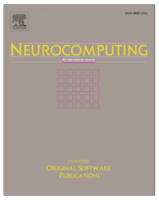
Neurocomputing 425 (2021) 300–322

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges

Juan Luis Suárez *, Salvador García, Francisco Herrera

DaSCI, Andalusian Research Institute in Data Science and Computational Intelligence, University of Granada, Granada, Spain

ARTICLE INFO

Article history:
Received 16 March 2020
Revised 19 August 2020
Accepted 20 August 2020
Available online 11 September 2020
Communicated by Zidong Wang

Keywords:
Distance metric learning
Classification
Mahalanobis distance
Dimensionality
Similarity

ABSTRACT

Distance metric learning is a branch of machine learning that aims to learn distances from the data, which enhances the performance of similarity-based algorithms. This tutorial provides a theoretical background and foundations on this topic and a comprehensive experimental analysis of the most-known algorithms. We start by describing the distance metric learning problem and its main mathematical foundations, divided into three main blocks: convex analysis, matrix analysis and information theory. Then, we will describe a representative set of the most popular distance metric learning methods used in classification. All the algorithms studied in this paper will be evaluated with exhaustive testing in order to analyze their capabilities in standard classification problems, particularly considering dimensionality reduction and kernelization. The results, verified by Bayesian statistical tests, highlight a set of outstanding algorithms. Finally, we will discuss several potential future prospects and challenges in this field. This tutorial will serve as a starting point in the domain of distance metric learning from both a theoretical and practical perspective.

1. Introduction

The use of distances in machine learning has been present since its inception. Distances provide a similarity measure between the data, so that close data will be considered similar, while remote data will be considered dissimilar. One of the most popular examples of similarity-based learning is the well-known nearest neighbors rule for classification, where a new sample is labeled with the majority class within its nearest neighbors in the training set. This classifier was presented in 1967 by Cover and Hart [1], even though this idea had already been mentioned in earlier publications [2,3].

Algorithms in the style of the nearest neighbors classifier are among the main motivators of distance metric learning. These kinds of algorithms have usually used a standard distance, like the Euclidean distance, to measure the data similarity. However, a standard distance may ignore some important properties available in our dataset, and therefore the learning results might be non optimal. The search for a distance that brings similar data as close as possible, while moving non similar data away, can significantly increase the quality of these algorithms.

During the first decade of the 21st century some of the most well-known distance metric learning algorithms were developed, and perhaps the algorithm from Xing et al. [4] is responsible for drawing attention to this concept for the first time. Since then, distance metric learning has established itself as a promising domain in machine learning, with applications in many fields, such as medicine [5,6], security [7,8], social media mining [9,10], speech recognition [11,12], information retrieval [13,14], recommender systems [15,16] and many areas of computer vision, such as person re-identification [17,18], kinship verification [19,20] or image classification [21,22].

Although distance metric learning has proven to be an alternative to consider with small and medium datasets [23], one of its main limitations is the treatment of large data sets, both at the level of number of samples and at the level of number of attributes [24]. In recent years, several alternatives have been explored in order to develop algorithms for these areas [25,26].

Several surveys on distance metric learning have been proposed. Among the well-known surveys we can find the work of Yang and Jin [27], Kulis et al. [28], Bellet et al. [29] and Moutafis et al. [30]. However, we must point out several 'loose ends' present in these studies. On the one hand, they do not provide an in-depth study of the theory behind distance metric learning. Such a study would help to understand the motivation behind the mechanics that give rise to the various problems and tools in this field. On

* Corresponding author.
E-mail addresses: jlsuarezdiaz@ugr.es (J.L. Suárez), salvagl@decsai.ugr.es (S. García), herrera@decsai.ugr.es (F. Herrera).

https://doi.org/10.1016/j.neucom.2020.08.017
0925-2312/© 2020 Elsevier B.V. All rights reserved.

the other hand, previous studies do not carry out enough experimental analyses that evaluate the performance of the algorithms on sufficiently diverse datasets and circumstances.

In this paper we undergo a theoretical study of supervised distance metric learning, in which we show the mathematical foundations of distance metric learning and its algorithms. We analyze several distance metric learning algorithms for classification, from the problems and the objective functions they try to optimize, to the methods that lead to the solutions to these problems. Finally, we carry out several experiments involving all the algorithms analyzed in this study. In our paper, we want to set ourselves apart from previous publications by focusing on a deeper analysis of the main concepts of distance metric learning, trying to get to its basic ideas, as well as providing an experimental framework with the most popular metric learning algorithms. We will also discuss some opportunities for future work in this topic.

Regarding the theoretical background of distance metric learning, we have studied three mathematical fields that are closely related to this topic. The first one is convex analysis [31,32]. Convex analysis is present in many distance metric learning algorithms, as these algorithms they try to optimize convex functions over convex sets. Some interesting properties of convex sets, as well as how to deal with constrained convex problems, will be shown in this study. We will also see how the use of matrices is a fundamental part when modeling our problem. Matrix analysis [33] will therefore be the second field studied. The third is information theory [34], which is also used in some of the algorithms we will present.

As explained before, our work focuses on supervised distance metric learning techniques. A large number of algorithms have been proposed over the years. These algorithms were developed with different purposes and based on different ideas, so that they could be classified into different groups. Thus, we can find algorithms whose main goal is dimensionality reduction [35–37], algorithms specifically oriented to improve distance based classifiers, such as the nearest neighbors classifier [38,39], or the nearest centroid classification [40], and a few techniques that are based on information theory [41–43]. Some of these algorithms also allow kernel versions [44,45,36,42], that allow for the extension of distance metric learning to highly dimensional spaces.

As can be seen in the experiments, we compare all the studied algorithms using up to 34 different datasets. In order to do so, we define different settings to explore their performance and capabilities when considering maximum dimension, centroid-based methods, different kernels and dimensionality reduction. Bayesian statistical tests are used to assess the significant differences among algorithms [46].

In summary, the aims of this tutorial are:

- To know and understand the discipline of distance metric learning and its foundations.
- To gather and study the foundations of the main supervised distance metric learning algorithms.
- To provide experiments to evaluate the performance of the studied algorithms under several case studies and to analyze the results obtained. The code of the algorithms is available in the Python library `pydml` [47].

So as to avoid overloading the paper with a large theoretical content a publicly downloadable theoretical supplement is provided as appendix of the current public draft in ArXiv [48]. In this document Appendix B presents in a rigorous and detailed manner the mathematical foundations of distance metric learning, structured in the three blocks discussed previously, and Appendix C provides a detailed explanation of the algorithms.

Our paper is organized as follows. Section 2 introduces the distance metric problem and its mathematical foundations, explains the family of distances we will work with and shows several examples and applications. Section 3 discusses all the distance metric learning algorithms chosen for this tutorial. Section 4 describes the experiments done to evaluate the performance of the algorithms and shows the obtained results. Finally, Sections 5 and 6 conclude the paper by indicating possible future lines of research in this area and summarizing the work done, respectively. We also provide a glossary of terms at the end of the document (Appendix A) with the acronyms used in the paper.

2. Distance metric learning and mathematical foundations

In this section we will introduce the distance metric learning problem. To begin with, we will provide reasons to support the distance metric learning problem in Section 2.1. Then, we will go over the concept of distance, with special emphasis on the Mahalanobis distances, which will allow us to model our problem (Section 2.2). Once these concepts are defined, in Section 2.3 we will describe the distance metric learning problem, explaining what it consists of, how it is handled in the supervised case and how it is modeled so that it can be treated computationally. To understand the basics of distance metric learning we provide a summary of the mathematical foundations underlying this field in Section 2.4. These foundations support the theoretical description of this discipline as well as the numerous distance metric learning algorithms that will be discussed in Section 3 and [Appendix C] [48]. The mathematical background is then developed extensively in [Appendix B] [48]. Finally, we will finish with Section 2.5 by detailing some of the uses of distance metric learning in machine learning.

2.1. Distance metric learning: why and what for?

Similarity-based learning algorithms are among the earliest used in the field of machine learning. They are inspired by one of the most important components in many human cognitive processes: the ability to detect similarities between different objects. This ability has been adapted to machine learning by designing algorithms that learn from a dataset according to the similarities present between the data. These algorithms are present in most areas of machine learning, such as classification and regression, with the k-Nearest Neighbors (k-NN) rule [1]; in clustering, with the k-means algorithm [49]; in recommender systems, with collaborative approaches based also on nearest neighbors [50]; in semi-supervised learning, to construct the graph representations [51]; in some kernel methods such as the radial basis functions [52], and many more.

To measure the similarity between data, it is necessary to introduce a distance, which allows us to establish a measure whereby it is possible to determine when a pair of samples is more similar than another pair of samples. However, there is an infinite number of distances we can work with, and not all of them will adapt properly to our data. Therefore, the choice of an adequate distance is a crucial element in this type of algorithm.

Distance metric learning arises to meet this need, providing algorithms that are capable of searching for distances that are able to capture features or relationships hidden in our data, which possibly a standard distance, like the Euclidean distance, would not have been able to discover. From another perspective, distance metric learning can also be seen as the missing training step in many similarity-based algorithms, such as the *lazy* nearest-neighbor approaches. The combination of both the distance learning algorithms and the distance-based learners allows us to build more complete learning algorithms with greater capabilities to extract information of interest from our data.

Choosing an appropriate distance learned from the data has proven to be able to greatly improve the results of distance-based algorithms in many of the areas mentioned above. In addition to its potential when adhering to these learners, a good distance allows data to be transformed to facilitate their analysis, with mechanisms such as dimensionality reduction or axes selection, as we will discuss later.

2.2. Mahalanobis distances

We will start by reviewing the concept of distance and some of its properties.

Definition 1. Let X be a non-empty set. A *distance* or *metric* over X is a map $d : X \times X \to \mathbb{R}$ that satisfies the following properties:

1. Coincidence: $d(x,y) = 0 \iff x = y$, for every $x, y \in X$.
2. Symmetry: $d(x,y) = d(y,x)$, for every $x, y \in X$.
3. Triangle inequality: $d(x,z) \leqslant d(x,y) + d(y,z)$, for every $x, y, z \in X$.

The ordered pair (X, d) is called a *metric space*.

The coincidence property stated above will not be of importance to us. That is why we will also consider mappings known as *pseudodistances*, which only require that $d(x,x) = 0$, instead of the coincidence property. In fact, pseudodistances are strongly related with dimensionality reduction, which is an important application of distance metric learning. From now on, when we talk about distances, we will be considering proper distances as well as pseudodistances.

Remark 1. As an immediate consequence of the definition, we have the following additional properties of distances:

4. Non negativity: $d(x,y) \geqslant 0$ for every $x, y \in X$.
5. Reverse triangle inequality: $|d(x,y) - d(y,z)| \leqslant d(x,z)$ for every $x, y, z \in X$.
6. Generalized triangle inequality: $d(x_1, x_n) \leqslant \sum_{i=1}^{n-1} d(x_i, x_{i+1})$ for $x_1, \ldots, x_n \in X$.

When we work in the d-dimensional Euclidean space, a family of distances become very useful in the computing field. These distances are parameterized by positive semidefinite matrices and are known as *Mahalanobis distances*. In what follows, we will refer to $\mathscr{M}_{d' \times d}(\mathbb{R})$ (resp. $\mathscr{M}_d(\mathbb{R})$) as the set of matrices of dimension $d' \times d$ (resp. square matrices of dimension d), and to $S_d(\mathbb{R})_0^+$ as the set of positive semidefinite matrices of dimension d.

Definition 2. Let $d \in \mathbb{N}$ and $M \in S_d(\mathbb{R})_0^+$. The *Mahalanobis distance* corresponding to the matrix M is the map $d_M : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by

$$d_M(x,y) = \sqrt{(x-y)^T M (x-y)}, \quad x, y \in \mathbb{R}^d.$$

Mahalanobis distances come from the (semi-) dot products in $\mathbb{R}^d$ defined by the positive semidefinite matrix M. When M is full-rank, Mahalanobis distances are proper distances. Otherwise, they are pseudodistances. Note that the Euclidean usual distance is a particular example of a Mahalanobis distance, when M is the identity matrix I. Mahalanobis distances have additional properties specific to distances over normed spaces.

7. Homogeneousness: $d(ax, ay) = |a| d(x,y)$, for $a \in \mathbb{R}$, and $x, y \in \mathbb{R}^d$.
8. Translation invariance: $d(x,y) = d(x+z, y+z)$, for $x, y, z \in \mathbb{R}^d$.

Sometimes the term "Mahalanobis distance" is used to describe the squared distances of the form $d_M^2(x,y) = (x-y)^T M (x-y)$. In the area of computing, it is much more efficient to work with d_M^2 rather than with d_M, as this avoids the calculation of square roots. Although d_M^2 is not really a distance, it keeps the most useful properties of d_M from the distance metric learning perspective, as we will see, such as the greater or lesser closeness between different pairs of points. That is why the use of the term "Mahalanobis distance" for both d_M and d_M^2 is quite widespread.

To end this section, we return to the issue of dimensionality reduction that we mentioned when introducing the concept of pseudodistance. When we work with a pseudodistance σ over a set X, it is possible to define an equivalence relationship given by $x \sim y$ if and only if $\sigma(x,y) = 0$, for each $x, y \in X$. Using this relationship we can consider the quotient space $X/_\sim$, and the map $\hat{\sigma} : X/_\sim \times X/_\sim \to \mathbb{R}$ given by $\hat{\sigma}([x],[y]) = \sigma(x,y)$, for each $[x],[y] \in X/_\sim$. This map is well defined and is a distance over the quotient space. When σ is a Mahalanobis distance over $\mathbb{R}^d$, with rank $d' < d$ (we define the rank of a Mahalanobis distance as the rank of the associated positive semidefinite matrix matrix), then the previous quotient space becomes a vector space isomorphic to $\mathbb{R}^{d'}$, and the distance $\hat{\sigma}$ is a full-rank Mahalanobis distance over $\mathbb{R}^{d'}$. That is why, when we have a Mahalanobis pseudodistance on $\mathbb{R}^d$, we can view this as a proper Mahalanobis distance over a lower dimensional space, hence we have obtained a dimensionality reduction.

2.3. Description of distance metric learning

Distance Metric Learning (DML) is a machine learning discipline with the purpose of learning distances from a dataset. In its most general version, a dataset $\mathscr{X} = \{x_1, \ldots, x_N\}$ is available, on which certain similarity measures between different pairs or triplets of data are collected. These similarities are determined by the sets

$$\begin{aligned} S &= \{(x_i, x_j) \in \mathscr{X} \times \mathscr{X} : x_i \text{ and } x_j \text{ are similar.}\}, \\ D &= \{(x_i, x_j) \in \mathscr{X} \times \mathscr{X} : x_i \text{ and } x_j \text{ are not similar.}\}, \\ R &= \{(x_i, x_j, x_l) \in \mathscr{X} \times \mathscr{X} \times \mathscr{X} : x_i \text{ is more similar to } x_j \text{ than to } x_l.\}. \end{aligned}$$

With these data and similarity constraints, the problem to be solved consists in finding, after establishing a family of distances $\mathscr{D}$, those distances that best adapt to the criteria specified by the similarity constraints. To do this, a certain loss function ℓ is set, and the sought-after distances will be those that solve the optimization problem

$$\min_{d \in \mathscr{D}} \ell(d, S, D, R).$$

When we focus on supervised learning, in addition to dataset $\mathscr{X}$ we have a list of labels $y_1, \ldots, y_N$ corresponding to each sample in $\mathscr{X}$. The general formulation of the DML problem is easily adapted to this new situation, just by considering the sets S and D as sets of pairs of same-class samples and different-class samples, respectively. Two main approaches are followed to establish these sets. The *global DML* approach considers the sets S and D to be

$$\begin{aligned} S &= \{(x_i, x_j) \in \mathscr{X} \times \mathscr{X} : y_i = y_j\}, \\ D &= \{(x_i, x_j) \in \mathscr{X} \times \mathscr{X} : y_i \neq y_j\}. \end{aligned}$$

On the other hand, the *local DML* approach replaces the previous definition of S with

$$S = \{(x_i, x_j) \in \mathscr{X} \times \mathscr{X} : y_i = y_j \text{ and } x_j \in \mathscr{U}(x_i)\},$$

where $\mathscr{U}(x_i)$ denotes a *neighborhood* of x_i, that is, a set of points that should be close to x_i, which has to be established before the learning process by using some sort of prior information, or a standard similarity measure. The set D remains the same in the local approach, since different-class samples are not meant to be similar in a supervised learning setting. In addition, the set R may be also available in both approaches by defining triplets (x_i, x_j, x_l) where in general $y_i = y_j \neq y_l$, and they verify certain conditions imposed on the distance between x_i and x_j, as opposed to the distance between x_i and x_l. This is the case, for example, for impostors in the LMNN algorithm (see Section 3.2.1 and [38]). In any case, labels have all the necessary information in the field of supervised DML. From now on we will focus on this kind of problem.

Furthermore, focusing on the nature of the dataset, practically all of the DML theory is developed for numerical data. Although it is possible to define relevant distances for non-numerical attributes [53,54] and although some learning processes can be performed with them [55,56], the richness of the distances available to numerical features, their ability to be parameterized computationally, and the fact that nominal data can be converted to numerical variables or ordinal variables, with appropriate encoding [57], cause the relevant distances in this discipline to be those defined for numerical data. For this reason, from now on, we will focus on supervised learning problems with numerical datasets.

We will suppose then that $\mathscr{X} \subset \mathbb{R}^d$. As we saw in the previous section, for finite-dimensional vector spaces we have the family of Mahalanobis distances, $\mathscr{D} = \{d_M : M \in S_d(\mathbb{R})_0^+\}$. With this family, we have at our disposal all the distances associated with dot products in $\mathbb{R}^d$ (and in lower dimensions). In addition, this family is determined by the set of positive semidefinite matrices, and therefore, we can use these matrices, which we will call *metric matrices*, to parameterize distances. In this way, the general problem adapted to supervised learning with Mahalanobis distances can be rewritten as

$$\min_{M \in S_d(\mathbb{R})_0^+} \ell(d_M, (x_1, y_1), \dots, (x_N, y_N)).$$

However, this is not the only way to parameterize this type of problem. We know, from the *matrix decomposition theorem* discussed in Section 2.4 and [Theorem 5] [48], that if $M \in S_d(\mathbb{R})_0^+$, then there is a matrix $L \in \mathscr{M}_d(\mathbb{R})$ so that $M = L^T L$, and this matrix is unique except for an isometry. So, then we get

$$\begin{aligned} d_M^2(x,y) &= (x-y)^T M (x-y) = (x-y)^T L^T L (x-y) \\ &= (L(x-y))^T (L(x-y)) = \|L(x-y)\|_2^2. \end{aligned}$$

Therefore, we can also parameterize Mahalanobis distances through any matrix, although in this case the interpretation is different. When we learn distances through positive semidefinite matrices we are learning a new metric over $\mathbb{R}^d$. When we learn distances with the previous L matrices, we are learning a linear map (given by $x \mapsto Lx$) that transforms the data in the space, and the corresponding distance is the usual Euclidean distance after projecting the data onto the new space using the linear map. Both approaches are equivalent thanks to the matrix decomposition theorem [Theorem 5] [48].

