian kernel parameter (σ) was assigned to a value in a specific interval. For all

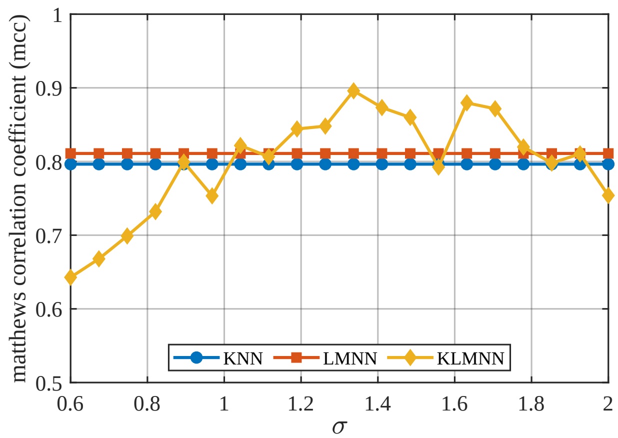


Fig. 4: Hyperparameter tuning on the real KGD dataset (see Section 4.3): outlier filtering achieved better results when our non-linear metric was employed compared to the Euclidean and linear metric. However, hyperparameter tuning was essential, since only when kernel parameter σ in the interval $[1.0, 1.8]$ were used for KPCA projection, resulting in non-linear distance outperformed the other distances (note that the other distances do not depend on σ).

KNN-based methods, we also performed cross-validation experiments to tune the hyperparameters K, κ, and τ for each dataset. For K and κ, we considered all pairs of integers in the range $[1, 5]$ where $\kappa \leq K$. We evaluate 50 equally spaced values in the range $[0.5, 3.0]$ for τ. Tables 2 and 4 reveal results of tuning experiments on a synthetic and on a real dataset, respectively.

Compared methods. We perform comparisons with well established previous work regularly used for novelty and anomaly detection: one-against-all multi-class SVM (Hsu & Lin, 2002) (MCSVM), in which an instance is classified as novelty if it does not belong to any trained class; one-class SVM (Schölkopf et al., 2001) (OSVM); KPCA for novelty detection (Hoffmann, 2007) (KP-CANOV); and Kernel Null Space Method (Bodesheim et al., 2013) (KNFST), which was recently enhanced by Liu et al. (2017) and still achieves state of the art accuracy. We also compare the usage of our proposed non-linear metrics (KLMNN) with linear (LMNN) and Euclidean (KNN) metrics. For all compared methods, we experimented with both the usual Gaussian and polynomial kernels. Regarding the KNFST, we adjust the kernel parameter and a decision threshold that varies from zero to the minimum distance between the trained classes' target points. For the KPCANOV, we adjust the kernel parameter and the number of relevant eigenvectors to the feature space's error reconstruction. For the OSVM algorithm, we only adjust the kernel parameter. Already for MCSVM, both the kernel parameter and a posterior probability threshold are adjusted. Although this posterior probability is not standard for SVM methods, it is an attractive way to estimate decision boundaries for support vector machines. More details can be found in the official documentation of the main libraries that implement SVM.

4.2. Simulation studies on synthetic data

To evaluate our novelty detection approach in different situations, we created simulation studies on four datasets depicted in Figures 5 and 6: three horizontal lines; three parabolas; three concentric circles; and four uniform distributions. Note that two datasets are linearly separable (horizontal lines and uniform distributions), and two are not linearly separable (parabola and concentric circles). The uniform distribution simulations will be employed to evaluate our novelty detection approach when the number of training examples or the number of features grows. For the process of hyperparameters tuning and methods evaluation, we apply the cross-validation protocol **P.1** (see Subsection 4.1).

In the first simulation, we created three classes with 25, 50, and 25 points from the sampling of 3 horizontal lines. Similarly, we created the simulation of parabolas and concentric circles. To assist the process of hyperparameters tuning, we added a fourth class of random samples composed of 192, 187, and 193 samples for the first, second, and third simulation, respectively.

During the evaluation of hyperparameters, we noticed that the kernel-based methods presented better results with a polynomial kernel of type $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^d$ than the traditional Gaussian kernel. On the other hand, for OSVM and MCSVM, the Gaussian kernel was more advantageous. To study the best hyperparameters K and κ for KNN, LMNN and KLMNN methods, we chose to apply the protocol **P.1** individually for different combinations of K and κ, so that the other hyperparameters are adjusted according to each combination.