In relation to dimensionality, it is important to note that, when the learned metric M is not full-rank, we are actually learning a distance over a space of lower dimension (as we mentioned in the previous section), which allows us to reduce the dimensionality of our dataset. The same occurs when we learn linear maps that are not full-rank. We can extend this case and opt to learn directly linear maps defined by $L \in \mathscr{M}_{d' \times d}(\mathbb{R})$, with $d' < d$. In this way, we ensure that data are directly projected into a space of dimension no greater than d'.

Both learning the metric matrix M and learning the linear transformation L, are useful approaches to model DML problems, each one with its advantages and disadvantages. For example, parameterizations via M usually lead to convex optimization problems. In contrast, convexity in problems parameterized by L is not so easy to achieve. On the other hand, parameterizations using L make it possible to learn projections directly onto lower dimensional spaces, while dimensional constraints for problems parameterized by M are not so easy to achieve. Let us examine these differences with simple examples.

Example 1. Many of the functions we will want to optimize will depend on the squared distance defined by the metric M or by the linear transformation L, that is, either they will have terms of the form $\|v\|_M^2 = v^T M v$, or of the form $\|v\|_L^2 = \|Lv\|_2^2$. Both the maps $M \mapsto \|v\|_M^2$ and $L \mapsto \|v\|_L^2$ are convex (the first is actually affine). However, if we want to substract terms in this way, we lose convexity in L, because the mapping $L \mapsto -\|v\|_L^2$ is no longer convex. In contrast, the mapping $M \mapsto -\|v\|_M^2$ is still affine and, therefore, convex.

Example 2. Rank constraints are not convex, and therefore we may not dispose of a projection onto the set corresponding to those constraints, unless we learn the mapping (parameterized by L) directly to the space with the desired dimension, as explained before. For example, if we consider the set $C = \{M \in S_2(\mathbb{R})_0^+ : r(A) \leqslant 1\}$, we get $A = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \in C$ and $B = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix} \in C$. However, $(1-\lambda)A + \lambda B = I \notin C$, for $\lambda = 1/2$.

2.4. Mathematical foundations of distance metric learning

There are three main mathematical areas that support DML: convex analysis, matrix analysis and information theory. The first provides the necessary tools so that many of the algorithms can address their optimization problems. Thanks to the second we can parameterize DML, and in this way we can compute the different problems. It also provides us with some interesting results when it comes to solving certain problems related to dimensionality reduction. Finally, the third field provides us with concepts and tools that are very useful for designing algorithms that use probability distributions associated with the data.

2.4.1. Convex analysis

Let us begin with convex analysis. One of the properties of convex sets that makes convex analysis of great interest in DML is known as the *convex projection theorem* [Theorem 2] [48], which ensures that for any non-empty convex closed set K in $\mathbb{R}^d$ and for every point $x \in \mathbb{R}^d$ there is a single point $x_0 \in K$ for which the distance from x to K is the same to the distance from x to x_0. That is, the distance from x to K is materialized in the point x_0, which is called the *projection of x to K*.

The existence of a projection mapping onto any closed and convex set in $\mathbb{R}^d$ is fundamental when optimizing convex functions with convex constraints, which are frequent, in particular, in many DML algorithms. Let us first discuss optimization mechanisms when working with unconstrained differentiable functions, which, although they do not strictly take part in convex analysis, are also present in some DML algorithms and are the basis for convex optimization mechanisms. In these cases, the most popular techniques are the well-known *gradient descent methods*, which are iterative methods. The basic idea of gradient descent methods is to move in the direction of the gradient of the objective function in order to optimize it. We show in [Appendix B.1.2] [48] that, indeed, small displacements in the gradient direction guarantee the improvement of the objective function, proving the effectiveness of these methods.

Returning to the constrained case, we can see that gradient descent methods are no longer valid, since the displacement in the gradient direction can no longer fulfill the constraints. However, we will show that, if after the gradient step we project the obtained point onto the convex set determined by the constraints, the combination of both movements contributes to improving the value of the objective function as long as the initial gradient step is small enough. This extension of gradient descent to the constrained convex case is known as the *projected gradient method*. There are also other approaches, such as the penalty methods, which allow these problems to be handled by transforming the constrained objective function into a new unconstrained objective function in which the violations of the previous constraints are converted into penalties that worsen the value of the new objective function [58]. They will not usually, however, be preferred in the matrix problems we will be dealing with, as they may be computationally expensive and difficult to adapt [59].

Finally, we must highlight other tools of interest for the optimization problems to be studied. Firstly, when working with convex problems with multiple constraints, the projections on each individual constraint are often known, but the projection onto the set determined by all the constraints is not. With the method known as the *iterated projections method* [Appendix B.1.2] [48] we can approach this projection by subsequently projecting onto each of the individual constraints until convergence (which is guaranteed) is obtained. Lastly, convex functions that are not differentiable everywhere can still be optimized following the approaches discussed here, as they admit sub-gradients at every point. Sub-gradient descent methods [Appendix B.1.2] [48] can work in the same way as gradient descent methods and can therefore be applied with convex functions that may not be differentiable at some points.

2.4.2. Matrix analysis

As we have already seen, matrices are a key element in DML. There are several results that are essential for the development of the DML theory and its algorithms. The first of these is the *matrix decomposition theorem* [Theorem 5] [48], which was already mentioned in Section 2.3. This theorem states that for any positive matrix $M \in S_d(\mathbb{R})_0^+$ there is a matrix $L \in \mathscr{M}_d(\mathbb{R})$ so that $M = L^TL$ and L is unique except for an isometry. This result allows us to approach DML from the two perspectives (learning the metric M or learning the linear map L) already discussed in Section 2.3.

An important aspect when designing DML algorithms is the geometric manipulation of the matrices (for learning both M and L). Observe that to be able to talk about the convex analysis concepts discussed previously over the set of matrices, we first need to establish an inner product over them. The *Frobenius inner product* allows us to identify matrices as vectors where we add the matrix rows one after the other, and then compute the usual vectorial inner product with these vectors. With the Frobenius product we convert the matrices set in a Hilbert space, and therefore can apply the convex analysis theory studied in the previous section.

Staying on this subject, we have to highlight a case study of particular interest. We will see many situations where we want to optimize a convex function defined on a matrix space, with the restriction that the variable is positive semidefinite. These optimization problems are convex and are usually called *semidefinite programming* problems. We can optimize these objective functions using the projected gradient descent method. The *semidefinite projection theorem* [Theorem 4] [48] states that we can compute the projection of a matrix onto the positive semidefinite cone by performing an eigenvalue decomposition, nullifying the negative eigenvalues and recomposing the matrix with the new eigenvalues. Therefore, we know how to project onto the constraint set and consequently we can apply the projected gradient method.

Finally, we will see that certain algorithms, especially those associated with dimensionality reduction, use optimization problems with similar structures. These problems involve one or more symmetric matrices and the objective function is obtained as a trace after performing certain operations with these matrices. This is, for instance, the case of the objective function of the well-known *principal components analysis*, which can be written as

$$\begin{aligned} \max_{L \in \mathscr{M}_{d' \times d}(\mathbb{R})} \quad & \operatorname{tr}\left(LAL^T\right) \\ \text{s.t. :} \quad & LL^T = I, \end{aligned}$$

where A is a symmetric matrix of dimension d. These problems have the property that they can be optimized without using gradient methods, since an optimum can be built by taking the eigenvectors associated with the largest eigenvalues of the matrices involved in the problem (in the case described here, the eigenvectors of A). In [Appendix B.2.3] [48] we present these problems in more detail and show how their solution is obtained.

2.4.3. Information theory

Information theory is a branch of mathematics and computer theory with the purpose of establishing a rigorous measure to quantify the information and disorder found in a communication message. It has been applied in most science and engineering fields. In DML, information theory is used in several algorithms to measure the closeness between probability distributions. Then, these algorithms try to find a distance metric for which these probability distributions are as close as possible or as far as possible, depending on what distributions are defined. The measures used in this area, unlike distances, only require the properties of non-negativity and coincidence, and are called *divergences*. We will use two different divergences throughout this study:

- The *relative entropy* or the *Kullback–Leibler divergence*, defined for probability distributions p and q, and X the random variable corresponding to p as

$$\mathrm{KL}(p\|q) = \mathbb{E}_p\left[\log \frac{p(X)}{q(X)}\right].$$

- The *Jeffrey divergence* or the *symmetric relative entropy*, defined for p, q and X in the same conditions as above, as

$$\mathrm{JF}(p\|q) = \mathrm{KL}(p\|q) + \mathrm{KL}(q\|p).$$

The key fact that makes divergences very useful in DML is that, when the distributions involved are multivariate gaussian, these divergences can be expressed in terms of matrix divergences, which give rise to problems that can be dealt with quite effectively using the tools described in this section. In [Appendix B.3] [48] we present the matrix expressions obtained for the Kullback–Leibler and the Jeffrey divergences for the most remarkable cases.

2.5. Use cases in machine learning

This section describes some of the most prominent uses of DML in machine learning, illustrated with several examples.

- **Improve the performance of distance-based classifiers.** This is one of the main purposes of DML. Using such learning, a distance that fits well with the dataset and the classifier can be found, improving the performance of the classifier [38,39]. An example is shown in Fig. 1.
- **Dimensionality reduction.** As we have already commented, learning a low-rank metric implies a dimensionality reduction on the dataset we are working with. This dimensionality reduction provides numerous advantages, such as a reduction in the computational cost, both in space and time, of the algorithms that will be used later, or the removal of the possible noise introduced when picking up the data. In addition, some distance-based classifiers are exposed to a problem called *curse of dimensionality* (see, for example, [60], Section 19.2.2). By reducing the dimension of the dataset, this problem also becomes less serious. Finally, if deemed necessary, projections onto dimension 1, 2 and 3 would allow us to obtain visual representations of our data, as shown in Fig. 2. In general, many real-world problems arise with a high dimensionality, and need a dimensionality reduction to be handled properly.

维度灾难
维度诅咒

- **Axes selection and data rearrangement.** Closely related to dimensionality reduction, this application is a result of algorithms that learn transformations which allow the coordinate axes to be moved (or selected according to the dimension), so that in the new coordinate system the vectors concentrate certain measures of information on their first components [61]. An example is shown in Fig. 3.
- **Improve the performance of clustering algorithms.** Many of the clustering algorithms use a distance to measure the closeness between data, and thus establish the clusters so that data in the same cluster are considered close for that distance. Sometimes, although we do not know the ideal groupings of the data or the number of clusters to establish, we can know that certain pairs of points must be in the same cluster and that other specific pairs must be in different clusters [4]. This happens in numerous problems, for example, when clustering web documents [62]. These documents have a lot of additional information, such as links between documents, which can be included as similarity constraints. Many clustering algorithms are particularly sensitive to the distance used, although many also depend heavily on the parameters with which they are initialized [63,64]. It is therefore important to seek a balance or an appropriate aggregation between these two components. In any case, the parameter initialization is beyond the scope of this paper.
- **Semi-supervised learning.** Semi-supervised learning is a learning model in which there is one set of labeled data and another set (generally much larger) of unlabeled data. Both datasets are intended to learn a model that allows new data to be labeled. Semi-supervised learning arises from the fact that in many situations collecting unlabeled data is relatively straightforward, but assigning labels can require a supervisor to assign them manually, which may not be feasible. In contrast, when a lot of unlabeled data is used along with a small amount of labeled data, it is possible to improve learning outcomes considerably, as exemplified in Fig. 4. Many of these techniques consist of constructing a graph with weighted edges from the data, where the value of the edges depends on the distances between the data. From this graph we try to infer the labels of the whole dataset, using different propagation algorithms [51]. In the construction of the graph, the choice of a suitable distance is important, thus DML comes into play [65].

From the applications we have seen, we can conclude that DML can be viewed as a preprocessing step for many distance-based learning algorithms. The algorithms analyzed in our work focus on the first three applications of the above enumeration. It should be noted that, although the fields above are those where DML has traditionally been used, today, new prospects and challenges for DML are being considered. They will be discussed in Section 5.

3. Algorithms for distance metric learning

This section introduces some of the most popular techniques currently being used in supervised DML. Due to space issues, the section will give a brief description of each of the algorithms, while a detailed description can be found in [Appendix C] [48], where the problems the algorithms try to optimize are analyzed, together with their mathematical foundations and the techniques used to solve them.

Table 1 shows the algorithms studied throughout this work, including name, references and a short description. These algorithms will be empirically analyzed in the next section. This study is not intended to be exhaustive and therefore only some of the most popular algorithms have been selected for the theoretical study and the subsequent experimental analysis.

We will now provide a brief introduction to these algorithms. According to the main purpose of each algorithm, we can group them into different categories: dimensionality reduction techniques (Section 3.1), algorithms directed at improving the nearest neighbors classifiers (Section 3.2), algorithms directed at improving the nearest centroid classifiers (Section 3.3), or algorithms based on information theory (Section 3.4). These categories are not necessarily exclusive, but we have considered each of the algorithms in the category associated with their dominant purpose. We also introduce, in Section 3.5 several algorithms with less specific goals, and finally, in Section 3.6, the kernel versions for some of the algorithms studied.

We will explain the problems that each of these techniques try to solve. For a more detailed description of each algorithm, the reader can refer to the corresponding section of the appendix supplement [48], as shown in Table 1.

3.1. Dimensionality reduction techniques

Dimensionality reduction techniques try to learn a distance by searching a linear transformation from the dataset space to a lower dimensional Euclidean space. We will describe the algorithms PCA [66], LDA [35] and ANMM [36].

3.1.1. Principal Components Analysis (PCA)

PCA [66] is one of the most popular dimensionality reduction techniques in unsupervised DML. Although PCA is an unsupervised learning algorithm, it is necessary to talk about it in our paper, firstly because of its great relevance, and more particularly, because when a supervised DML algorithm does not allow a dimensionality reduction, PCA can be first applied to the data in order to be able to use the algorithm later in the lower dimensional space.

The purpose of PCA is to learn a linear transformation from the original space $\mathbb{R}^d$ to a lower dimensional space $\mathbb{R}^{d'}$ for which the loss when recomposing the data in the original space is minimized. This has been proven to be equivalent to iteratively finding orthogonal directions for which the projected variance of the dataset is maximized. The linear transformation is then the projection onto these directions. The optimization problem can be formulated as

$$\max_{\substack{L \in \mathscr{M}_{d' \times d}(\mathbb{R}) \\ LL^T = I}} \operatorname{tr}\left(L\Sigma L^T\right),$$

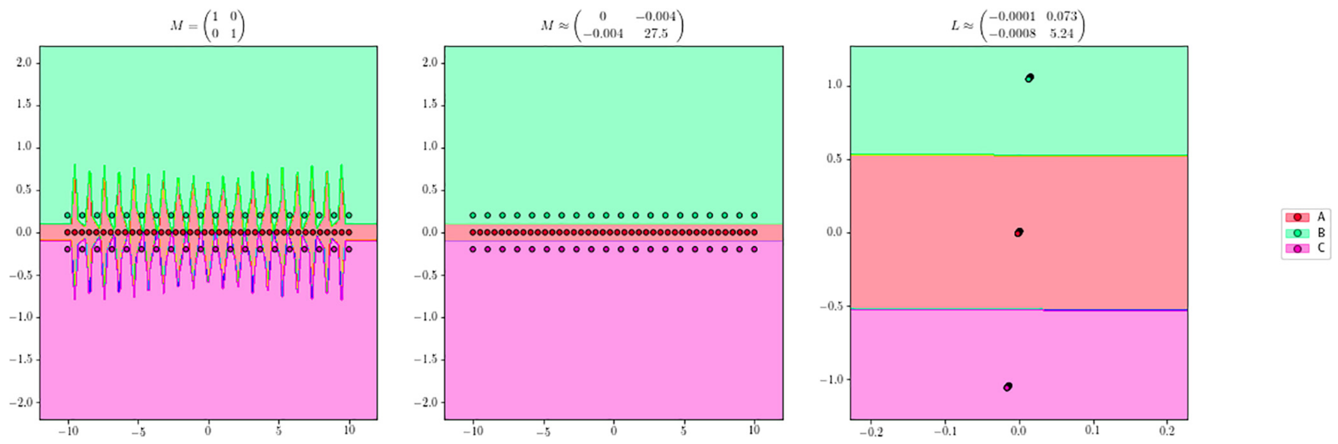


Fig. 1. Suppose we have a dataset in the plane, where data can belong to three different classes, whose regions are defined by parallel lines. Suppose that we want to classify new samples using the one nearest neighbor classifier. If we use Euclidean distance, we would obtain the classification regions shown in the image on the left, because there is a greater separation between each sample in class B and class C than there is between the regions. However, if we learn an adequate distance and try to classify with the nearest neighbor classifier again, we obtain much more effective classification regions, as shown in the center image. Finally, as we have seen, learning a metric is equivalent to learning a linear map and to use Euclidean distance in the transformed space. This is shown in the right figure. We can also observe that data are being projected, except for precision errors, onto a line, thus we are also reducing the dimensionality of the dataset.