Table 1 summarizes the results, where we also show the value of the F_1 score. The best methods were KLMNN and KNFST, which achieved maximum MCC and F_1 scores in all simulation studies. KPCANOV performed well for the simulation of horizontal lines and parabolas, while for the simulation of circles, the result was unsatisfactory. SVM-based methods were

Table 1: Comparison of the methods in 4 simulation studies (best results in bold).

simulation studies	train / validation		test		metrics	KNN-based methods			compared methods			
	normal	novelty	normal	novelty		KNN	LMNN	KLMNN	KNFST	OSVM	MCSVM	KPCANOV
horizontal lines	100	192	900	89 100	F_1	0.14	**1.00**	**1.00**	**1.00**	0.15	0.14	0.99
					MCC	0.25	**1.00**	**1.00**	**1.00**	0.26	0.26	0.99
parabolas	100	187	1 500	98 500	F_1	0.31	0.31	**1.00**	**1.00**	0.23	0.16	0.99
					MCC	0.30	0.30	**1.00**	**1.00**	0.32	0.24	0.99
concentric circles	100	193	3 300	197 970	F_1	0.16	0.16	**1.00**	**1.00**	0.10	0.17	0.25
					MCC	0.26	0.26	**1.00**	**1.00**	0.18	0.22	0.35

unsatisfactory in all simulations. These results can be attested visually by Fig. 5, which shows the decision regions for each method in the three simulations. Any point outside these regions is considered abnormal data.

For the simulation of horizontal lines, both linear (LMNN) and non-linear metric learning (KLMNN) were able to capture the exact distribution of the 3 lines. In this case, the MCC score was 1.0 for all tested K and κ combinations. On the other hand, for KNN the best result was only 0.25 with hyperparameters $K = 2$ and $\kappa = 1$.

Meanwhile, in the 3 parabolas simulation (Fig. 5, *middle*), we observed that the linear metric learning was insufficient to capture the correct distribution of the data. In this case, the best result for KNN and LMNN was 0.3 when $K = 1$ and $\kappa = 1$. In turn, with non-linear metric learning, it was possible to capture the exact distribution of the 3 parabolas, regardless of K and κ values. These results can be checked in more detail in Table 2, where κ is defined as a function of K.

Table 2: MCC of the KNN-based methods for different combinations of hyperparameters K and κ in the 3 parabolas simulation. The scores for values of κ less than 1 are omitted with symbol $\emptyset$. Best results in bold.

K	**KNN**				**LMNN**				**KLMNN**			
	κ				κ				κ			
	K	$K-1$	$K-2$	$K-3$	K	$K-1$	$K-2$	$K-3$	K	$K-1$	$K-2$	$K-3$
1	**0.30**	$\emptyset$	$\emptyset$	$\emptyset$	**0.30**	$\emptyset$	$\emptyset$	$\emptyset$	**1.00**	$\emptyset$	$\emptyset$	$\emptyset$
2	0.14	0.29	$\emptyset$	$\emptyset$	0.07	0.29	$\emptyset$	$\emptyset$	1.00	1.00	$\emptyset$	$\emptyset$
3	0.12	0.14	0.16	$\emptyset$	0.12	0.16	0.29	$\emptyset$	1.00	1.00	1.00	x
4	0.10	0.13	0.17	0.17	0.11	0.13	0.16	0.17	1.00	1.00	1.00	1.00
5	0.10	0.12	0.13	0.15	0.10	0.12	0.13	0.14	1.00	1.00	1.00	1.00

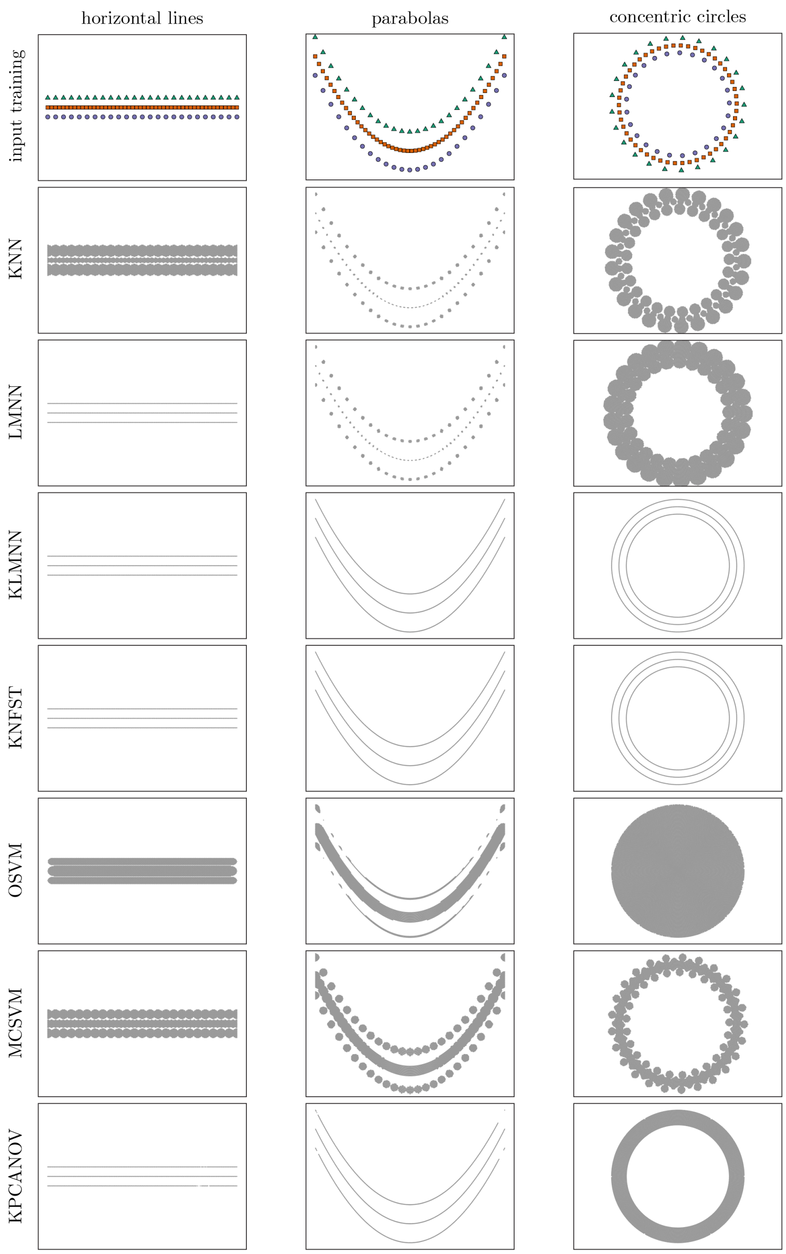


Fig. 5: Decision regions of all methods on simulation studies 1, 2 and 3. Every point belonging to the gray region is considered normal data, which indicates that they belong to a class of the training data. On the other hand, any point outside the region is considered novelty (abnormal class).

As in the previous case, in the simulation of concentric circles, only the KLMNN succeeded to correctly determine the circles. In this case, all tested combinations (K, κ) resulted in an MCC score of 1.0. In turn, with KNN and LMNN, the results were unsatisfactory, with a better score of only 0.26 for both methods when $K = 2$ and $\kappa = 1$.

Fig. 6 shows another simulation study to evaluate the novelty detection performance of our KNN-based approach when we increase the number of training samples and the number of dimensions (features) as well. In this case, we create five classes with uniform probability distribution in 5 hypercubes centered on the xy-plane of $\mathbb{R}^n$. For instance, in $\mathbb{R}^2$ classes are composed of random points in squares of side 0.9 centered on $(1.0, 0.5)$, $(-0.5, 1.0)$, $(-1.0, -0.5)$, $(0.5, -1.0)$ and $(0.0, 0.0)$. For the purpose of hyperparameters tuning and methods evaluation, we select one class and considered it as abnormal. In the first experiment, we fixed the number of dimensions at ten and varied the number of training samples by 100, 200, 300, 400, 500, 750, 1000, 1500, and 2000. While in the second experiment, we fixed the number of training points and varied the number of dimensions from 2 to 20. Again, we adopted the protocol **P.1** to adjust the hyperparameters, and the overall performance of novelty detection was calculated in a test set with 50K samples (40K from trained classes and 10K from the abnormal class). For both KNN, LMNN, and KLMNN, the other hyperparameters were adjusted with $K = 3$ and $\kappa = 1$. Fig. 7 depicts the results.

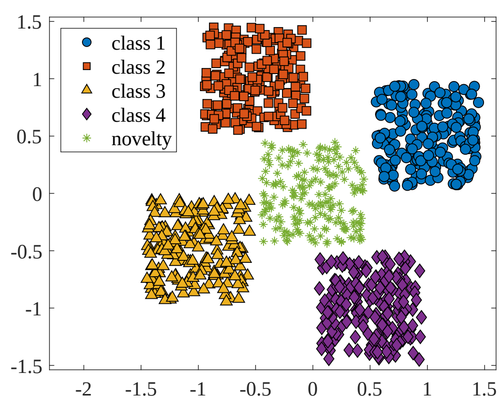


Fig. 6: Synthetic dataset with 5 hypercubes to evaluate the performance of the methods when the number of samples grows and also when the number of dimensions grows.

The first experiment (Fig. 7, *left*), when the number of training samples grows, our both LMNN and KLMNN improves the performance of KNN

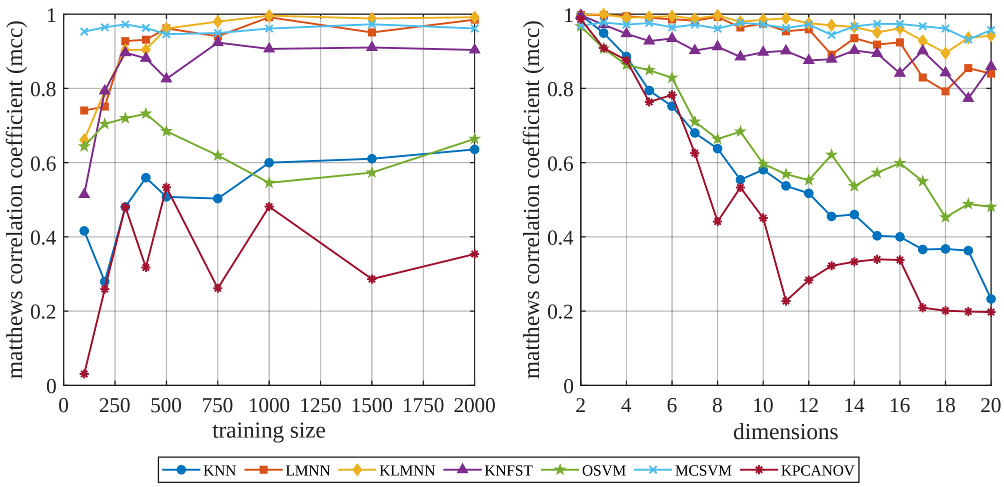


Fig. 7: Evaluation of methods with MCC when the number of training examples increases (*left*) and when the number of dimensions increases (*right*). All experiments were evaluated in a test set with 50K samples (40K normal and 10K novelty).