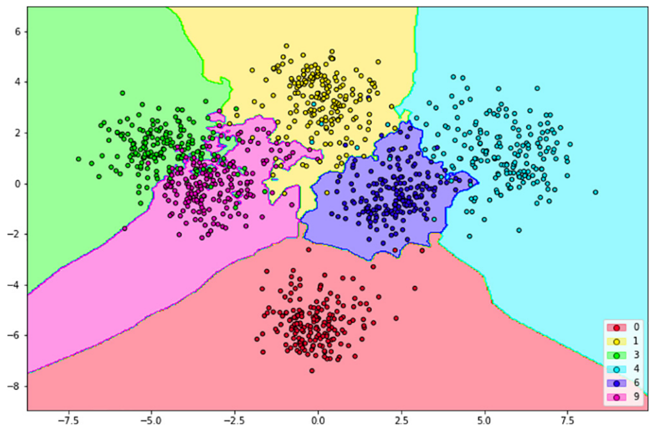


Fig. 2. 'Digits' dataset consists of 1797 examples. Each of them consists of a vector with 64 attributes, representing intensity values on an 8x8 image. The examples belong to 10 different classes, each of them representing the numbers from 0 to 9. By learning an appropriate transformation we are able to project most of the classes on the plane, so that we can clearly perceive the differentiated regions associated with each of the classes.

where Σ is, except for a constant, the covariance matrix of $\mathscr{X}$, and tr is the trace operator. The solution to this problem can be obtained by taking as the rows of L the eigenvectors of Σ associated with its largest eigenvalues.

3.1.2. Linear Discriminant Analysis (LDA)

LDA [35] is a classical DML technique with the purpose of learning a projection matrix that maximizes the separation between classes in the projected space using *within-class* and *between-class* variances. It follows a scheme similar to the one proposed by PCA, but in this case it takes the supervised information provided by the labels into account.

The optimization problem of LDA is formulated as

$$\max_{L \in \mathscr{M}_{d' \times d}(\mathbb{R})} \quad \operatorname{tr}\left(\left(LS_wL^T\right)^{-1} \left(LS_bL^T\right) \right),$$

where S_b and S_w are, respectively, the between-class and within-class *scatter matrices*, which are defined as

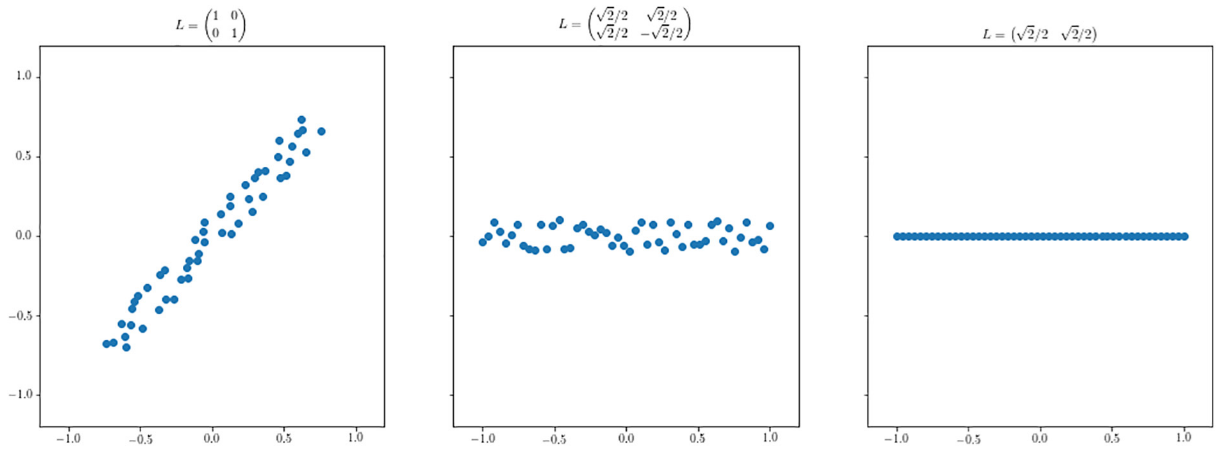


Fig. 3. The dataset in the left figure seems to concentrate most of its information on the diagonal line that links the lower left and upper right corners. By learning an appropriate transformation, we can get that direction to fall on the horizontal axis, as shown in the center image. As a result, the first coordinate of the vectors in this new basis concentrates a large part of the variability of the vector. In addition, it seems reasonable to think that the values introduced by the vertical coordinate might be due to noise, and so, we can even just keep the first component, as shown in the right image.

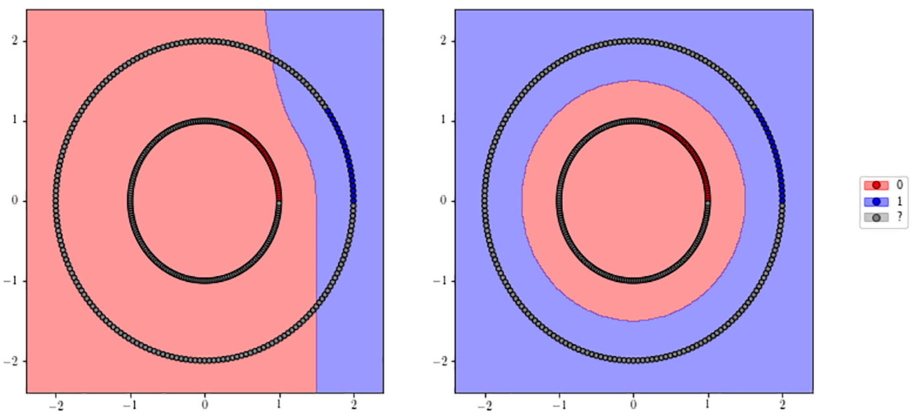


Fig. 4. Learning with only supervised information (left) versus learning with all unsupervised information (right).

Table 1
Description of the DML algorithms analyzed in this study.

Name	References	Appendix section [48]	Description
PCA	Jolliffe [66]	C.1.1	A dimensionality reduction technique that obtains directions maximizing variance. Although not supervised, it is important to allow dimensionality reduction in other algorithms that are not able to do so on their own.
LDA	Fisher [35]	C.1.2	A dimensionality reduction technique that obtains direction maximizing a ratio involving *between-class* variances and *within-class* variances.
ANMM	Wang and Zhang [36]	C.1.3	A dimensionality reduction technique that aims at optimizing an average neighborhood margin between same-class neighbors and different-class neighbors, for each of the samples.
LMNN	Weinberger and Saul [38]	C.2.1	An algorithm aimed at improving the accuracy of the *k*-neighbors classifier. It tries to optimize a two-term error function that penalizes, on the one hand, large distance between each sample and its *target neighbors*, and on the other hand, small distances between different-class samples.
NCA	Goldberger et al. [39]	C.2.2	An algorithm aimed at improving the accuracy of the *k*-neighbors classifier, by optimizing the expected *leave-one-out* accuracy for the nearest neighbor classification.
NCMML	Mensink et al. [40]	C.3.1	An algorithm aimed at improving the accuracy of the *nearest class mean* classifier, by optimizing a log-likelihood for the labeled data in the training set.
NCMC	Mensink et al. [40]	C.3.2	A generalization of the NCMML algorithm aimed at improving nearest centroids classifiers that allow multiple centroids per class.
ITML	Davis et al. [41]	C.4.1	An information theory based technique that aims at minimizing the Kullback–Leibler divergence with respect to an initial gaussian distribution, but while keeping certain similarity constraints between data.
DMLMJ	Nguyen et al. [42]	C.4.2	An information theory based technique that aims at maximizing the Jeffrey divergence between two distributions, associated to similar and dissimilar points, respectively.
MCML	Globerson and Roweis [43]	C.4.3	An information theory based technique that tries to collapse same-class points in a single point, as far as possible from the other classes collapsing points.
LSI	Xing et al. [4]	C.5.1	A DML algorithm that globally minimizes the distances between same-class points, while fulfilling minimum-distance constraints for different-class points.
DML-eig	Ying and Li [67]	C.5.2	A DML algorithm similar to LSI that offers a different resolution method based on eigenvalue optimization.
LDML	Guillaumin et al. [68]	C.5.3	A probabilistic approach for DML based on the logistic function.
KLMNN	Weinberger and Saul [38]; Torresani and Lee [44]	C.6.1	The kernel version of LMNN.
KANMM	Wang and Zhang [36]	C.6.2	The kernel version of ANMM.
KDMLMJ	Nguyen et al. [42]	C.6.3	The kernel version of DMLMJ.
KDA	Mika et al. [45]	C.6.4	The kernel version of LDA.

$$\begin{aligned} S_b &= \sum_{c\in\mathscr{C}} N_c(\mu_c-\mu)(\mu_c-\mu)^T, \\ S_w &= \sum_{c\in\mathscr{C}}\sum_{i\in\mathscr{C}_c}(x_i-\mu_c)(x_i-\mu_c)^T, \end{aligned}$$

where $\mathscr{C}$ is the set of all the labels, $\mathscr{C}_c$ is the set of indices i for which $y_i = c \in \mathscr{C}, N_c$ is the number of samples in $\mathscr{X}$ with class c, μ_c is the mean of the training samples in class c and μ is the mean of the whole training set. The solution to this problem can be found by taking the eigenvectors of $S_w^{-1}S_b$ associated with its largest eigenvalues to be the rows of L.

3.1.3. Average Neighborhood Margin Maximization (ANMM)

ANMM [36] is another DML technique specifically oriented to dimensionality reduction that tries to solve some of the limitations of PCA and LDA.

The objective of ANMM is to learn a linear transformation $L \in \mathscr{M}_{d'\times d}(\mathbb{R})$, with $d' \leqslant d$ that maximizes an *average neighborhood margin* defined, for each sample, by the difference between its average distance to its nearest neighbors of different class and the average distance to its nearest neighbors of same class.

If we consider the set $\mathscr{N}_i^o$ of the ξ samples in $\mathscr{X}$ nearest to x_i and with the same class as x_i, and the set $\mathscr{N}_i^e$ of the ζ samples in $\mathscr{X}$ nearest to x_i and with a different class to x_i, we can express the global average neighborhood margin, for the distance defined by L, as

$$\gamma^L = \sum_{i=1}^{N}\left(\sum_{k:x_k\in\mathscr{N}_i^e}\frac{\|Lx_i-Lx_k\|^2}{|\mathscr{N}_i^e|} - \sum_{j:x_j\in\mathscr{N}_i^o}\frac{\|Lx_i-Lx_j\|^2}{|\mathscr{N}_i^o|}\right).$$

In this expression, each summand is associated with each sample x_i in the training set, and the positive term inside each summand represents the average distance to its ζ nearest neighbors of different classes, while the negative term represents the average distance to its ξ nearest neighbors of the same class. Therefore, these differences constitute the average neighborhood margins for each sample x_i. The global margin γ^L can be expressed in terms of a scatterness matrix containing the information related to different-class neighbors, and a compactness matrix that stores the information corresponding to the same-class neighbors, that is,

$$\gamma^L = \mathrm{tr}(L(S-C)L^T),$$

where S and C are, respectively, the *scatterness* and *compactness* matrices, defined as

$$\begin{aligned} S &= \sum_i \sum_{k:x_k \in \mathscr{N}_i^e} \frac{(x_i-x_k)(x_i-x_k)^T}{|\mathscr{N}_i^e|} \\ C &= \sum_i \sum_{j:x_j \in \mathscr{N}_i^o} \frac{(x_i-x_j)(x_i-x_j)^T}{|\mathscr{N}_i^o|}. \end{aligned}$$

If we impose the scaling restriction $LL^T = I$ (scaling would increase the average neighborhood margin indefinitely), the average neighborhood margin can be maximized by taking the eigenvectors of $S-C$ associated with its largest eigenvalues to be the rows of L.

3.2. Algorithms to improve nearest neighbors classifiers

One of the main applications of DML is to improve other distance based learning algorithms. Since the nearest neighbors classifier is one of the most popular distance based classifiers many DML algorithms are designed to improve this classifier, as is the case with LMNN [38] and NCA [39].

3.2.1. Large Margin Nearest Neighbors (LMNN)

LMNN [38] is a DML algorithm aimed specifically at improving the accuracy of the k-nearest neighbors classifier.

LMNN tries to bring each sample as close as possible to its *target neighbors*, which are k pre-selected same-class samples requested to become the nearest neighbors of the sample, while trying to prevent samples from other classes from invading a margin defined by those target neighbors. This setup allows the algorithm to locally separate the classes in an optimal way for k-neighbors classification.

Assuming the sets of target neighbors are chosen (usually they are taken as the nearest neighbors for Euclidean distance), the error function that LMNN minimizes is a two-term function. The first term is the target neighbors pulling term, given by

$$\varepsilon_{pull}(M) = \sum_{i=1}^N \sum_{j \rightsquigarrow i} d_M(x_i, x_j)^2,$$

where d_M is the Mahalanobis distance corresponding to $M \in S_d(\mathbb{R})_0^+$ and $j \rightsquigarrow i$ iff x_j is a target neighbor of x_i. The second term is the *impostors* pushing term, given by

$$\varepsilon_{push}(M) = \sum_{i=1}^N \sum_{j \rightsquigarrow i} \sum_{l=1}^N (1-y_{il})[1 + d_M(x_i,x_j)^2 - d_M(x_i,x_l)^2]_+,$$

where $y_{il} = 1$ if $y_i = y_l$ and 0 otherwise, and $[\cdot]_+$ is defined as $[z]_+ = \max\{z, 0\}$. Finally, the objective function is given by

$$\varepsilon(M) = (1-\mu)\varepsilon_{pull}(M) + \mu\varepsilon_{push}(M), \quad \mu \in]0,1[.$$

This function can be optimized using semidefinite programming. It is possible to optimize this function in terms of L, using gradient methods, as well. By optimizing in terms of M we gain convexity in the problem, while by optimizing in terms of L we can use the algorithm to force a dimensionality reduction.

3.2.2. Neighborhood Components Analysis (NCA)

NCA [39] is another DML algorithm aimed specifically at improving the accuracy of the nearest neighbors classifiers. It is designed is to learn a linear transformation with the goal of minimizing the leave-one-out error expected by the nearest neighbor classification.

To do this, we define the probability that a sample $x_i \in \mathscr{X}$ has $x_j \in \mathscr{X}$ as its nearest neighbor for the distance defined by $L \in \mathscr{M}_d(\mathbb{R}), p_{ij}^L$, as the softmax

$$p_{ij}^L = \frac{\exp\left(-\|Lx_i - Lx_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|Lx_i - Lx_k\|^2\right)} \quad (j \neq i), \qquad p_{ii}^L = 0.$$

The expected number of correctly classified samples according to this probability is obtained as

$$f(L) = \sum_{i=1}^N \sum_{j \in C_i} p_{ij}^L,$$

where C_i is the set of indices j so that $y_j = y_i$. The function f can be maximized using gradient methods, and the distance resulting from this optimization is the one that minimizes the expected leave-one-out error, and therefore, the one that NCA learns.

3.3. Algorithms to improve nearest centroids classifiers

Apart from the nearest neighbors classifiers, other distance-based classifiers of interest are the so-called nearest centroid classifiers. These classifiers obtain a set of centroids for each class and classify a new sample by considering the nearest centroids to the sample. There are also DML algorithms designed for these classifiers, as is the case for NCMML and NCMC [40].

3.3.1. Nearest Class Mean Metric Learning (NCMML)

NCMML [40] is a DML algorithm specifically designed to improve the Nearest Class Mean (NCM) classifier. To do this, it uses a probabilistic approach similar to that used by NCA to improve the accuracy of the nearest neighbors classifier.

In this case, we define the probability that a sample $x_i \in \mathscr{X}$ will be labeled with the class c, according to the nearest class mean criterion, for the distance defined by $L \in \mathscr{M}_{d' \times d}(\mathbb{R})$, as

$$p_L(c|x) = \frac{\exp\left(-\frac{1}{2}\|L(x-\mu_c)\|^2\right)}{\sum_{c' \in \mathscr{C}} \exp\left(-\frac{1}{2}\|L(x-\mu_{c'})\|^2\right)},$$

where $\mathscr{C}$ is the set of all the classes and μ_c is the mean of the training samples with class c. The objective function that NCMML tries to maximize is the log-likelihood for the labeled data in the training set, according to the probability defined above, that is,

$$\mathscr{L}(L) = \frac{1}{N}\sum_{i=1}^N \log p_L(y_i|x_i).$$

This function can be optimized using gradient methods.

3.3.2. Nearest Class with Multiple Centroids (NCMC)

NCMC is the generalization of the nearest class mean classifier. In this classifier, a set with an arbitrary number of centroids is calculated for each class, using a clustering algorithm. Then, a new sample is classified by assigning the label of its nearest centroid.

An immediate generalization of NCMML allows us to learn a distance directed at improving NCMC. This DML algorithm is also referred to as NCMC. In this case, instead of the class means, we have a set of centroids $\{m_{c_j}\}_{j=1}^{k_c}$, for each class $c \in \mathscr{C}$. The generalized probability that a sample $x_i \in \mathscr{X}$ will be labeled with the class c is now given by $p_L(c|x) = \sum_{j=1}^{k_c} p_L(m_{c_j}|x)$, where $p_L(m_{c_j}|x)$ are the probabilities that m_{c_j} is the closest centroid to x, and is given by

$$p_L(m_{c_j}|x) = \frac{\exp\left(-\frac{1}{2}\|L(x-m_{c_j})\|^2\right)}{\sum_{c\in\mathscr{C}}\sum_{i=1}^{k_c}\exp\left(-\frac{1}{2}\|L(x-m_{c_i})\|^2\right)}.$$

Again, NCMC maximizes the log-likelihood function $\mathscr{L}(L) = \frac{1}{N}\sum_{i=1}^{N} p_L(y_i|x_i)$ using gradient methods.

3.4. Information Theory based algorithms

Several DML algorithms rely on information theory to learn their corresponding distances. The information theory concepts used in the algorithms we will introduce below are described in [Appendix B.3] [48]. These algorithms have similar working schemes. First, they establish different probability distributions on the data, and then they try to bring these distributions closer or further away using divergences. The information theory based algorithms we will study are ITML [41], DMLMJ [42] and MCML [43].

3.4.1. Information Theoretic Metric Learning (ITML)

ITML [41] is a DML technique intended to find a distance metric as close as possible to an initial pre-defined distance, on which similarity and dissimilarity constraints for same-class and different-class samples are satisfied. This approach tries to preserve the properties of the original distance while adapting it to our dataset thanks to the restrictions it adds.

We will denote the positive definite matrix associated with the initial distance as M_0. Given any positive definite matrix $M \in S_d(\mathbb{R})^+$ and a fixed mean vector μ, we can construct a normal distribution $p(x|M)$ with mean μ and covariance M. ITML tries to minimize the Kullback–Leibler divergence between $p(x|M_0)$ and $p(x|M)$, subject to several similarity constraints on the data, that is

$$\begin{aligned} \min_{M\in S_d(\mathbb{R})^+} \quad & \mathrm{KL}(p(x|M_0)\|p(x|M)) \\ \text{s.t. :} \quad & d_M(x_i,x_j) \leqslant u, \quad (i,j) \in S \\ & d_M(x_i,x_j) \geqslant l, \quad (i,j) \in D, \end{aligned}$$

where S and D are sets of pairs of indices on the elements of $\mathscr{X}$ that represent the samples considered similar and not similar, respectively (normally, same-labeled pairs and different-labeled pairs), and u and l are, respectively, upper and lower bounds for the similarity and dissimilarity constraints. This problem can be optimized using gradient methods combined with iterated projections in order to fulfill the constraints.