novelty detector, note that in the range 100 to 500 the MCC value it rises from approximately 0.66 to above 0.95 with KLMNN, reaching a value of 0.99 with 1000 and 2000 training samples. However, our version of the KNN with Euclidean distance improves from 0.40 to just 0.55 in the same range of training data, reaching a maximum of 0.63 with 2000 training samples. For the other methods, MCSVM proved to be efficient in this experiment, reaching a score of 0.95 in almost all cases. This possibly occurs due to the simplicity of positioning and distribution of the classes, since the multi-class SVM calculates hyperplanes that best separate the classes from each other. OSVM is inefficient in the same task, achieving MCC values ranging around 0.6, with a maximum value of 0.73. KNFST improves considerably in the range $[100, 750]$, varying from 0.50 to 0.92, but it remained at the same level of 0.90 for experiments with more than 750 training samples. KPCANOV had the worst performance, with MCC of at most 0.53 remaining the score unchanged, even increasing the training samples.

The second experiment (Figure 7, *right*), when the number of features grows, both linear and non-linear metrics mitigate the impact of the curse of dimensionality on the KNN. It is enough to note that in the KNN, the MCC metric varies from 0.99 in dimension 2 to 0.23 in dimension 20, while LMNN and KLMNN varied from 0.99 to 0.84 and 0.95, respectively. Regarding the compared methods, we see that both OSVM and KPCANOV had the same problem as KNN with Euclidean distance. In contrast, KNFST remained relatively stable, with an MCC value of 0.85 in dimension 20. MCSVM is also stable, with a score always around 0.95 due to positioning characteristics and class distribution.

Therefore, the experiments above presented provide answers to the questions from **Q.1** to **Q.4** of our research (Section 4.1).

Computational performance. We use the simulation depicted by Fig. 6 to measure the computational performance of our method and previous work. We aim to evaluate how training/evaluation times grow when the number of dimensions (features) and the number of training examples grows when evaluated on a set of 50K test samples. The experiments were performed on a single core 1.8GHz of an Intel Core i7-8550U processor and 8GB of RAM, without using GPU processing. All models were trained using Matlab R2018b on Windows 10. To measure training/evaluation time in terms of the number of training examples, we fixed the dimension to 10. As shown in the left chart of Fig. 8, both SVM-based approaches performed better than the other methods. KNN-based methods reached satisfactory results, although some additional time was required to learn the non-linear metrics of our KLMNN. It is worth noting that the state-of-the-art KNFST method performed similarly to KLMNN, in which case 2000 samples were trained in less than a minute. Contrastingly, KPCANOV required substantially higher training times and showed the worst growth pattern. The results of the experiments varying the number of features are shown in the right chart of Fig. 8. Here, we fixed the number of training examples to 800. Notably, all methods showed almost constant training times when the number of dimensions varies. Specifically, in our KLMNN, the number of features only influences kernel computations, which do not substantially contribute to the total training time. Thus, answering the question **Q.5**.

4.3. Experiments on real datasets

We also experimented on four multi-class (more than two classes) datasets with different characteristics. All datasets contain noisy data. Two accessible datasets were chosen from the UCI repository of machine learning databases (Dheeru & Karra Taniskidou, 2017): *iris* and *glass*. The small iris dataset contains three classes of 50 instances each, where each class refers to a type of iris plant, and four attributes represent each instance. Due to its simplicity, we chose this dataset to demonstrate that general multi-class

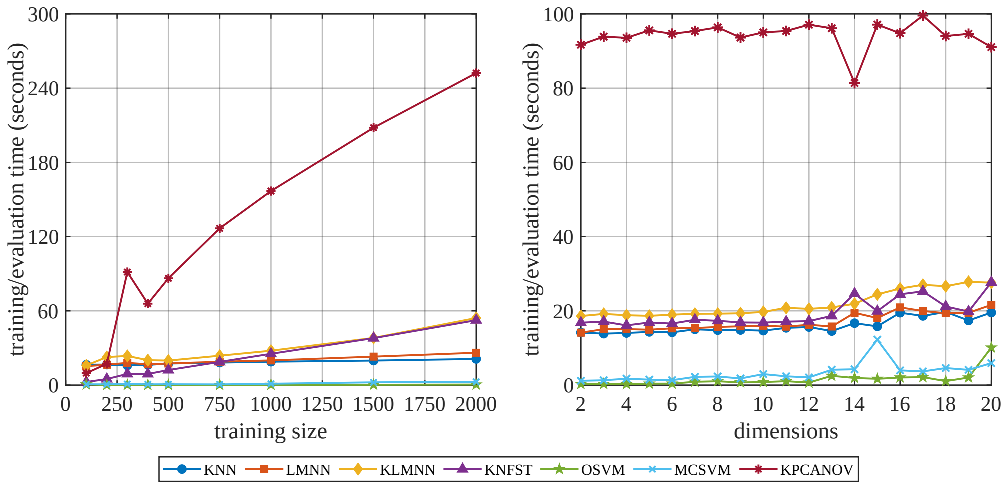


Fig. 8: Computational performance of the methods when the number of training examples grows (left) and when the number of dimensions grows (right). All experiments were evaluated on a test set with 50K samples (40K normal and 10K abnormal).

classifiers are not appropriate to handle data from untrained classes. The unbalanced glass is an extremely noisy dataset composed of instances of 6 types of glass (classes), where the number of instances of each class is, respectively: 70, 76, 17, 13, 9 and 29, as depicted in glass. This dataset was adopted to investigate our method's performance when dealing with low-quality data compared with the methods. The other experimented datasets are related to the gesture recognition problem, which strongly relies on multi-class novelty detection. The first one is a public gesture dataset for body key poses named *KGD* (Miranda et al., 2014a,b). This dataset is composed of instances from 18 body key poses (classes), described by nine joint angles extracted from body skeletons, with approximately 11 instances per class. The second is a novel hand pose dataset representing signs of the Brazilian Sign Language (*libras*), and now made publicly available[1]. The dataset is composed of 20 signs (classes) represented by 14 joint angles of hand skeletons captured from a Leap Motion sensor (Leap Motion, 2018), with approximately 19 instances per class.

In Table 3, we summarize the results for all experimented methods and datasets, when the protocol **P.2** (Section 4.1) and optimal hyperparame-

[1]https://ic.ufal.br/professor/thales/leaplibras/

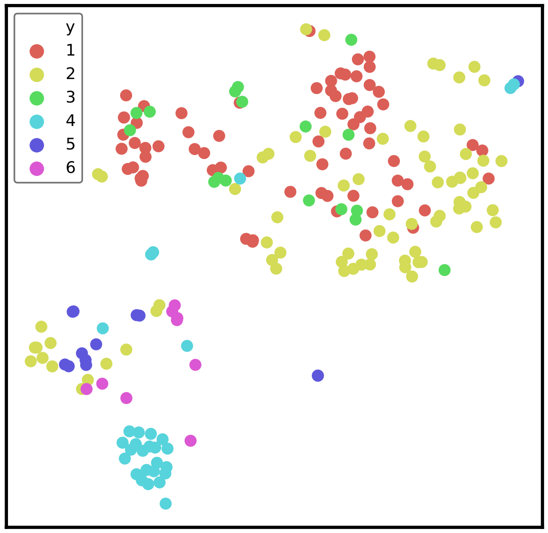


Fig. 9: t-SNE projection of the *glass* dataset: all classes are noisy and/or not well-clustered.

ters are employed on the test sets (for several random selections of normal/abnormal classes). For instance, the K and κ calibration on the KGD dataset are revealed in Table 4 for all KNN-based methods.

Results on the iris dataset corroborate the limitations of MCSVM for novelty detection: by excluding one of the three classes from the training set, MCSVM fails to identify that instances from the excluded class are novelties, achieving the worst MCC result among all compared methods. This result may be visually explained by the sketch already shown in Fig. 1a: since only two classes were used for training, we claim that the resulting MCSVM hyperplane separates the trained classes, but does not define any untrained region as in the illustration shown in Fig. 1b for a distance-based novelty classifier. Our approach outperformed all methods under both metrics, closely followed by its linear variant.

Except for the KGD dataset, the best results were achieved by the metric learning approaches KLMNN and LMNN. When both approaches are directly compared, KLMNN is only outperformed by the LMNN variant on the glass dataset under the F_1 score. This may be a consequence of severe noise found in that dataset, as shown in Fig. 9. In such scenarios, we claim that non-linear metric learning is more susceptible to degenerate the data. Overall, considering the MCC as the reference error metric, our approach outperformed the other methods in three out of the four datasets.

Finally, the study performed in this section provides the answer to the question **Q.6**.

Table 3: Comparison between our non-linear novelty classifier (KLMNN) and other methods on four real datasets. Here, **nc** represents the number of classes of the dataset, and **nuc** the number of excluded classes. The best results are highlighted in bold and the second best results underlined.

dataset	nc	nuc	acc. score	KNN-based methods			other methods			
				KNN	**LMNN**	**KLMNN**	**KNFST**	**OSVM**	**MCSVM**	**KPCANOV**
iris	3	1	F_1	0.94	0.96	**0.96**	0.91	0.94	0.91	0.94
			MCC	0.80	0.88	**0.89**	0.78	0.82	0.76	0.83
glass	6	3	F_1	0.66	**0.74**	0.68	0.65	0.72	0.62	0.73
			MCC	0.35	0.35	**0.39**	0.31	0.22	0.36	0.35
KGD	18	9	F_1	0.94	0.92	0.95	**0.96**	0.85	0.94	0.95
			MCC	0.87	0.84	0.90	**0.92**	0.71	0.89	0.90
libras	20	5	F_1	0.93	0.93	**0.93**	0.91	0.92	0.93	0.93
			MCC	0.69	0.69	**0.70**	0.66	0.67	0.68	0.70

Table 4: MCC of the KNN-based methods for different combinations of hyperparameters K and κ in the *KGD* dataset. The scores for values of κ less than 1 are omitted with symbol $\emptyset$. Best results in bold.