3.4.2. Distance Metric Learning through the Maximization of the Jeffrey divergence (DMLMJ)

DMLMJ [42] is another DML technique based on information theory that tries to separate, with respect to the Jeffrey divergence, two probability distributions, the first associated with similar points while the second is associated with dissimilar points.

DMLMJ defines two difference spaces: a *k-positive difference space* that contains the differences between each sample in the dataset and its k-nearest neighbors from the same class, and a *k-negative difference space* that contains the differences between each sample and its k-nearest neighbors from different classes. Over these spaces, for a distance determined by a linear transformation $L \in \mathscr{M}_{d'\times d}(\mathbb{R})$, two gaussian distributions P_L and Q_L with equal mean are assumed. Then, the problem that DMLMJ optimizes is

$$\max_{L\in\mathscr{M}_{d'\times d}(\mathbb{R})} \quad f(L) = \mathrm{JF}(P_L\|Q_L) = \mathrm{KL}(P_L\|Q_L) + \mathrm{KL}(Q_L\|P_L).$$

This problem can be transformed into a trace optimization problem similar to those of PCA and LDA, and can also be solved by taking eigenvectors from the covariance matrices involved in the problem.

3.4.3. Maximally Collapsing Metric Learning (MCML)

MCML [43] is another DML technique based on information theory. The key idea of this algorithm is the fact that we would obtain an ideal class separation if we could project all the samples from the same class on a same point, far enough away from the points on which the rest of the classes would be projected.

In order to try to achieve this, MCML defines a probability that a sample x_j will be classified with the same label as x_i, with the distance given by a positive semidefinite matrix $M \in S_d(\mathbb{R})_0^+, p^M(j|i)$, as the softmax

$$p^M(j|i) = \frac{\exp(-d_M(x_i,x_j)^2)}{\sum_{k\neq i}\exp(-d_M(x_i,x_k)^2)}.$$

Then, it also defines a probability $p_0(j|i)$ for the ideal situation in which all the same-class samples collapse into the same point, far enough away from the collapsing points of the other classes, given by

$$p_0(j|i) \propto \begin{cases} 1, & y_i = y_j \\ 0, & y_i \neq y_j \end{cases}.$$

MCML tries to bring $p^M(\cdot|i)$ as close to the ideal $p^0(\cdot|i)$ as possible, for each i, using the Kullback–Leibler divergence between them. Therefore, the optimization problem is formulated as

$$\min_{M\in S_d(\mathbb{R})_0^+} \quad f(M) = \sum_{i=1}^{N} \mathrm{KL}\left[p_0(\cdot|i)\|p^M(\cdot|i)\right].$$

This function can be minimized using semidefinite programming.

3.5. Other distance metric learning techniques

In this section we will study some different proposals for DML techniques. The algorithms we will analyze are LSI [4], DML-eig [67] and LDML [68].

3.5.1. Learning with Side Information (LSI)

LSI [4], also sometimes referred to as Mahalanobis Metric for Clustering (MMC) is possibly one of the first algorithms that has helped make the concept of DML more well known. This algorithm is a global approach that tries to bring same-class data closer together while keeping data from different classes far enough apart.

Assuming that the sets S and D represent, respectively, pairs of samples that should be considered similar or dissimilar (i.e. samples that belong to the same class or to different classes, respectively), LSI looks for a positive semidefinite matrix $M \in S_d(\mathbb{R})_0^+$ that optimizes the following problem:

$$\begin{aligned} \min_M \quad & \sum_{(x_i,x_j)\in S} d_M(x_i,x_j)^2 \\ \text{s.t. :} \quad & \sum_{(x_i,x_j)\in D} d_M(x_i,x_j) \geqslant 1 \\ & M \in S_d(\mathbb{R})_0^+. \end{aligned}$$

This problem can be optimized using gradient descent together with iterated projections in order to fulfill the constraints.

3.5.2. Distance Metric Learning with eigenvalue optimization (DML-eig)

DML-eig [67] is a DML algorithm inspired by the LSI algorithm of the previous section, proposing a very similar optimization problem but offering a completely different resolution method, based on eigenvalue optimization.

We will once again consider the two sets S and D, of pairs of samples that are considered similar and dissimilar, respectively. DML-eig proposes an optimization problem that slightly differs from that proposed by the LSI algorithm, given by

$$\begin{aligned} \max_{M} \quad & \min_{(x_i,x_j)\in D} d_M(x_i,x_j)^2 \\ \text{s.t. :} \quad & \sum_{(x_i,x_j)\in S} d_M(x_i,x_j)^2 \leqslant 1 \\ & M \in S_d(\mathbb{R})_0^+. \end{aligned}$$

This problem can be transformed into a minimization problem for the largest eigenvalue of a symmetric matrix. This is a well-known problem and there are some iterative methods that allow this minimum to be reached [69].

3.5.3. Logistic Discriminant Metric Learning (LDML)

LDML [68] is a DML algorithm in which the optimization model makes use of the logistic function.

Recall that the *logistic* or *sigmoid* function is the map $\sigma : \mathbb{R} \to \mathbb{R}$ given by

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

In LDML, the logistic function is used to define a probability, which will assign the greater probability the smaller the distance between points. Given a positive semidefinite matrix $M \in S_d(\mathbb{R})_0^+$, this probability is expressed as

$$p_{ij,M} = \sigma(b - \|x_i - x_j\|_M^2),$$

where b is a positive threshold value that will determine the maximum value achievable by the logistic function, and that can be estimated by cross validation. LDML tries to maximize the log-likelihood given by

$$\mathscr{L}(M) = \sum_{i,j=1}^{N} y_{ij} \log p_{ij,M} + (1-y_{ij}) \log(1-p_{ij,M}),$$

where y_{ij} is a binary variable that takes the value 1 if $y_i = y_j$ and 0 otherwise. This function can be optimized using semidefinite programming.

3.6. Kernel distance metric learning

Kernel methods constitute a paradigm within machine learning that is very useful in many of the problems addressed in this discipline. They usually arise in problems where the learning algorithm capability is reduced, typically due to the shape of the dataset. A classic learning algorithm where the kernel trick is very useful is the *Support Vector Machines (SVM)* classifier [70]. An example for this case is given in Fig. 5.

In DML, the usefulness of kernel learning is a consequence of the limitations given by the Mahalanobis distances. Although learned metrics can later be used with non-linear classifiers, such as the nearest neighbors classifier, the metrics themselves are determined by linear transformations, which, in turn, are determined by the image of a basis in the departure space, which results in the fact that we only have the freedom to choose the image of as much data as the dimension has the space, mapping the rest of the vectors by linearity. When the amount of data is much larger than the space dimension this can become a limitation.

The kernel approach for DML follows a similar scheme to that of SVM. If we work with a dataset $\mathscr{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$, the idea is to send the data to a higher dimensional space, using a mapping $\phi : \mathbb{R}^d \to \mathscr{F}$, where $\mathscr{F}$ is a Hilbert space called the *feature space*, and then to learn in the feature space using a DML algorithm. The way we will learn a distance in the feature space will be via a continuous linear transformation $L : \mathscr{F} \to \mathbb{R}^{d'}$, where $d' \leqslant d$ (observe that L is not necessarily a matrix, since $\mathscr{F}$ is not necessarily finite dimensional), which we will also denote as $L \in \mathscr{L}(\mathscr{F}, \mathbb{R}^{d'})$.

As occurs with SVM, a great inconvenience arises when sending the data to the feature space, and that is that the problem dimension can greatly increase, and therefore the application of the algorithms can be very expensive computationally. In addition, if we want to work in infinite dimensional feature spaces, it is impossible to deal with the data in this case, unless we turn to the kernel trick.

We define the *kernel function* as the mapping $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by $K(x,x') = \langle \phi(x), \phi(x') \rangle$. The success of kernel functions is due to the fact that many learning algorithms only need to know the dot products between the elements in the training set to be able to work. This will happen in the DML algorithms we will study later. We can observe, as an example, that the calculation of Euclidean distances, which is essential in many DML algorithms, can be made using only the kernel function. Indeed, for $x, x' \in \mathbb{R}^d$, we have

$$\begin{aligned} \|\phi(x) - \phi(x')\|^2 &= \langle \phi(x) - \phi(x'), \phi(x) - \phi(x') \rangle \\ &= \langle \phi(x), \phi(x) \rangle - 2\langle \phi(x), \phi(x') \rangle + \langle \phi(x'), \phi(x') \rangle \\ &= K(x,x) + K(x',x') - 2K(x,x'). \end{aligned} \quad (1)$$

The next common problem for all the kernel-based DML algorithms is how to deal with the learned transformation. Since we are trying to learn a map $L \in \mathscr{L}(\mathscr{F}, \mathbb{R}^{d'})$, we may not be able to write it as a matrix, and when we can, this matrix may have dimensions that are too large. However, as L is continuous and linear, using the Riesz representation theorem, we can rewrite L as a vector of dot products by fixed vectors, that is, $L = (\langle \cdot, w_1 \rangle, \ldots, \langle \cdot, w_{d'} \rangle)$, where $w_1, \ldots, w_{d'} \in \mathscr{F}$. Furthermore, for the algorithms we will study, several *representer theorems* are known [71,52,45,42,72]. These theorems allow the vectors w_i to be expressed as a linear combination of the samples in the feature space, that is, for each $i \in \{1, \ldots, d'\}$, there is a vector $\alpha^i = (\alpha^i_1, \ldots, \alpha^i_N) \in \mathbb{R}^N$ so that $w_i = \sum_{j=1}^N \alpha^i_j \phi(x_j)$. Consequently, we can see that

$$L\phi(x) = A \begin{pmatrix} K(x_1, x) \\ \vdots \\ K(x_N, x) \end{pmatrix}, \quad (2)$$

where $A \in \mathscr{M}_{d' \times N}(\mathbb{R})$ is given by $A_{ij} = \alpha^i_j$.

Thanks to these theorems, we can address the problem computationally as long as we are able to calculate the coefficients of matrix A. When transforming a new sample it will suffice to construct the previous column matrix by evaluating the kernel function between the sample and each element in the training set, and then multiplying A by this matrix. On a final note, when training it is useful to view the kernel map as a matrix $K \in S_N(\mathbb{R})$, where $K_{ij} = K(x_i, x_j)$. A similar (in this case not necessarily square) matrix can be constructed when testing, with all the dot products between the train and test samples. By choosing the appropriate

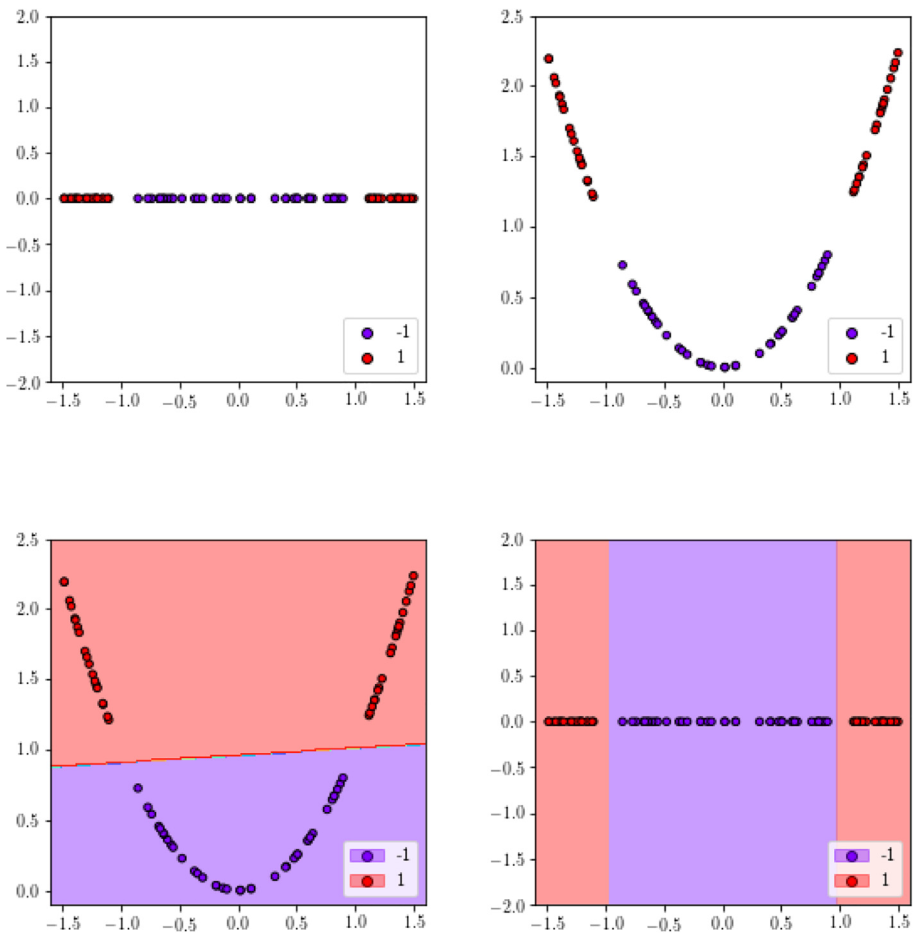


Fig. 5. SVM and kernel trick. This binary classifier looks for the hyperplane that best separates both classes. Therefore, it is highly limited when the dataset is not separable by hyperplanes, as is the case for the dataset in the upper-left image. A solution consists of sending the data into a higher dimensional space, where data can be separated by hyperplanes, and apply the algorithm there, as it can be seen in the remaining images. The kernel trick allows us to execute the algorithm only in terms of the dot products of the samples in the new space, which makes it possible to work on very high dimensional spaces, or even infinite dimensional spaces. The existence of a *representer theorem* for SVM also allows the solution to be rewritten in terms of a vector with the size of the number of samples.

column of this matrix, we will be able to transform the corresponding test sample using Eq. 2.

Each DML technique that supports the use of kernels will use different tools for its performance, each one based on the original algorithms. In [Appendix C.6] [48] we describe the kernelizations of some of the algorithms already introduced, namely, LMNN, ANMM, DMLMJ and LDA.

4. Experimental framework and results

With the algorithms introduced in the previous section, several experiments have been carried out. This section describes these experiments and shows the results.

4.1. Description of the experiments

For the DML algorithms studied, a collection of experiments has been developed, consisting of the following procedures.

1. Evaluation of all the algorithms capable of learning at maximum dimension, applied to the k-NN classification, for different values of k.
2. Evaluation of the algorithms aimed at improving nearest centroid classifiers, applied to the corresponding centroid-based classifiers.
3. Evaluation of kernel-based algorithms, experimenting with different kernels, applied to the nearest neighbors classification.
4. Evaluation of algorithms capable of reducing dimensionality, for different dimensions, applied to the nearest neighbors classification.

When we mention in experiment 1 that an algorithm is "capable of learning at maximum dimension" we are excluding those dimensionality reduction algorithms that only learn a change of axes, as is the case with both PCA and ANMM, which at maximum dimension learn a transformation whose associated distance is still the Euclidean. LDA is kept, assuming that it will always take the maximum dimension that it is able to, which will be the number of classes of the problem. The algorithms directed at centroid-based classifiers are also excluded from experiment 1, together with those based on kernels, which will be analyzed in experiments 2 and 3, respectively.

The stated experiments indicate that the magnitude with which we will measure the performance of the algorithms is the result of the k-neighbors classification, except in the case of the algorithms based on centroids, which will use their corresponding classifier. These classifiers will be evaluated by a 10-fold cross validation. The results obtained from the predictions on the training set will also be included, in order to evaluate possible overfitting.

To evaluate the algorithms, we will use the implementations available in the Python library `pyDML` [47]. The algorithms will be executed using their default parameters, which can be found in the `pyDML` documentation.[1] These default parameters have been set with standard values. The following exceptions to the default parameters have been made:

- The LSI algorithm will have the parameter `supervised = True`, as it will be used for supervised learning.
- In the dimensionality reduction experiment (4), the algorithms will have the dimension number parameter set with the value of the dimension being evaluated.
- The parameter `k` of LMNN and KLMNN will be equal to the number of neighbors being considered in the nearest neighbors classification.
- LMNN will be executed with stochastic gradient descent, instead of semidefinite programming, in dimensionality reduction experiments, thus learning a linear transformation instead of a metric.
- The parameters `n_friends` and `n_enemies` of ANMM and KANMM will be equal to the number of neighbors being considered in the nearest neighbors classification.
- The parameter `n_neighbors` of DMLMJ and KDMLMJ will be equal to the number of neighbors being considered in the nearest neighbors classification.
- The parameter `centroids_num` of NCMC will be equal to the parameter `centroids_num` being considered in its corresponding classifier, `NCMC_Classifier`.

As for the datasets used in the experiments, up to 34 datasets have been collected and all of them are available in KEEL.[2] All these datasets are numeric, do not contain missing values, and are oriented to standard classification problems. In addition, although some of the DML algorithms scale well with the number of samples, others cannot deal with datasets that are too large, so it was decided that for sets with a high number of samples, a subset of a size that all algorithms can deal with, keeping the class distribution the same, would be selected. The characteristics of these datasets are described in Table 2. All datasets have been *min–max* normalized to the interval $[0,1]$, feature to feature, prior to the execution of the experiments.

Finally, we describe the details of the experiments 1, 2, 3 and 4:

1. Algorithms will be evaluated with the classifiers 3-NN, 5-NN and 7-NN.
2. NCMML will be evaluated with the `Scikit-Learn` NCM classifier, while NCMC will be evaluated with its associated classifier, available in `pyDML`, for two different values: 2 centroids per class and 3 centroids per class.
3. Algorithms will be evaluated with 3-NN classifier, using the following kernels: linear (`Linear`), grade-2 (`Poly-2`) and grade-3 (`Poly-3`) polynomials, gaussian (`RBF`) and laplacian (`Laplacian`). The kernel version of PCA[3] will be also included in the comparison. Only the smallest datasets will be considered, so that they can be applicable to the algorithms that scale the worst with the dimension (recall that the kernel trick forces algorithms to work in dimensions of the order of the number of samples).