K	**KNN**				**LMNN**				**KLMNN**			
	κ				κ				κ			
	K	$K-1$	$K-2$	$K-3$	K	$K-1$	$K-2$	$K-3$	K	$K-1$	$K-2$	$K-3$
1	**0.87**	$\emptyset$	$\emptyset$	$\emptyset$	**0.84**	$\emptyset$	$\emptyset$	$\emptyset$	0.89	$\emptyset$	$\emptyset$	$\emptyset$
2	0.73	0.80	$\emptyset$	$\emptyset$	0.81	0.81	$\emptyset$	$\emptyset$	0.89	**0.90**	$\emptyset$	$\emptyset$
3	0.51	0.60	0.73	$\emptyset$	0.75	0.82	0.81	$\emptyset$	0.89	0.89	0.89	$\emptyset$
4	0.52	0.52	0.61	0.76	0.75	0.75	0.82	0.82	0.86	0.88	0.88	0.89
5	0.56	0.50	0.53	0.65	0.62	0.64	0.68	0.75	0.79	0.83	0.83	0.83

4.4. Multi-class classification results

We evaluated the non-linear multi-class KNN classifier described in Section 3.3 for a specific classification task. In this case, we compared only to the KNFST and MCSVM methods, since the other OSVM and KPCANOV were not developed for this purpose. To evaluate each method's performance, we calculated the precision, recall, and F_1 score for each class individually.

Then, we calculated the well-known macro and weighted averages of each of these measures. We calculated these measures from the aggregate confusion matrix for experiments on real datasets, summarizing the results of 10 validation experiments.

In Tables 5 and 6, we respectively show comparisons of our approach for multi-class classification on both a synthetic dataset (uniform distributions with 1000 training examples in dimension 10); and a real data set (iris). In both tables, **pre** represent the precision, **rec** is the recall and **f1** is the F_1 score. In addition, **m. avg** is the macro average and **w. avg** is the weighted average measure. Both LMNN and KLMNN variations of our multi-class classifier performed better than the KNFST and MCSVM. Also, it is worth mentioning that learning metrics was important to considerably improved the classification performance of the knn classifier. The KNFST and MCSVM also performed well in this case, reaching average recalls of approximately 1.0, but with average precisions slightly lower. As for the iris dataset, we can see from Table 6 that our approach KLMNN outperformed the compared methods, reaching 0.95 and 0.96 for macro average and weighted average F_1 score, respectively.

Table 5: Multi-class classification results on the uniform distributions from experiment of Fig. 6. Best results for macro average and weighted average F_1 score are in bold and underlined.

	KNN			**LMNN**			**KLMNN**			**KNFST**			**MCSVM**		
	pre	**rec**	**f1**	**pre**	**rec**	**f1**	**pre**	**rec**	**f1**	**pre**	**rec**	**f1**	**pre**	**rec**	**f1**
class 1	0.90	0.94	0.92	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	0.98	0.99	1.00	0.99
class 2	0.91	0.96	0.93	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	0.98	0.98	1.00	0.99
class 3	0.91	0.94	0.93	1.00	1.00	1.00	1.00	1.00	1.00	0.97	1.00	0.98	0.99	1.00	0.99
class 4	0.91	0.93	0.92	1.00	1.00	1.00	1.00	1.00	1.00	0.96	1.00	0.98	0.98	1.00	0.99
m. avg	0.91	0.94	0.93	1.00	1.00	<u>1.00</u>	1.00	1.00	**1.00**	0.97	1.00	0.98	0.98	1.00	0.99
w. avg	0.91	0.94	0.93	1.00	1.00	<u>1.00</u>	1.00	1.00	**1.00**	0.97	1.00	0.98	0.98	1.00	0.99

4.5. Proof-of-concept visual experiment for gestures

We finally performed a simple visual experiment on gestures represented by key pose classes, to demonstrate that gesture recognition can be improved by applying our non-linear novelty filtering step. In Fig. 10, we show a t-SNE projection of instances of key pose classes and poses of a gesture example with its respective dashed trajectory. In Fig. 10a, although the

Table 6: Multi-class classification results on the iris dataset. The first and second best scores for macro average and weighted average F_1 score are highlighted in bold and underline, respectively.