[1] https://pydml.readthedocs.io/

[2] KEEL, *knowledge extraction based on evolutionary learning* [73]:http://www.keel.es/.

[3] It is implemented in `Scikit-Learn`:http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html. Its theoretical details can be found in [72].

Table 2
Datasets used in the experiments.

Dataset	Number of samples	Number of features	Number of classes
appendicitis	106	7	2
balance	625	4	3
bupa	345	6	2
cleveland	297	13	5
glass	214	9	7
hepatitis	80	19	2
ionosphere	351	33	2
iris	150	4	3
monk-2	432	6	2
newthyroid	215	5	3
sonar	208	60	2
wine	176	13	3
movement_libras	360	90	15
pima	768	8	2
vehicle	846	18	4
vowel	990	13	11
wdbc	569	30	2
wisconsin	683	9	2
banana (20 %)	1,060	2	2
digits	1,797	64	10
letter (10 %)	2,010	16	26
magic (10 %)	1,903	10	2
optdigits	1,127	64	10
page-blocks (20 %)	1,089	10	4
phoneme (20 %)	1,081	5	2
ring (20 %)	1,480	20	2
satimage (20 %)	1,289	36	7
segment (20 %)	462	19	7
spambase (10 %)	460	57	2
texture (20 %)	1,100	40	11
thyroid (20 %)	1,440	21	3
titanic	2,201	3	2
twonorm (20 %)	1,481	20	2
winequality-red	1,599	11	11

4. Algorithms will be evaluated with the classifiers 3-NN, 5-NN and 7-NN. The dimensions used are: $1,2,3,5,10,20,30,40,50$, the maximum dimension of the dataset, and the number of classes of the dataset minus 1. In this case, the following high-dimensionality datasets are selected: `sonar`, `movement_libras` and `spambase`. The algorithms to be evaluated in this experiment are: PCA, LDA, ANMM, DMLMJ, LMNN and NCA.

4.2. Results

This section shows the results of the cross-validation for the different experiments. We will only show the results of the 3-NN classifier in the experiments that use nearest neighbors classifiers in this text. The results obtained for the remaining k-NN used in the experiments are available on the pyDML-Stats[4] website, where the results of all these experiments have been stored. The scripts used to do the experiments can also be found on this website. We have added the average score obtained and the average ranking to the results of the experiments 1, 2 and 3. The ranking has been made by assigning integer values between 1 and m, where m is the number of algorithms being compared in each experiment (adding half fractions in case of a tie), according to the position of the algorithms over each dataset, 1 being the best algorithm, and m the worst. The content of the different tables elaborated is described below.

[4] Source code:https://github.com/jlsuarezdiaz/pyDML-Stats. The current website is located athttps://jlsuarezdiaz.github.io/software/pyDML/stats/versions/0.0.1-1/

Table 3
Results of cross-validation with 3-NN.

	Euclidean		LDA		ITML		DMLMJ		NCA		LMNN		LSI		DML-eig		MCML		LDML	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
appendicitis	.8428	.8339	.8428	.8522	.8533	.8604	.8490	.8256	**.8700**	.8504	.8407	.8422	.8659	**.8630**	.8585	.8622	.8502	.8513	.8669	.8422
balance	.8049	.8082	.8885	.8992	.8986	.8943	.8286	.8191	**.9592**	**.9584**	.8202	.8175	.9182	.9280	.8947	.8945	.8816	.8737	.8874	.8895
bupa	.6231	.6546	.6338	.6465	.6466	.6281	.6660	**.6776**	**.6943**	.5994	.6099	.6342	.6363	.6284	.5993	.6120	.5716	.5742	.5826	.5854
cleveland	.5570	.5468	.5694	.5502	.5488	.5523	.5626	.5636	**.6804**	.5436	.5780	.5803	.5518	.5722	.5896	.5829	.5978	.5785	.5784	**.5972**
glass	.6759	.7015	.6235	.6231	.6453	.6549	**.7092**	.7041	.7065	.6917	.6780	**.7067**	.6495	.6235	.6407	.6263	.6319	.5850	.6242	.6063
hepatitis	.8236	.8325	.9402	.8609	.9002	.8815	.8821	.8894	**.9569**	.8325	.9514	.8418	.9139	.9130	.9125	**.9176**	.9250	.8829	.9458	.8547
ionosphere	.8569	.8550	.8834	.8394	.8771	.8862	.8752	.8605	**.9534**	**.9084**	.9281	.8859	.8898	.8768	.8904	.8741	.9053	.8630	.8907	.8512
iris	.9533	.9533	.9681	.9533	.9703	.9733	.9585	.9666	.9755	.9666	.9481	.9400	.9703	**.9800**	.9585	.9600	.9688	.9466	**.9807**	.9600
monk-2	.9578	.9655	.9451	.9561	.9223	.9352	.9709	.9724	**1.000**	**1.000**	.9812	.9816	**1.000**	**1.000**	.9878	.9909	.9665	.9676	.9382	.9495
newthyroid	.9421	.9538	.9596	.9586	.9452	.9398	.9436	.9448	**.9700**	.9722	.9658	**.9725**	.9591	.9634	.9602	.9629	.9565	.9582	.9509	.9675
sonar	.8317	.8370	.9011	.7782	.8435	.8120	.9097	.8361	.9823	.8703	**.9941**	**.8742**	.8531	.8506	.8547	.7975	.8755	.8563	.8766	.7886
wine	.9606	.9606	.9968	**.9888**	.9900	.9773	.9812	.9662	.9956	.9882	.9956	.9832	.9837	.9662	.9975	.9767	**.9975**	.9832	.9956	**.9888**
movement_libras	.7972	.8139	**.8685**	.6642	.8038	.7992	.8460	**.8649**	.8516	.8319	.8065	.8020	.7351	.7440	.7970	.7872	.8063	.8073	.7256	.7360
pima	.7372	.7396	.7259	**.7525**	.7148	.7149	.7366	.7422	**.7841**	.7370	.7290	.7278	.7206	.7395	.7174	.7266	.7173	.7239	.7285	.7240
vehicle	.7077	.7125	.7698	**.7623**	.7625	.7516	.7643	.7551	**.8186**	.7550	.6855	.6757	.6590	.6666	.6506	.6501	.7398	.7369	.7186	.7170
vowel	.9699	.9787	.9680	.9777	.9423	.9535	.9751	**.9808**	**.9799**	**.9808**	.9693	.9777	.9436	.9474	.6719	.6757	.8558	.8737	.8885	.9090
wdbc	.9679	**.9716**	.9732	.9664	.9714	.9664	.9669	.9648	**.9751**	.9700	.9638	.9630	.9705	.9682	.9546	.9507	.9714	.9648	.9476	.9438
wisconsin	.9694	.9678	.9663	.9677	.9609	.9590	.9695	.9678	**.9723**	.9648	.9692	.9663	.9684	**.9722**	.9673	.9707	.9585	.9546	.9650	.9663
banana	.8543	.8555	.6504	.6469	.8536	.8556	.8550	.8565	.8553	**.8583**	**.8574**	.8583	.8535	.8517	.6718	.6878	.6282	.6102	.6268	.6319
digits	.9878	.9866	.9769	.9683	.9798	.9728	.9869	.9834	.9980	**.9894**	**.9993**	.9860	.9264	.9102	.8269	.8168	.9734	.9688	.9797	.9816
letter	.7174	.7208	.7955	.7967	.7161	.7195	.8163	.8204	**.8565**	**.8610**	.7048	.7162	.5396	.5496	.3191	.3214	.7600	.7534	.6217	.6372
magic	.8070	.8050	.7436	.7361	.8069	.8061	.8161	.8071	**.8396**	**.8145**	.7979	.7945	.7946	.7924	.7508	.7525	.7738	.7766	.7077	.6951
optdigits	.9756	.9777	.9671	.9512	.9731	.9669	.9770	.9761	.9956	.9759	**.9986**	**.9840**	.9398	.9306	.8164	.8022	.9761	.9591	.9596	.9591
page-blocks	.9495	.9495	**.9697**	**.9679**	.9614	.9614	.9515	.9504	.9637	.9577	.9459	.9439	-	-	.9515	.9523	.9613	.9642	.9438	.9404
phoneme	.7957	.7992	.7321	.7243	.7853	.7770	.7960	**.8002**	**.8044**	.7936	.7928	.7946	.7642	.7668	.7361	.7483	.7654	.7632	.7320	.7112
ring	.6410	.6432	.7289	.7101	.7290	.7352	.6440	.6453	**.9267**	**.8459**	.6750	.6615	.8331	.8162	.7308	.7223	.8315	.8223	.5634	.5648
satimage	.8585	.8564	.8541	.8387	.8495	.8341	.8670	**.8643**	**.8764**	.8511	.8565	.8558	.8490	.8465	.8153	.8130	.8246	.8171	.5427	.5501
segment	.8970	.9020	.9353	**.9370**	.9357	.9265	.9071	.9081	**.9451**	.9187	.9076	.8928	.8898	.8853	.9095	.9068	.9367	.9319	.8818	.8710
spambase	.8500	.8654	.9215	.8871	.8801	.8766	.8635	.8525	**.9391**	**.9154**	.9215	.9070	.9210	.9111	.9077	.9046	.9176	.9047	.9229	.8982
texture	.9560	.9618	**.9983**	**.9981**	.9801	.9754	.9864	.9854	.9843	.9800	.9180	.9218	.9333	.9400	.8979	.9009	.9740	.9745	.8658	.8718
thyroid	.9313	.9319	.9375	.9450	.9397	.9402	.9355	.9361	.9459	.9395	.9320	.9319	.9357	.9354	.9458	.9485	.9377	.9320	**.9587**	**.9583**
titanic	.7607	.7583	**.7727**	**.7804**	.7682	.7609	.7612	.7587	.5709	.6764	.6018	.6964	-	-	.7107	.7331	.7150	.7253	.7108	.7341
twonorm	.9609	.9595	.9778	.9750	.9685	.9669	.9612	.9561	**.9817**	.9790	.9778	.9756	.9776	.9770	.9782	**.9810**	.9708	.9730	.9789	.9804
winequality-red	.5808	**.5865**	.5657	.5733	.5754	.5828	.5828	.5860	**.6022**	.5766	.5647	.5772	.5656	.5809	.5281	.5292	.5675	.5611	.5376	.5471
AVG RANKING	6.558	5.661	5.147	5.426	5.808	5.382	4.926	4.544	**1.705**	**3.661**	5.220	5.279	6.411	5.382	6.691	6.220	5.735	6.279	6.794	7.161
AVG SCORE	.8383	.8425	.8515	.8363	.8500	.8470	.8560	.8526	**.8886**	**.8634**	.8490	.8432	.8410	.8405	.8059	.8041	.8438	.8359	.8125	.8062

- Table 3 shows the cross-validation results obtained for experiment 1, using the 3-NN score as the evaluation measure. Some cells do not show results because the algorithm did not converge.
- Table 4 shows the results of experiment 2. NCM and NCMC classifiers with 2 and 3 centroids per class were used as evaluation measures. For each classifier, the Euclidean distance (`Euclidean + CLF`) and the distance learning algorithm associated with the classifier (`NCMML/ NCMC (2 ctrd)/ NCMC (3 ctrd)`) have been evaluated.
- Table 5 shows the cross-validation results obtained on the training set for the kernel-based algorithms using the 3-NN classifier. Table 6 shows the corresponding results obtained on the test set.
- Table 7 shows the cross-validation results for experiment 4 in dataset `sonar`, using the classifier 3-NN. On the left are the results for the training set, and on the right, the results for the test set. Each row shows the results for the different dimensions evaluated. Tables 8 and 9 show the corresponding dimensionality results over the datasets `movement_libras` and `spambase`, respectively.

4.3. Analysis of results

4.3.1. In-depth analysis

Below we will describe the main details observed in the algorithms for the different experiments carried out.

- **NCA.** In terms of the results obtained in the first experiment, we can clearly see that NCA has obtained the best results. This is partly due to the fact that the algorithms have been evaluated with nearest neighbors classifiers, and that NCA was specifically designed to improve this classifier. NCA came first in most of the validations over the training set, showing its ability to fit to the data, but it has also obtained clear victories in many of the datasets over the test set, thus also demonstrating a great capacity for generalization. We have to note that NCA (and also other algorithms such as LMNN) commits substantial errors in datasets such as *titanic* [73]. This is a numerical-transformed dataset, but of a categorical nature, and with many repeated elements that may belong to different classes. This may be causing highly discriminative algorithms such as NCA or LMNN not being able to transform the dataset appropriately. This justifies how in certain situations other algorithms can be more useful than those that show better behavior in general [74].
- **LMNN and DMLMJ.** We can also see that DMLMJ and LMNN algorithms stand out, although not as much as NCA. These algorithms are also directed at nearest neighbor classification, which justifies these good results. LMNN seems to have a slow convergence with the projected gradient method, and it could have achieved better results with a greater number of iterations. In fact, in the analysis of dimensionality reduction experiments we will observe that LMNN performs much better with the stochastic gradient descent method.
- **LSI.** LSI is another algorithm that is capable of obtaining very good results on certain datasets, but it is penalized by many others, where it is not able to optimize enough, not even being able to converge in several datasets.
- **ITML and MCML.** ITML and MCML are two algorithms that, despite getting the best results in a very small number of cases, they get decent results in most datasets, resulting in quite a stable performance. ITML does not learn too much from the characteristics of the training set, but is able to generalize what has been learned in quite an effective way, being possibly the algorithm that loses the least accuracy over the test set, with respect to the training set. On the other hand, MCML has more learning capacity, even showing a slight overfitting, as its results are worse than those of many algorithms on the test set.
- **LDA.** Another algorithm in which we can see overfitting, perhaps more clearly, is LDA. This algorithm is capable of getting very good results on the training set, surpassing most of the algorithms, but it gets noticeably worse when evaluated on the test dataset. Recall that LDA is able to learn only a maximum

Table 4
Results of the experiments with NCMML and NCMC.

	Euclidean + NCM		NCMML		Euclidean + NCM (2 ctrd)		NCMC (2 ctrd)		Euclidean + NCM (3 ctrd)		NCMC (3 ctrd)	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
appendicitis	.8365	**.8640**	**.8512**	.8440	.6479	.6031	.6248	.6246	.8102	.7800	.7557	.7266
balance	.7496	.7475	.6865	.6864	.7004	.6721	.8280	**.8225**	.7160	.6654	**.8321**	.8222
bupa	.5996	.6004	**.6628**	**.6407**	.6270	.6058	.6247	.6260	.6589	.5903	.6409	.5826
cleveland	.5742	.5377	.6296	**.5516**	.5727	.5050	.6280	.5093	.6203	.4871	**.6435**	.5135
glass	.5218	.4862	.6221	.5290	.6479	.5849	.5877	.5372	.6427	.5208	**.7461**	**.6421**
hepatitis	.8500	.8293	.9832	**.8688**	.8792	.8436	.9513	.8373	.8986	.7946	**.9833**	.8672
ionosphere	.7445	.7378	.9313	.8797	.9186	**.8889**	.9018	.8764	.9062	.8750	**.9477**	.8886
iris	.9318	.9133	**.9814**	.9600	.9696	**.9666**	.9711	.9600	.9696	.9466	.9800	.9600
monk-2	.8078	.8104	.7950	.7923	.8071	.8082	.8112	.8038	.8127	.7895	**.8457**	**.8237**
newthyroid	.9359	.9352	**.9798**	.9634	.9498	.9448	.9741	**.9725**	.9695	.9681	.9757	.9629
sonar	.7265	.7017	.9257	.7641	.7120	.6929	.9219	.7839	.8360	.7437	**.9412**	**.7982**
wine	.9687	.9495	**1.000**	.9663	.9825	.9659	.9918	**.9826**	.9762	.9432	.9912	.9704
movement_libras	.6358	.5946	.8176	.7575	.7777	.6764	.8803	.7852	.8717	.7749	**.9420**	**.8373**
pima	.7337	.7279	**.7717**	**.7604**	.7565	.7382	.6720	.6641	.7521	.7461	.7322	.7253
vehicle	.4545	.4491	**.7998**	**.7797**	.6066	.5824	.7429	.7219	.6537	.6179	.7558	.7244
vowel	.5367	.5070	.6689	.6383	.6087	.5717	.7272	.6868	.5732	.5333	**.7690**	**.7292**
wdbc	.9388	.9367	.9794	.9649	.9548	.9385	.9796	**.9736**	.9703	.9665	**.9810**	.9701
wisconsin	.9648	.9648	**.9681**	**.9662**	.9586	.9502	.9655	.9603	.9515	.9486	.9541	.9487
banana	.5769	.5737	.5571	.5558	.6417	.6359	.5846	.5859	.7516	.7573	**.7828**	**.7766**
digits	.9066	.8981	.8047	.8083	.9521	.9415	.8664	.8586	**.9723**	**.9644**	**.8893**	.8554
letter	.5707	.5341	.7114	.6868	.5895	.5282	.7251	.6902	.6601	.5714	**.7825**	**.7301**
magic	.7712	.7693	.7790	.7751	.7786	.7745	**.7947**	**.7861**	.7600	.7509	.7820	.7751
optdigits	.9173	.9104	.8018	.7978	.9479	.9352	.8542	.8185	**.9675**	**.9609**	.8923	.8641
page-blocks	.8133	.8165	**.9636**	**.9576**	.8219	.8221	.9090	.9071	.8680	.8696	.8966	.8953
phoneme	.7417	.7399	.7587	.7538	**.7683**	**.7667**	.7084	.7159	.7207	.7149	.7091	.7048
ring	.7780	.7723	.7819	**.7784**	.8002	.7601	.7197	.7012	**.8160**	.7750	.6844	.6636
satimage	.7868	.7844	**.8478**	**.8255**	.8052	.7921	.8342	.8123	.8244	.7844	.8362	.8007
segment	.8460	.8367	**.9437**	**.9037**	.8596	.8452	.9203	.8976	.8581	.8030	.9242	.8874
spambase	.8874	.8827	**.9584**	.9154	.8816	.8763	.9400	**.9241**	.8985	.8893	.9335	.9112
texture	.7445	.7372	**.9912**	**.9781**	.8586	.8500	.9694	.9581	.9079	.8900	.9759	.9654
thyroid	.4532	.4394	**.8163**	**.8082**	.5724	.5558	.6875	.6922	.5959	.5630	.7469	.7358
titanic	.7540	.7459	**.7825**	**.7854**	.6746	.6516	.5624	.6550	.5630	.6824	.7279	.7350
twonorm	.9807	**.9824**	.9854	.9797	.9799	.9723	.9847	.9790	.9787	.9743	**.9855**	.9777
winequality-red	.3519	.3371	**.4535**	**.4359**	.4067	.3838	.4031	.3870	.3940	.3582	.4024	.3738
AVG RANKING	4.941	4.441	**2.382**	**2.500**	4.117	4.000	3.411	2.911	3.764	4.235	**2.382**	2.911
AVG SCORE	.7468	.7369	.8233	.7958	.7770	.7538	.8014	.7793	.7978	.7647	**.8344**	**.7984**

dimension equal to the number of classes of the dataset minus one. This may be causing a loss of important information on many datasets by the projection it learns.

- **DML-eig and LDML.** Finally, although DML-eig and LDML are able to get better results than Euclidean distance on the training sets, on several datasets they have obtained quite low quality results. On many of the test datasets, they are surpassed by the Euclidean distance.
- **Untrained kNN**. The untrained kNN or, equivalently, the classical kNN with Euclidean distance, is always outperformed by some of the distance-learned*k*-NNs in the training set, and is also mostly outperformed in the test set. This shows the benefits of learning a distance as opposed to the traditional use of the nearest neighbor classifier. The untrained kNN also shows better average results on the test set than on the training set, and is the only one among all the compared algorithms. This may be due to the fact that, as it is not using a pretrained distance, it is unlikely to overfit, although according to the results there is a lot of room for improvement in both training and test sets for this basic version.
- **NCMML and NCMC.** If we analyze the results of the centroid-based classifiers, we can easily observe that in the vast majority of cases the classifier has worked much better after learning the distance with its associated learning algorithm, than it has by using the Euclidean distance. It can also be observed that the results are subject to great variability, depending on the number of centroids chosen. This shows that the choice of an adequate number of centroids that adapt well to the disposition of the different classes is fundamental to achieve successful learning with these algorithms.
- **Kernel algorithms.** Focusing now on the kernel-based algorithms, it is interesting to note how KLMNN with laplacian kernel is able to adjust as much as possible to the data, getting a 100 % success rate on most of the datasets. This success rate is not transferred, in general, to the test data, showing that this algorithm overfits with laplacian kernel. We can also observe that the best results are distributed in a varied way among the different evaluated options. The choice of a suitable kernel that fits well with the disposition of the data is decisive for the performance of kernel-based algorithms.
- **Dimensionality reduction experiments.** To conclude our analysis, dimensionality experiments allow us to observe that the best results are not always obtained when considering the maximum dimension. This may be due to the fact that the algorithms are able to denoise the data, ensuring that the classifier used later does not overfit. We also see that we cannot reduce the dimension as much as we want, because at some point we start losing information, which happens in many cases with LDA, which is its great limitation. In general, we can observe that all algorithms improve their results by reducing dimensionality until a certain value, although the best results are provided by LMNN, DMLMJ and NCA. The results obtained by LMNN open the possibility of using this algorithm with stochastic gradient descent, instead of the semidefinite programming algorithm used in the first experiment, since the results it provides are quite good. Although these algorithms have obtained better results, the use of ANMM and LDA (as long as the dimension allows it) is important for the estimation of an adequate dimension, since they are much more efficient than the first ones. As for PCA, it gets the worst results in low dimensions, probably due to not considering the information of the labels.

4.3.2. Global analysis

In order to complete the verbal analysis, we have developed a series of Bayesian statistical tests to assess the extent to which

Table 5
Results of kernel experiments on the training set.

	EUC	KPCA					KDA					KANMM					KDMLMJ					KLMNN				
		Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.
appendicitis	.8428	.8428	.8407	.8355	.8428	.8438	.8481	.8491	.8502	.8679	.8564	.8543	.8543	.8523	.8763	.8889	.8491	.8397	.8407	.8575	.8805	.8355	.8397	.8355	.8523	**1.000**
balance	.8049	.8217	.7770	.7923	.8136	.8288	.4547	.6756	.6279	.7497	.8838	.7764	.7980	.8224	.8462	.8327	.8298	.9509	.9696	.9374	.9491	.8202	.8426	.9079	.8865	**.9852**
bupa	.6231	.6231	.6235	.6222	.6231	.6560	.5623	.5677	.5758	.5526	.5665	.5903	.5890	.5832	.5639	.5735	.6586	.6254	.6457	.6247	.7075	.6437	.6547	.6627	.6721	**.9987**
cleveland	.5570	.5570	.5537	.5552	.5570	.5417	.5301	.5424	.5480	.5585	.5394	.5664	.5705	.5698	.5559	.5435	.5615	.5514	.5544	.5473	.5499	.5536	.5862	.6389	.6753	**.9943**
glass	.6759	.6759	.6801	.6801	.6759	.6931	.6376	.6723	.6459	.6474	.6661	.6884	.6853	.6874	.6724	.6848	.6977	.6957	.7024	.6702	.7342	.6790	.7200	.7289	.7372	**.9890**
hepatitis	.8236	.8236	.8083	.8111	.8236	.8403	.8318	.8416	.8471	.8181	.8111	.8695	.8709	.8653	.8363	.8306	.8834	.8708	.9179	.8708	.9208	.9777	**1.000**	**1.000**	**1.000**	**1.000**
ionosphere	.8569	.8569	.8518	.8493	.8569	.8882	.6986	.6749	.6860	.7641	.7429	.8512	.8518	.8502	.9259	.9449	.8752	.8714	.8774	.8819	.9398	.8695	.9670	.9692	.9838	**1.000**
iris	.9533	.9533	.9540	.9548	.9533	.9518	.8555	.9429	.9488	.9562	.9362	.9459	.9503	.9518	.9451	.9466	.9592	.9503	.9555	.9474	.9592	.9651	.9681	.9659	.9681	**1.000**
monk-2	.9578	.9503	.9603	.9580	.9480	.9511	.7047	.7119	.6995	.8320	.8459	.6877	.6365	.6252	.8996	.9964	.9709	.9763	.9863	.9588	.9938	.9789	.9873	.9920	.9989	**1.000**
newthyroid	.9421	.9421	.9395	.9385	.9421	.9488	.9581	.9576	.9571	.9638	.9612	.9390	.9400	.9410	.9617	.9643	.9441	.9478	.9421	.9503	.9612	.9690	.9700	.9731	.9741	**1.000**
sonar	.8317	.8317	.8365	.8370	.8317	.8392	.6261	.6287	.6410	.6057	.6832	.7499	.7516	.7521	.8263	.8450	.9076	.8696	.8637	.8691	.9332	.9706	**1.000**	**1.000**	**1.000**	**1.000**
wine	.9606	.9606	.9625	.9606	.9606	.9669	.9269	.9169	.9200	.9644	.9694	.9325	.9332	.9325	.9844	.9825	.9825	.9793	.9831	.9819	.9956	.9993	**1.000**	**1.000**	**1.000**	**1.000**
movement_libras	.7972	.7972	.7866	.7805	.7972	.7775	.3322	.4953	.4980	.7419	.7544	.4684	.4684	.4689	.7195	.7376	.8590	.8118	.8272	.8124	.8450	.7945	.8776	.8924	.9246	**1.000**
pima	.7372	.7372	.7361	.7377	.7372	.7181	.6820	.6753	.6653	.6535	.6650	.7170	.7144	.7141	.7164	.7076	.7359	.7381	.7460	.7455	.7628	.7378	.7453	.7397	.7505	**.9968**
banana	.8543	.8546	.8549	.8562	.8545	.8536	.6730	.6622	.6672	.7037	.6493	.8593	.8603	.8624	.7854	.8167	.8554	.8545	.8551	.8423	.7643	.8546	.8546	.8547	.8540	**.9801**
optdigits	.9756	.9756	.9761	.9753	.9756	.9664	.9186	.9190	.9198	.9434	.9386	.9379	.9387	.9394	.9558	.9543	.9767	.9790	.9813	.9791	.9857	.9905	.9999	.9997	.9998	**1.000**
phoneme	.7957	.7957	.7940	.7935	.7957	.7962	.6960	.7019	.6929	.7112	.7271	.7727	.7726	.7732	.7735	.7880	.7959	.7940	.7945	.7898	.7858	.8021	.7963	.7850	.7994	**.9829**
satimage	.8585	.8585	.8583	.8594	.8585	.8641	.8104	.8114	.8097	.8422	.8452	.8188	.8186	.8190	.8535	.8601	.8707	.8623	.8622	.8590	.8680	.8623	.8682	.8682	.8803	**.9935**
segment	.8970	.8970	.8972	.8968	.8970	.9004	.8360	.8278	.8276	.8187	.8737	.8309	.8300	.8300	.8323	.8723	.9049	.8965	.8994	.8879	.9343	.9244	.9383	.9405	.9415	**.9985**
spambase	.8500	.8500	.8500	.8504	.8500	.8328	.8490	.8323	.8313	.7263	.7500	.8777	.8775	.8777	.8635	.8988	.8642	.8850	.8828	.8830	.9014	.9369	.9495	.9459	.9497	**.9990**
twonorm	.9609	.9609	.9636	.9648	.9609	.9531	.9765	.9758	.9758	.9765	.9756	.9798	.9800	.9804	.9763	.9751	.9616	.9642	.9624	.9672	.9810	.9773	.9847	.9800	.9861	**.9915**
AVG RANKING	15.45	15.26	15.52	15.95	15.47	14.14	22.19	21.64	21.50	18.95	19.47	17.14	16.61	16.30	15.52	13.66	9.690	11.71	9.880	12.33	7.023	9.476	5.380	5.833	3.619	**1.214**
AVG SCORE	.8360	.8365	.8336	.8338	.8360	.8387	.7337	.7563	.7541	.7808	.7924	.7959	.7949	.7952	.8271	.8402	.8545	.8530	.8595	.8506	.8740	.8639	.8833	.8895	.8969	**.9957**

Table 6
Results of kernel experiments on the test set.

	EUC	KPCA					KDA					KANMM					KDMLMJ					KLMNN				
		Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.	Lin.	Poly-2	Poly-3	RBF	Lapl.
appendicitis	.8339	.8339	.8339	.8248	.8339	.8546	.8613	.8813	.8813	.8713	.8622	.8646	.8446	.8446	.8813	**.8904**	.8339	.8248	.8339	.8339	.8248	.8322	.8248	.8331	.8057	.8422
balance	.8082	.8383	.7823	.8032	.8428	.8336	.4597	.6699	.6150	.7407	.8899	.7738	.8058	.8382	.8559	.8320	.8144	.9582	**.9711**	.9374	.9519	.8222	.8367	.9118	.8736	.8464
bupa	.6546	.6546	.6661	.6662	.6546	.6862	.5620	.5191	.5566	.5244	.5389	.5963	.6022	.5908	.6199	.5794	.6777	.6375	.6371	.6310	**.6924**	.6574	.6402	.6516	.6632	.6467
cleveland	.5468	.5468	.5432	.5535	.5468	.5528	.5541	.5449	.5485	**.5860**	.5474	.5689	.5653	.5685	.5657	.5623	.5605	.5682	.5774	.5674	.5688	.5736	.5744	.5628	.5184	.5325
glass	.7015	.7015	.7102	.7102	.7015	.7117	.5971	.6708	.6543	.6582	.6630	.6777	.6677	.6677	.7026	.6890	.7020	.6910	.6910	.6715	**.7351**	.6907	.6911	.6827	.6845	.7334
hepatitis	.8325	.8325	.8343	.8343	.8325	.8436	.8123	.8140	.8265	.8325	.8436	.8732	.8732	.8732	.8575	.8575	**.8894**	.8672	.8547	.8547	.8644	.8408	.8672	.8404	.8672	.8845
ionosphere	.8550	.8550	.8520	.8491	.8550	.8885	.7119	.6695	.6977	.7638	.7598	.8517	.8489	.8461	.9258	**.9457**	.8634	.8607	.8606	.8666	.9316	.8463	.9081	.8910	.8969	.9396
iris	.9533	.9533	.9533	.9533	.9533	.9533	.8800	.9533	.9533	**.9800**	.9333	.9600	.9600	.9533	.9533	.9466	.9600	.9600	.9533	.9533	.9533	.9533	.9533	.9600	.9466	.9266
monk-2	.9655	.9539	.9677	.9655	.9634	.9585	.7294	.7203	.6880	.8176	.8385	.6877	.6529	.6435	.9213	.9930	.9724	.9699	.9862	.9654	.9908	.9817	.9863	.9817	.9863	**1.000**
newthyroid	.9538	.9538	.9448	.9448	.9538	.9493	.9538	.9538	.9538	.9586	.9491	.9396	.9396	.9396	.9580	.9627	.9493	.9493	.9448	.9491	.9489	.9584	.9632	**.9679**	.9632	.9625
sonar	.8370	.8370	.8515	.8515	.8370	.8560	.5812	.6540	.6431	.5948	.6872	.7451	.7451	.7451	.8225	.8267	.8461	.8556	.8560	.8651	**.8753**	.7653	.8701	.8408	.8704	.8316
wine	.9606	.9606	.9606	.9606	.9606	.9613	.9214	.9047	.9158	.9367	.9603	.9262	.9318	.9318	.9888	.9835	.9606	.9780	.9724	.9777	.9780	**.9888**	.9830	.9666	**.9888**	.9777
movement_libras	.8139	.8139	.8070	.7948	.8139	.8059	.3715	.5209	.5187	.7631	.7600	.4969	.5006	.4982	.7615	.7566	**.8718**	.8251	.8406	.8219	.8234	.8106	.8315	.8093	.8375	.7989
pima	.7396	.7396	.7370	.7318	.7396	.7175	.6967	.6850	.7150	.6810	.6784	.7238	.7251	.7278	.7134	.7109	.7383	.7396	**.7513**	.7487	.7408	.7462	.7461	.7370	.7370	.7005
banana	.8555	.8555	.8546	.8546	.8555	.8574	.6688	.6642	.6934	.7030	.6169	.8583	.8592	**.8611**	.7810	.8177	.8565	.8546	.8536	.8425	.7622	.8536	.8508	.8565	.8565	.8414
optdigits	.9777	.9777	.9795	.9777	.9777	.9706	.9146	.9155	.9164	.9419	.9350	.9359	.9359	.9377	.9565	.9529	.9752	**.9804**	.9787	.9777	.9804	.9607	.9760	.9697	.9671	.9572
phoneme	.7992	.7992	.7964	.7973	.7992	.8002	.6847	.7068	.7067	.7132	.7270	.7826	.7845	.7854	.7724	.7880	.8030	.8029	.7964	.7918	.7845	**.8047**	.8001	.7854	.7991	.7835
satimage	.8564	.8564	.8564	.8580	.8564	.8612	.8053	.8138	.8122	.8364	.8496	.8193	.8185	.8170	.8535	.8589	**.8689**	.8589	.8580	.8527	.8559	.8473	.8565	.8487	.8425	.8558
segment	.9020	.9020	.9020	.9000	.9020	.9000	.8404	.8363	.8323	.8112	.8625	.8346	.8346	.8326	.8227	.8687	.9098	.8768	.8840	.8738	.9183	.9153	.9241	.9224	.9285	.9517
spambase	.8654	.8654	.8676	.8676	.8632	.8110	.8372	.8348	.8282	.7151	.7303	.8807	.8807	.8807	.8567	.8849	.8546	.8740	.8784	.8826	.8827	.8914	.9027	.8981	**.9049**	.8999
twonorm	.9595	.9595	.9649	.9642	.9595	.9567	.9804	.9777	.9770	.9797	.9730	**.9810**	.9797	.9790	.9743	.9682	.9561	.9635	.9574	.9675	.9702	.9689	.9702	.9682	.9655	.9635
AVG RANKING	12.97	12.83	13.21	13.57	12.76	11.73	21.42	20.73	20.64	17.92	19.21	15.57	15.95	16.16	13.40	12.23	9.738	9.642	10.21	11.61	8.404	11.02	**7.880**	10.38	10.02	11.69
AVG SCORE	.8415	.8424	.8412	.8411	.8430	.8443	.7345	.7577	.7588	.7814	.7908	.7990	.7979	.7982	.8354	.8417	.8507	.8522	.8541	.8492	**.8588**	.8433	.8551	.8517	.8525	.8512

Table 7
Results of dimensionality reduction experiments on `sonar` with 3-NN (train – test).

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.5016	.9011	.6965	.7826	.9214	.7237
2	.5891	–	.7670	.8050	.9807	.8782
3	.7729	–	.8359	.8333	.9770	.9513
5	.8215	–	.8904	.9033	.9759	.9914
10	.8600	–	.8958	.9652	.9764	.9994
20	.8541	–	.8872	.9583	.9668	**1.000**
30	.8456	–	.8627	.9508	.9706	**1.000**
40	.8365	–	.8424	.9460	.9839	**1.000**
50	.8312	–	.8370	.9263	.9850	**1.000**
Max. Dimension	.8317	–	.8317	.9097	.9823	**1.000**
N. Classes – 1	.5016	.9011	.6965	.7826	.9214	.7237
1	.5619	.7782	.6770	.7256	.8073	.6640
2	.6293	–	.7541	.7113	.8077	.7593
3	.7641	–	.8395	.7741	.8265	.8077
5	.8075	–	.8263	.8182	.8220	.8408
10	.8699	–	.8751	.8651	.8270	.8654
20	.8601	–	.8749	**.8844**	.8699	.8703
30	.8610	–	.8649	.8749	.8653	.8754
40	.8465	–	.8610	.8697	.8792	.8613
50	.8565	–	.8515	.8654	.8558	.8706
Max. Dimension	.8370	–	.8370	.8361	.8703	.8706
N. Classes – 1	.5619	.7782	.6770	.7256	.8073	.6640

Table 8
Results of dimensionality reduction experiments on `movement_libras` with 3-NN (train – test)

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.1938	.3339	.2414	.2720	.3360	.2547
2	.2813	.5362	.4597	.4720	.6638	.5416
3	.5232	.6143	.6435	.6684	.7195	.6900
5	.6873	.7211	.7473	.7918	.8188	.8156
10	.7831	.8661	.8053	.8857	.8383	.8485
20	.7978	–	.7972	.8705	.8442	.8490
30	.7981	–	.7978	.8652	.8438	.8514
40	.7972	–	.7975	.8594	.8469	.8526
50	.7972	–	.7972	.8538	.8431	.8498
Max. Dimension	.7972	–	.7972	.8460	.8516	.8490
N. Classes – 1	.7932	.8685	.8061	**.8901**	.8398	.8438
1	.1747	.3169	.2675	.2694	.2606	.2673
2	.2574	.4553	.4800	.4476	.6181	.5251
3	.5483	.4978	.6680	.6684	.6920	.6499
5	.7177	.5938	.7763	.7774	.7655	.8012
10	.8007	.7001	.8119	.8711	.8017	.8220
20	.8139	–	.8106	**.8829**	.8143	.8333
30	.8139	–	.8139	.8696	.8191	.8133
40	.8139	–	.8139	.8605	.8323	.8233
50	.8139	–	.8139	.8627	.8310	.8255
Max. Dimension	.8139	–	.8139	.8649	.8319	.8133
N. Classes – 1	.8185	.6642	.8137	.8811	.8274	.8211

the performance of the different algorithms analyzed outperforms the other algorithms. To do this, we have elaborated several pairwise Bayesian sign tests [46]. In these tests, we will consider the differences between the obtained scores of two algorithms, assuming that their prior distribution is a Dirichlet Process [75], defined by a prior strength $s = 1$ and a prior pseudo-observation $z_0 = 0$. After considering the score observations obtained for each dataset, we obtain a posterior distribution which gives us the probabilities that one algorithm outperforms the other. We also introduce a *rope* (region of practically equivalent) region, in which we consider the algorithms to have equivalent performance. We have designated the rope region to be the one where the score differences are in the interval $[-0.01, 0.01]$. In summary, from the posterior distribution we obtain three probabilities: the probability that the first algorithm outperforms the second, the probability that the second algorithm outperforms the first one, and the probability that both algorithms are equivalent. These probabilities can be visualized in a simplex plot for a sample of the posterior distribution, in which a greater tendency of the points towards one of the regions will represent a greater probability.