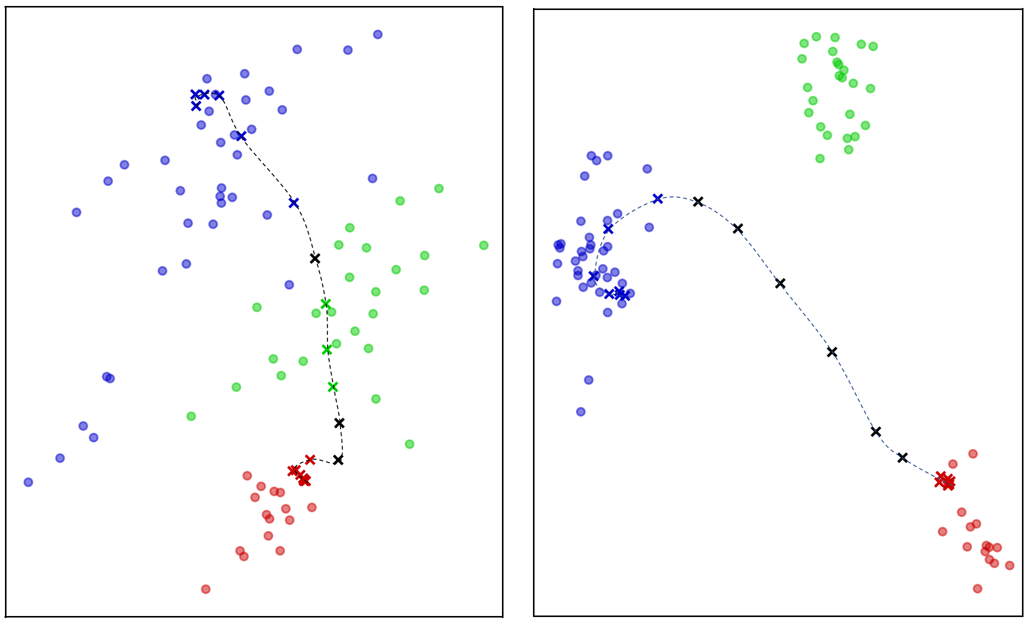
	KNN			**LMNN**			**KLMNN**			**KNFST**			**MCSVM**		
	pre	**rec**	f1	**pre**	**rec**	**f1**	**pre**	**rec**	**f1**	**pre**	**rec**	**f1**	**pre**	**rec**	f1
class 1	0.92	0.99	0.95	0.99	0.96	0.97	0.99	0.99	0.99	0.98	0.98	0.98	0.97	0.95	0.96
class 2	0.83	0.98	0.90	0.89	0.97	0.93	0.88	0.97	0.92	0.81	0.87	0.84	0.83	0.92	0.87
class 3	0.89	0.95	0.92	0.94	0.97	0.95	0.94	0.97	0.95	0.98	0.87	0.92	0.94	0.83	0.88
m. avg	0.88	0.97	0.92	0.94	0.97	<u>0.95</u>	0.93	0.97	**0.95**	0.92	0.90	0.91	0.92	0.90	0.91
w. avg	0.88	0.98	0.93	0.94	0.97	<u>0.95</u>	0.94	0.98	**0.96**	0.93	0.91	0.92	0.92	0.91	0.91

gesture trajectory intersects a region with several green instances, in the high-dimensional feature space, they do not. Thus, the green crosses are the result of incorrect pose classifications that were not detected as novelties. In Fig. 10b, only the correct key poses are detected (red and blue). This gives us evidence that our non-linear novelty filter may improve distance-based gesture recognition methods.

5. Limitations and future work

We proposed a method for multi-class novelty detection that is also capable of performing multi-class classification. The method proposed here is based on a non-linear distance learned from training data and may be applied for novelty, outlier or anomaly detection. Our main conclusions are: 1) multi-class SVMs are not appropriate for novelty detection and filtering in situations where data from abnormal classes are given as input for classification, while our methodology succeeds; 2) in many situations, non-linear distances learning (KLMNN) achieves better results when compared to linear distances learning (LMNN) and the standard Euclidean distance; and 3) our method outperforms previous work on novelty detection in most experiments. In addition, our experiments revealed that our method scales well when both the number of features and/or samples grows when compared to existing novelty detection approaches, both in terms of accuracy and computational performance.

Differently from one-class classification, such as OSVM and KPCANOV, the proposed solution is limited to multi-class novelty detection problems (Bodesheim et al., 2013), relying on training data from at least two normal classes to learn a distance function. Nevertheless, there is still great potential



(a) Euclidean metric: a few poses of the gesture are incorrectly classified as the green class.

(b) Non-linear metric: after transforming the space, the green class cluster is far from the trajectory, and only the correct classes red and blue are detected over the trajectory.

Fig. 10: t-SNE projection of instances of 3 key pose classes (dots), and the trajectory of a gesture (whose poses are represented by crosses) between two key pose classes (red and blue). The color of the crosses encodes pose classifications by our multi-class classifier (red, green or blue), or novelties (black).

for usage in a wide range of applications where only data from a subset of classes are given as input for training, as mentioned in Section 1.