To do the Bayesian sign tests, we have used the R package `rNPBST` [76]. In Fig. 6 we pairwise compare some of the algorithms that seem to have better performance in experiment 1 with 3-NN (NCA, DMLMJ and LMNN) with the results of the 3-NN classifier for Euclidean distance. In the comparison made between Euclidean distance and NCA, we can clearly see that the points are concentrated close to the $[\text{NCA}, \text{rope}]$ segment. This shows us that Euclidean distance is unlikely to outperform NCA, and there is also a high probability for NCA to outperform Euclidean distance, since a big concentration of points is in the NCA region. We obtain similar conclusions for DMLMJ against Euclidean distance, although in this case, despite the fact that Euclidean distance is still unlikely to win, there is a greater concentration of points in the rope region. In the comparison made between LMNN and Euclidean distance, we

Table 9
Results of dimensionality reduction experiments on `spambase` with 3-NN (train – test)

	PCA	LDA	ANMM	DMLMJ	NCA	LMNN
1	.8369	.9215	.8567	.6995	**.9420**	.9340
2	.8316	–	.8869	.7724	**.9420**	.9386
3	.8487	–	.8973	.8886	.9388	.9335
5	.8784	–	.9079	.9009	.9415	.9335
10	.8681	–	.9222	.9195	.9400	.9318
20	.8700	–	.9067	.9217	.9400	.9297
30	.8586	–	.8787	.8867	.9403	.9328
40	.8572	–	.8654	.8727	.9369	.9318
50	.8536	–	.8560	.8596	.9374	.9299
Max. Dimension	.8500	–	.8500	.8635	.9391	.9285
N. Classes – 1	.8369	.9215	.8567	.6995	**.9420**	.9340
1	.8106	.8871	.8587	.6588	.9044	.8872
2	.8261	–	.8850	.7173	**.9197**	.8958
3	.8543	–	.9090	.8807	.9152	.9068
5	.8782	–	.9049	.8700	.9111	.9069
10	.8826	–	.9198	.9044	.9153	.9113
20	.8695	–	.9048	.8937	.9155	.9005
30	.8502	–	.8675	.8851	.9154	.9027
40	.8547	–	.8567	.8611	.9111	.9005
50	.8655	–	.8633	.8569	.9133	.9070
Max. Dimension	.8654	–	.8654	.8525	.9154	.9092
N. Classes – 1	.8106	.8871	.8587	.6588	.9044	.8872

see a more centered concentration of points, that is slightly weighted towards the LMNN region. In the comparisons made between the DML algorithms we observe the points weighted to the [NCA, rope] segment, which concludes the difficulty of outperforming NCA, and between DMLMJ and LMNN we can see a pretty level playing field that is slightly biased to the DMLMJ algorithm.

The outperforming of Euclidean distance is even clearer in the results from experiment 2. For these algorithms, we can clearly observe that the points are concentrated in the region corresponding to the nearest centroid metric learning algorithm, as shown in Fig. 7. We have elaborated more pairwise Bayesian sign tests for the rest of the algorithms in experiment 1. The results of these tests can also be found on the pyDML-Stats website[4].

5. Prospects and challenges in distance metric learning

Throughout this tutorial we have seen what DML consists of and how it has traditionally been applied in machine learning. However, the development of technology in recent years has given rise to new problems that cannot be adequately addressed from the point of view of classic machine learning. In the same way, this technological development has led to the creation new tools that are very useful when facing new problems, as well as allowing better results to be obtained with the more traditional problems.

Focusing on DML, both the new problems and the new tools are generating new prospects where in which applying DML could be of interest, and as well as generating new challenges in the design and application of DML. Below we will describe some of the most outstanding ones.

5.1. Prospects of Distance metric learning in machine learning

Nowadays there are many fields where the further development of DML might be of interest. On the one hand, the large volumes of data that are usually being handled today make it necessary to adapt or design new algorithms that can work properly with both high-dimensional data and huge amounts of examples. Similarly, new problems are arising, which make it necessary to reconsider the algorithms so that they can handle these problems in an appropriate way. On the other hand, many of the tools provided by machine learning, from the classical ones to the most modern ones, can be used in line with DML to achieve better results. We outline these prospects below.

- **Hybridization with feature selection techniques to solve high dimensional data problems.** DML is of great interest in many real problems in high dimensionality, such as face recognition, where it is very useful to be able to measure the similarity between different images [30]. When we work with datasets of even greater dimensionality, the treatment of distances can become too expensive, since it would be necessary to store matrices of very large dimensions. In these situations, it may be of interest to combine DML with feature selection techniques prepared for very high dimensional data [77,26].
- **Big Data solutions.** The problem of learning when the amount of data we have is huge and heterogeneous is one of the challenges of machine learning nowadays [78]. In the case of the DML algorithms, although many of them, especially those based on gradient descent, are quite slow and do not scale well with the number of samples, they can be largely parallelized in both matrix computations and gradient descent batches. As a result, DML can be extended to handle Big Data by developing specialized algorithms and integrating them with frameworks such as Spark [79] and Cloud Computing architectures [80].
- **Application of distance metric learning to singular supervised learning problems.** In this paper, we have focused on DML for common problems, like standard classification and dimensionality reduction, and we have also mentioned its applications for clustering and semi-supervised learning. However, DML can be useful in a wide variety of *non-standard* learning tasks [81], and can be carried out either by designing new algorithms or by adapting known algorithms from standard problems to these tasks. In recent years, several DML proposals have been made in problems like regression [82], multi-dimensional classification [83], ordinal classification [84], multi-output learning [85] and even transfer learning [86,13].
- **Hybridization with shallow learning techniques.** Over the years, some distance-based algorithms, or some of their ideas or foundations, have been combined with other algorithms in order to improve their learning capabilities in certain problems. For example, the concept of nearest-neighbors has been com-

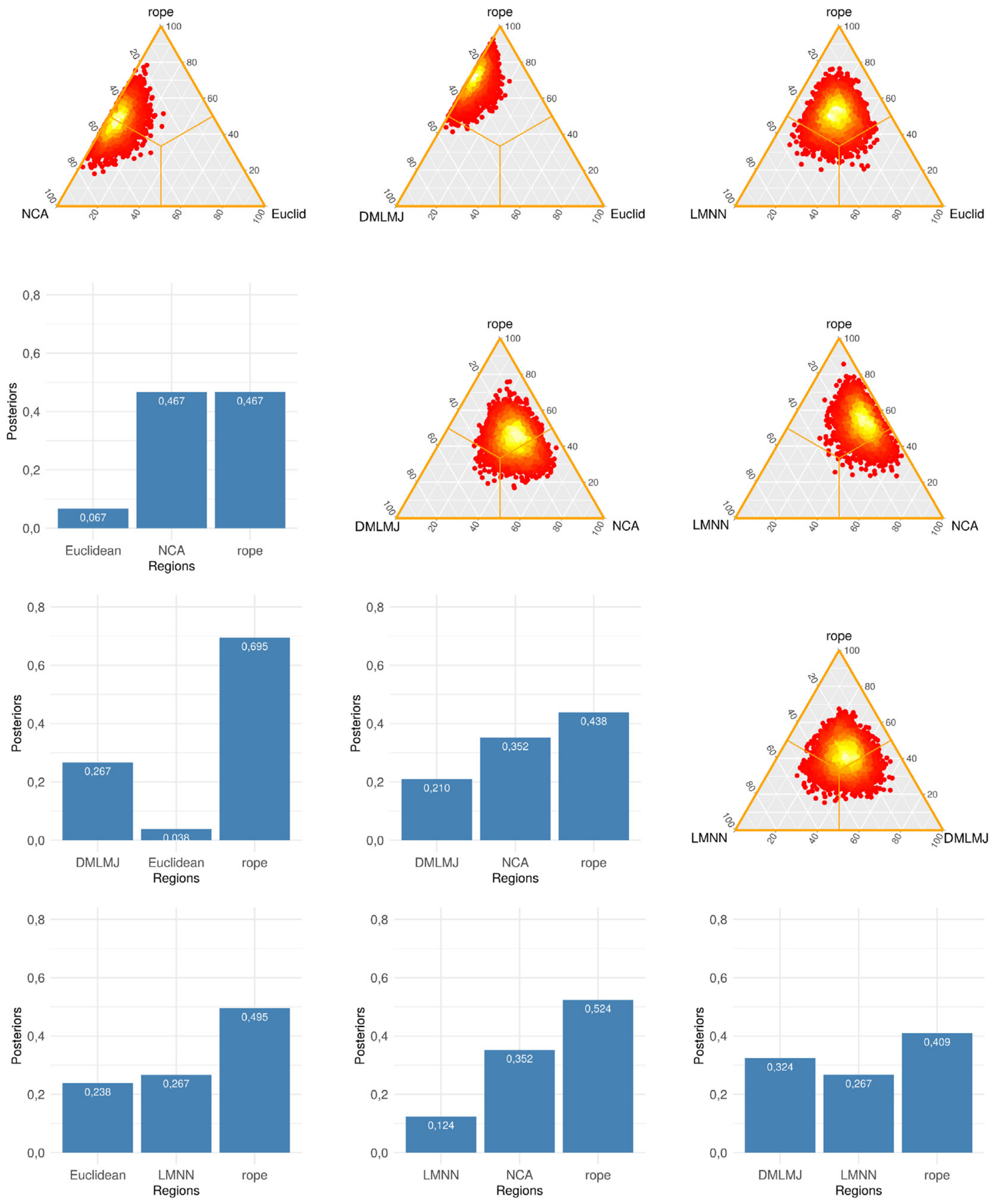


Fig. 6. Bayesian sign results for NCA, DMLMJ, LMNN and Euclidean distance with 3-NN.

bined with classifiers such as Naive-Bayes, obtaining a Naive-Bayes classifier whose feature distributions are determined by the nearest neighbors of each class [87]; with neural networks, to find the best neural network architecture [88]; with random forests, by exploiting the relationship between voting points and potential nearest neighbors [89]; with ensemble methods, like bootstrap [90,91]; with support vector machines, training them locally in neighborhoods [92]; or with rule-learning algorithms, obtaining the so-called *nested generalized exemplar* algorithms [93]. The distances used in these combinations of algorithms can condition their performance, so designing appropriate distance learning algorithms for each of these tasks can help achieve good results. Staying on this subject, another option is to hybridize directly DML with other techniques, like ensemble learning [94].

- **Hybridization with deep learning techniques.** In recent years, machine learning has experienced great popularity thanks to the development of deep learning, which is capable of obtaining very good results in different learning problems [95]. As in the previous case, it is possible to combine distance-based algorithms or their foundations with deep learning techniques to improve their learning capabilities. For instance, Papernot and McDaniel [96] use the k-nearest neighbors classifier to provide interpretability and robustness to deep neural networks. Another prospect that has gained popularity in recent years is based on the use of neural networks to learn distances, which is being referred to as *deep metric learning* [97–101]. Deep learning is likely to play an important role in the future of machine learning, and thus its combination with DML may lead to interesting advances in both fields.

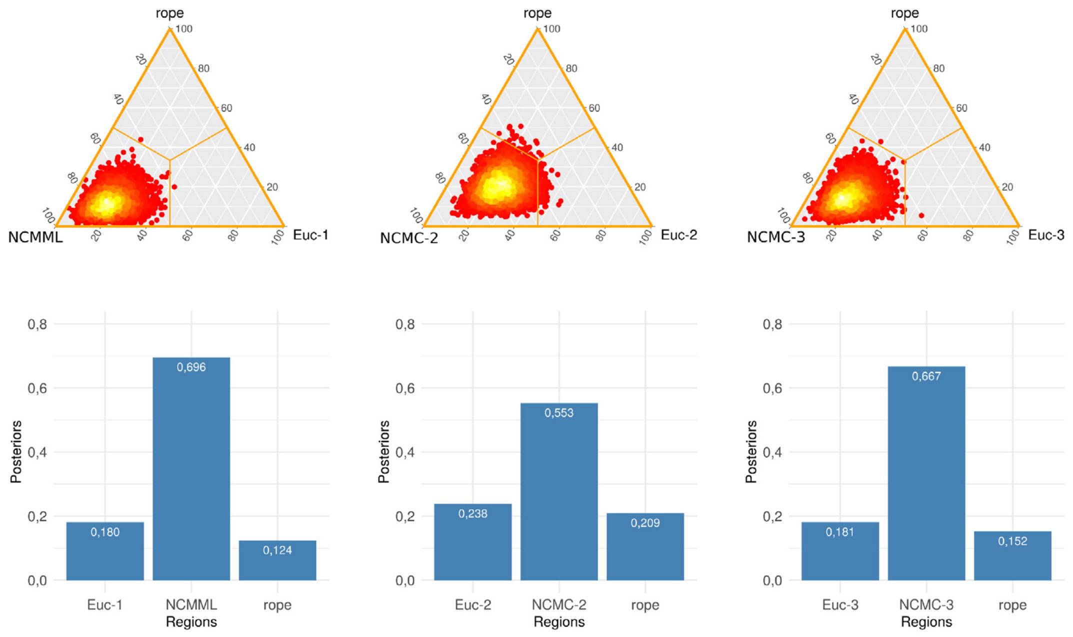


Fig. 7. Bayesian sign test results for the comparison between scores of nearest centroid classifiers with their corresponding DML algorithm against the same classifier with Euclidean distance. The results are shown for nearest class mean classifier (left), nearest class with 2 centroids (center) and nearest class with 3 centroids (right).

- **Other approaches for the concept of distance.** Most of the current DML theory focus on Mahalanobis distances. However, some articles open a door to learning about other possible distances, such as local Mahalanobis distances, that lead to a multi-metric learning [38], or approaches beyond the Mahalanobis theory [102,103]. The *deep metric learning* approach discussed above is another way of handling a wider range of distances. By developing new approaches, we will have a greater variety of distances to learn, and thus have a greater chance of success.

5.2. Challenges in distance metric learning

In addition to the numerous action horizons, DML presents several challenges in terms of the design of its algorithms, which can lead to substantial improvements. We describe these challenges below.

- **Non-linear distance metric learning.** As we have already mentioned, since learning a Mahalanobis distance is equivalent to learning a linear map, there are many problems where these distances are not able to capture the inherent non-linearity of the data. Although the non-linearity of a subsequent learning algorithm, such as the nearest neighbors classifier, may mitigate this fact, that algorithm could benefit much more from a distance capable of capturing the non-linearity of the dataset. In this sense, we have already seen how the kernelization of DML algorithms can be applied to fit non-linear data. Extending the kernel trick to other algorithms besides those presented, by searching for suitable parameterizations and representer theorems, is another possible task to carry out. Another possibility for non-linear DML is to adapt classical objective functions so that they can work with non-linear distances, such as the χ^2 histogram distance, or with non-linear transformations of the data learned by another algorithm, such as gradient boosting [104].
- **Multi-linear distance metric learning.** In learning problems where the data are images or videos, the traditional vector representation may not be the most appropriate to fit the data properly. Vector representation does not allow, for example, for the consideration of neighborhood relationships between pixels in an image. It is therefore better to consider images as matrices, or more generally, as multi-linear mappings or *tensors*. Some DML algorithms can be adapted so that they can learn distances in tensor spaces [105,36,106], which will be more suitable for similarity learning in datasets that support this representation. The development or extension of techniques for multi-linear DML is a challenge that has many applications in a field such as computer vision, where DML has been shown to be quite useful [17,19,21,68].
- **Other optimization mechanisms.** The algorithms we have studied optimize their objective functions by applying gradient descent methods. However, the possibilities in terms of optimization mechanisms are very broad, and choosing the most appropriate method can contribute to achieving better values in the objective functions. In addition, the consideration of different optimization methods may lead to the design of new objective functions that may be appropriate for new problems or approaches and that cannot be optimized by classical gradient methods. In this way, we have studied several differentiable objective functions in this tutorial, unconstrained or with convex constraints, but for those non-convex functions the gradient descent methods (even the stochastic version) cannot guarantee a convergence to the global optimum. If we wanted to consider functions with even worse analytical characteristics or constraints, such as non-differentiability or integer constraints, we could not even use this type of method. For the non-convex and differentiable case, we are still able to use the information of the derivatives of the objective function, and some refinements of the classic gradient methods, such as AdaDelta, RMSprop or Adam have shown good performance in this type of problem [107]. In the most general case, we are only able to afford to evaluate the objective function, and sometimes not a very high number of times, due to its complexity. This general case is usually called *black-box optimization*. To optimize these functions, a wide variety of proposals have been made. If we cannot afford to evaluate the objective function many times,

Bayesian optimization may be an interesting alternative [108]. If the objective function is not so complex, evolutionary algorithms can provide us with a great capability of exploration in the search space. Their repertoire is much broader and includes techniques such as *simulated annealing*, *particle swarm optimization* or *response surface methods*, among others [109], thus many tools are available to address the most diverse optimization problems. These heuristics can also be used over differentiable optimization problems, and sometimes they can even outperform gradient methods, thanks to their greater ability to escape from local optima [110]. The evolutionary approach has been explored recently in several DML problems [111,112].