As future work, we plan to apply the proposed solution for applications such as real-time pose and gesture recognition; and visual recognition systems. Experimenting with different kernels for the KPCA projection may also be the topic of future research, as well as more sophisticated hyperparameter tuning strategies.

Acknowledgments

We would like to thank the reviewers for their comments. The authors would like to thank Coordenação de Aperfeiçcoamento de Pessoal de Nível Superior – Brazil (CAPES), National Council for Scientific and Technological Development – Brazil (CNPq) fellowship #301642/2017-6, São Paulo Research Foundation (FAPESP) under grant #2019/23215-9, and Alagoas Research Foundation (FAPEAL) for partially financing this research.

References

Bodesheim, P., Freytag, A., Rodner, E., Kemmler, M., & Denzler, J. (2013). Kernel null space methods for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3374–3381).

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*, 15:1–15:58.

Chatpatanasiri, R., Korsrilabutr, T., Tangchanachaianan, P., & Kijsirikul, B. (2010). A new kernelization framework for mahalanobis distance learning algorithms. *Neurocomputing*, *73*, 1570–1579.

Clifton, L., Clifton, D. A., Watkinson, P. J., & Tarassenko, L. (2011). Identification of patient deterioration in vital-sign data using one-class support vector machines. In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on* (pp. 125–131).

Dheeru, D., & Karra Taniskidou, E. (2017). UCI machine learning repository. URL: `http://archive.ics.uci.edu/ml`.

Diehl, C. P., & Hampshire, J. B. (2002). Real-time object classification and novelty detection for collaborative video surveillance. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on* (pp. 2620–2625). IEEE volume 3.

Faria, E. R., Gonçalves, I. J., Gama, J., & Carvalho, A. C. (2013). Evaluation methodology for multiclass novelty detection algorithms. In *2013 Brazilian Conference on Intelligent Systems* (pp. 19–25). IEEE.

de Faria, E. R., de Leon Ferreira, A. C. P., Gama, J. et al. (2016). Minas: multiclass learning algorithm for novelty detection in data streams. *Data mining and knowledge discovery*, *30*, 640–680.

Hoffmann, H. (2007). Kernel pca for novelty detection. *Pattern recognition*, *40*, 863–874.

Hsu, C.-W., & Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Netw.*, *13*, 415–425.

Ilonen, J., Paalanen, P., Kamarainen, J.-K., & Kalviainen, H. (2006). Gaussian mixture pdf in one-class classification: computing and utilizing confidence values. In *ICPR* (pp. 577–580). IEEE volume 2.

J. Mercer, B. A. (1909). Functions of positive and negative type, and their connection the theory of integral equations. *Phil. Trans. R. Soc. Lond. A*, *209*, 415–446.

Jyothsna, V., Prasad, V. R., & Prasad, K. M. (2011). A review of anomaly based intrusion detection systems. *Int. J. Comput. Appl.*, *28*, 26–35.

Keogh, E., Lin, J., Lee, S.-H., & Van Herle, H. (2007). Finding the most unusual time series subsequence: algorithms and applications. *Knowl. Inf. Syst.*, *11*, 1–27.

Khan, S. S., & Madden, M. G. (2010). A survey of recent trends in one class classification. In *Artificial Intelligence and Cognitive Science* (pp. 188–197). Berlin, Heidelberg: Springer Berlin Heidelberg.

Leap Motion, I. (2018). Leap motion sdk. URL: `https://developer.leapmotion.com/get-started/`.

Liu, J., Lian, Z., Wang, Y., & Xiao, J. (2017). Incremental kernel null space discriminant analysis for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 792–800).

Lv, F., & Nevatia, R. (2007). Single view human action recognition using key pose matching and Viterbi path searching. In *CVPR* (pp. 1–8).

Miranda, L., Vieira, T., Martinez, D., Lewiner, T., Vieira, A. W., & Campos, M. F. M. (2014a). Keypose and Gesture Database. URL: `http://im.ufal.br/professor/thales/gesturedb.html`.

Miranda, L., Vieira, T., Martinez, D., Lewiner, T., Vieira, A. W., & Campos, M. F. M. (2014b). Online gesture recognition from pose kernel learning and decision forests. *Pat. Rec. Letters*, *39*, 65–73.

Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. *Signal Processing*, *99*, 215–249.

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural computation*, *13*, 1443–1471.

Schölkopf, B., Smola, A., & Müller, K.-R. (1997). Kernel principal component analysis. In *International Conference on Artificial Neural Networks* (pp. 583–588). Springer.

Schölkopf, B., & Smola, A. J. (2002). *Learning with Kernels*. MIT.

Sofman, B., Neuman, B., Stentz, A., & Bagnell, J. A. (2011). Anytime online novelty and change detection for mobile robots. *Journal of Field Robotics*, *28*, 589–618.

Van Der Maaten, L. (2014). Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, *15*, 3221–3245.

Weinberger, K. Q., Blitzer, J., & Saul, L. K. (2006). Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems* (pp. 1473–1480).

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, *10*, 207–244.

Zhang, J., & Wang, H. (2006). Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and information systems*, *10*, 333–355.