6. Conclusions

In this tutorial we have studied the concept of distance metric learning, showing what it is, what its applications are, how to design its algorithms, and the theoretical foundations of this discipline. We have also studied some of the most popular algorithms in this field and their theoretical foundations, and explained different resolution techniques.

In order to understand the theoretical foundations of distance metric learning and its algorithms, it was necessary to delve into three different mathematical theories: convex analysis, matrix analysis and information theory. Convex analysis has it possible to present many of the optimization problems studied in the algorithms, along with some methods for solving them. Matrix analysis provided many useful tools to help understand this discipline, from how to parameterize Mahalanobis distances, to the optimization with eigenvectors, going through the most basic algorithm of semidefinite programming. Finally, information theory has motivated several of the algorithms we have studied.

In addition, several experiments have been developed that have allowed for the evaluation of the performance of the algorithms analyzed in this study. The results of these experiments have allowed us to observe how algorithms such as LMNN, DMLMJ, and especially NCA can considerably improve the nearest neighbors classification, and how centroid-based distance learning algorithms also improve their corresponding classifiers. We have also seen the wide variety of possibilities offered by kernel-based algorithms, and the advantages that an appropriate reduction of the dimensionality of the datasets can offer.

CRediT authorship contribution statement

Juan Luis Suárez: Conceptualization, Methodology, Software, Investigation, Writing - review & editing, Validation. **Salvador García:** Conceptualization, Methodology, Writing - review & editing, Investigation, Supervision. **Francisco Herrera:** Funding acquisition, Project administration, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Our work has been supported by the research project TIN2017-89517-P and by a research scholarship (FPU18/05989), given to the author Juan Luis Suárez by the Spanish Ministry of Science, Innovation and Universities.

Appendix A. Glossary of terms

ANMM	Average Neighborhood Margin Maximization
DML	Distance Metric Learning
DML-eig	Distance Metric Learning with eigenvalue optimization
DMLMJ	Distance Metric Learning through the Maximization of the Jeffrey divergence
ITML	Information Theoretic Metric Learning
KANMM	Kernel Average Neighborhood Margin Maximization
KDA	Kernel Discriminant Analysis
KDMLMJ	Kernel Distance Metric Learning through the Maximization of the Jeffrey divergence
KLMNN	Kernel Large Margin Nearest Neighbors
k-NN	k-Nearest Neighbors
LDA	Linear Discriminant Analysis
LDML	Logistic Discriminant Metric Learning
LMNN	Large Margin Nearest Neighbors
LSI	Learning with Side Information
MCML	Maximally Collapsing Metric Learning
MMC	Mahalanobis Metric for Clustering
NCA	Neighborhood Components Analysis
NCM	Nearest Class Mean
NCMC	Nearest Class with Multiple Centroids
NCMML	Nearest Class Mean Metric Learning
PCA	Principal Components Analysis
SVM	Support Vector Machines

References

[1] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1967) 21–27.

[2] G.S. Sebestyen, Decision-Making Processes in Pattern Recognition, Macmillan, 1962.

[3] N.J. Nilsson, Learning Machines: Foundations of Trainable Pattern-Classifying Systems, McGraw-Hill, 1965.

[4] E.P. Xing, M.I. Jordan, S.J. Russell, A.Y. Ng, Distance metric learning with application to clustering with side-information, in: Advances in Neural Information Processing Systems, 2003, pp. 521–528.

[5] Z. Ma, S. Zhou, X. Wu, H. Zhang, W. Yan, S. Sun, J. Zhou, Nasopharyngeal carcinoma segmentation based on enhanced convolutional neural networks using multi-modal metric learning, Physics in Medicine & Biology 64 (2019) 025005.

[6] G. Wei, M. Qiu, K. Zhang, M. Li, D. Wei, Y. Li, P. Liu, H. Cao, M. Xing, F. Yang, A multi-feature image retrieval scheme for pulmonary nodule diagnosis, Medicine 99 (2020).

[7] T. Li, G. Kou, Y. Peng, Improving malicious urls detection via feature engineering: Linear and nonlinear space transformation methods, Information Systems 101494 (2020).

[8] Y. Luo, H. Hu, Y. Wen, D. Tao, Transforming device fingerprinting for wireless security via online multitask metric learning, IEEE Internet of Things Journal 7 (2020) 208–219.

[9] Y. Liu, D. Pi, L. Cui, Metric learning combining with boosting for user distance measure in multiple social networks, IEEE Access 5 (2017) 19342–19351.

[10] Y. Liu, Z. Gu, T.H. Ko, J. Liu, Multi-modal media retrieval via distance metric learning for potential customer discovery, in: 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), IEEE, 2019, pp. 310–317.

[11] R. Li, J.-Y. Jiang, J.L. Li, C.-C. Hsieh, W. Wang, Automatic speaker recognition with limited data, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 340–348.

[12] Z. Bai, X.-L. Zhang, J. Chen, Speaker verification by partial auc optimization with mahalanobis distance metric learning, IEEE/ACM Transactions on Audio, Speech, and Language Processing (2020).

[13] D. Lopez-Sanchez, A.G. Arrieta, J.M. Corchado, Visual content-based web page categorization with deep transfer learning and metric learning, Neurocomputing 338 (2019) 418–431.

[14] H. Hu, K. Wang, C. Lv, J. Wu, Z. Yang, Semi-supervised metric learning-based anchor graph hashing for large-scale image retrieval, IEEE Transactions on Image Processing 28 (2019) 739–754.

[15] H. Wu, Q. Zhou, R. Nie, J. Cao, Effective metric learning with co-occurrence embedding for collaborative recommendations, Neural Networks 124 (2020) 308–318.
[16] X. Li, Y. Tang, A social recommendation based on metric learning and network embedding, in: 2020 IEEE 5th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA), IEEE, 2020, pp. 55–60.
[17] B. Nguyen, B. De Baets, Kernel distance metric learning using pairwise constraints for person re-identification, IEEE Transactions on Image Processing 28 (2019) 589–600.
[18] C. Zhao, X. Wang, W. Zuo, F. Shen, L. Shao, D. Miao, Similarity learning with joint transfer constraints for person re-identification, Pattern Recognition 97 (2020).
[19] J. Liang, Q. Hu, C. Dang, W. Zuo, Weighted graph embedding-based metric learning for kinship verification, IEEE Transactions on Image Processing 28 (2019) 1149–1162.
[20] F. Dornaika, I. Arganda-Carreras, O. Serradilla, Transfer learning and feature fusion for kinship verification, Neural Computing and Applications 32 (2020) 7139–7151.
[21] D. Wang, Y. Cheng, M. Yu, X. Guo, T. Zhang, A hybrid approach with optimization-based and metric-based meta-learner for few-shot learning, Neurocomputing 349 (2019) 202–211.
[22] C. Wang, G. Peng, B. De Baets, Deep feature fusion through adaptive discriminative metric learning for scene recognition, Information Fusion (2020).
[23] Y. Du, C. Liu, B. Zhang, Detection of gh pituitary tumors based on mnf, in: 2019 Chinese Control And Decision Conference (CCDC), IEEE, 2019, pp. 635–639.
[24] J.R. Wells, S. Aryal, K.M. Ting, Simple supervised dissimilarity measure: Bolstering iforest-induced similarity with class information without learning, Knowledge and Information Systems (2020) 1–14.
[25] B. Nguyen, C. Morell, B. De Baets, Scalable large-margin distance metric learning using stochastic gradient descent, IEEE Transactions on Cybernetics 50 (2020) 1072–1083.
[26] K. Liu, A. Bellet, Escaping the curse of dimensionality in similarity learning: Efficient frank-wolfe algorithm and generalization bounds, Neurocomputing 333 (2019) 185–199.
[27] L. Yang, R. Jin, Distance metric learning: A comprehensive survey, Michigan State University 2 (2006) 4.
[28] B. Kulis et al., Metric learning: A survey, foundations and trends in machine, Learning 5 (2013) 287–364.
[29] A. Bellet, A. Habrard, M. Sebban, A survey on metric learning for feature vectors and structured data, 2013, arXiv preprint arXiv:1306.6709.
[30] P. Moutafis, M. Leng, I.A. Kakadiaris, An overview and empirical comparison of distance metric learning methods, IEEE Transactions on Cybernetics 47 (2017) 612–625.
[31] R.T. Rockafellar, Convex Analysis, Princeton University Press, 2015.
[32] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[33] R.A. Horn, C.R. Johnson, Matrix Analysis, Cambridge University Press, 1990.
[34] T.M. Cover, J.A. Thomas, Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing), Wiley-Interscience, 2006.
[35] R.A. Fisher, The use of multiple measurements in taxonomic problems, Annals of Eugenics 7 (1936) 179–188.
[36] F. Wang, C. Zhang, Feature extraction by maximizing the average neighborhood margin, in, 2007 IEEE Conference on Computer Vision and Pattern Recognition (2007) 1–8.
[37] J.P. Cunningham, Z. Ghahramani, Linear dimensionality reduction: Survey, insights, and generalizations, Journal of Machine Learning Research 16 (2015) 2859–2900.
[38] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, Journal of Machine Learning Research 10 (2009) 207–244.
[39] J. Goldberger, G.E. Hinton, S.T. Roweis, R.R. Salakhutdinov, Neighbourhood components analysis, in: Advances in neural information processing systems, 2005, pp. 513–520.
[40] T. Mensink, J. Verbeek, F. Perronnin, G. Csurka, Distance-based image classification: Generalizing to new classes at near-zero cost, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 2624–2637.
[41] J.V. Davis, B. Kulis, P. Jain, S. Sra, I.S. Dhillon, Information-theoretic metric learning, in: Proceedings of the 24th International Conference on Machine Learning, ACM, 2007, pp. 209–216.
[42] B. Nguyen, C. Morell, B. De Baets, Supervised distance metric learning through maximization of the jeffrey divergence, Pattern Recognition 64 (2017) 215–225.
[43] A. Globerson, S.T. Roweis, Metric learning by collapsing classes, in: Advances in Neural Information Processing Systems, 2006, pp. 451–458.
[44] L. Torresani, K.-C. Lee, Large margin component analysis, in: Advances in Neural Information Processing Systems, 2007, pp. 1385–1392.
[45] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, K.-R. Mullers, Fisher discriminant analysis with kernels, in: Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop., IEEE, 1999, pp. 41–48.
[46] A. Benavoli, G. Corani, J. Demšar, M. Zaffalon, Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis, Journal of Machine Learning Research 18 (2017) 2653–2688.
[47] J.L. Suárez, S. García, F. Herrera, pydml: A python library for distance metric learning, Journal of Machine Learning Research 21 (2020) 1–7.
[48] J.L. Suárez, S. García, F. Herrera, A tutorial on distance metric learning: Mathematical foundations, algorithms, experiments, prospects and challenges (with appendices on mathematical background and detailed algorithms explanation), 2020, arXiv preprint arXiv:1812.05944.
[49] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, Oakland, CA, USA, 1967, pp. 281–297.
[50] Y. Dou, H. Yang, X. Deng, A survey of collaborative filtering algorithms for social recommender systems, in: 2016 12th International Conference on Semantics, Knowledge and Grids (SKG), IEEE, 2016, pp. 40–46.
[51] X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation, Technical Report, Carnegie Mellon University, 2002.
[52] T. Hofmann, B. Schölkopf, A.J. Smola, Kernel methods in machine learning, The Annals of Statistics (2008) 1171–1220.
[53] A. Ahmad, L. Dey, A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set, Pattern Recognition Letters 28 (2007) 110–118.
[54] D.B. Blumenthal, J. Gamper, On the exact computation of the graph edit distance, Pattern Recognition Letters 134 (2020) 46–57.
[55] M. Norouzi, D.J. Fleet, R.R. Salakhutdinov, Hamming distance metric learning, in: Advances in Neural Information Processing Systems, 2012, pp. 1061–1069.
[56] L. Ma, H. Li, F. Meng, Q. Wu, K.N. Ngan, Discriminative deep metric learning for asymmetric discrete hashing, Neurocomputing 380 (2020) 115–124.
[57] A. Zheng, A. Casari, Feature engineering for machine learning: principles and techniques for data scientists, O'Reilly Media Inc, 2018.
[58] Ö. Yeniay, Penalty function methods for constrained optimization with genetic algorithms, Mathematical and Computational Applications 10 (2005) 45–56.
[59] T. Yang, Q. Lin, L. Zhang, A richer theory of convex constrained optimization with reduced projections and improved rates, in: International Conference on Machine Learning, 2017, pp. 3901–3910.
[60] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.
[61] E. Kokiopoulou, J. Chen, Y. Saad, Trace optimization and eigenproblems in dimension reduction methods, Numerical Linear Algebra with Applications 18 (2011) 565–602.
[62] C.C. Aggarwal, Y. Zhao, S.Y. Philip, On text clustering with side information, Proceedings – International Conference on Data Engineering, IEEE (2012) 894–904.
[63] P.S. Bradley, U.M. Fayyad, Refining initial points for k-means clustering, International Conference on Machine Learning, vol. 98, Citeseer (1998) 91–99.
[64] P. Probst, A.-L. Boulesteix, B. Bischl, Tunability: Importance of hyperparameters of machine learning algorithms, Journal of Machine Learning Research 20 (2019) 1–32.
[65] P.S. Dhillon, P.P. Talukdar, K. Crammer, Inference-driven metric learning for graph construction, in: 4th North East Student Colloquium on Artificial Intelligence, 2010.
[66] I. Jolliffe, Principal Component Analysis, Springer Series in Statistics, Springer, 2002.
[67] Y. Ying, P. Li, Distance metric learning with eigenvalue optimization, Journal of Machine Learning Research 13 (2012) 1–26.
[68] M. Guillaumin, J. Verbeek, C. Schmid, Is that you? Metric learning approaches for face identification, in: in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 498–505.
[69] M.L. Overton, On minimizing the maximum eigenvalue of a symmetric matrix, SIAM Journal on Matrix Analysis and Applications 9 (1988) 256–268.
[70] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (1998) 121–167.
[71] R. Chatpatanasiri, T. Korsrilabutr, P. Tangchanachaianan, B. Kijsirikul, A new kernelization framework for mahalanobis distance learning algorithms, Neurocomputing 73 (2010) 1570–1579.
[72] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (1998) 1299–1319.
[73] I. Triguero, S. González, J.M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M.J. del Jesús, L. Sánchez, F. Herrera, Keel 3.0: an open source software for multi-stage analysis in data mining, International Journal of Computational Intelligence Systems 10 (2017) 1238–1249.
[74] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82.
[75] A. Benavoli, G. Corani, F. Mangili, M. Zaffalon, F. Ruggeri, A bayesian wilcoxon signed-rank test based on the dirichlet process, in: International Conference on Machine Learning, 2014, pp. 1026–1034.
[76] J. Carrasco, S. García, M. del Mar Rueda, F. Herrera, rnpbst: An r package covering non-parametric and bayesian statistical tests, in: International Conference on Hybrid Artificial Intelligence Systems, Springer, 2017, pp. 281–292.
[77] M. Tan, I.W. Tsang, L. Wang, Towards ultrahigh dimensional feature selection for big data, Journal of Machine Learning Research 15 (2014) 1371–1429.
[78] X. Wu, X. Zhu, G.-Q. Wu, W. Ding, Data mining with big data, IEEE Tansactions on Knowledge and Data Engineering 26 (2014) 97–107.

[79] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., Mllib: Machine learning in apache spark, Journal of Machine Learning Research 17 (2016) 1235–1241.
[80] I.A.T. Hashem, I. Yaqoob, N.B. Anuar, S. Mokhtar, A. Gani, S.U. Khan, The rise of "big data" on cloud computing: Review and open research issues, Information Systems 47 (2015) 98–115.
[81] D. Charte, F. Charte, S. García, F. Herrera, A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations, Progress in Artificial Intelligence 8 (2019) 1–14.
[82] B. Nguyen, C. Morell, B. De Baets, Large-scale distance metric learning for k-nearest neighbors regression, Neurocomputing 214 (2016) 805–814.
[83] Z. Ma, S. Chen, Multi-dimensional classification via a metric approach, Neurocomputing 275 (2018) 1121–1131.
[84] B. Nguyen, C. Morell, B. De Baets, Distance metric learning for ordinal classification based on triplet constraints, Knowledge-Based Systems 142 (2018) 17–28.
[85] W. Liu, D. Xu, I. Tsang, W. Zhang, Metric learning for multi-output tasks, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (2019) 408–422.
[86] Y. Luo, Y. Wen, T. Liu, D. Tao, Transferring knowledge fragments for learning distance metric from a heterogeneous domain, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (2019) 1013–1026.
[87] X. Yang, Y.L. Tian, Eigenjoints-based action recognition using naive-bayes-nearest-neighbor, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, IEEE, 2012, pp. 14–19.
[88] L. Wang, B. Yang, Y. Chen, X. Zhang, J. Orchard, Improving neural-network classifiers using nearest neighbor partitioning, IEEE Transactions on Neural Networks and Learning Systems 28 (2017) 2255–2267.
[89] Y. Lin, Y. Jeon, Random forests and adaptive nearest neighbors, Journal of the American Statistical Association 101 (2006) 578–590.
[90] B.M. Steele, Exact bootstrap k-nearest neighbor learners, Machine Learning 74 (2009) 235–255.
[91] Y. Hamamoto, S. Uchimura, S. Tomita, A bootstrap technique for nearest neighbor classifier design, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (1997) 73–79.
[92] H. Zhang, A.C. Berg, M. Maire, J. Malik, Svm-knn: Discriminative nearest neighbor classification for visual category recognition, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, IEEE, 2006, pp. 2126–2136.
[93] D. Wettschereck, T.G. Dietterich, An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms, Machine Learning 19 (1995) 5–27.
[94] Y. Mu, W. Ding, D. Tao, Local discriminative distance metrics ensemble learning, Pattern Recognition 46 (2013) 2337–2349.
[95] A. Gómez-Ríos, S. Tabik, J. Luengo, A. Shihavuddin, B. Krawczyk, F. Herrera, Towards highly accurate coral texture images classification using deep convolutional neural networks and data augmentation, Expert Systems with Applications 118 (2019) 315–328.
[96] N. Papernot, P. McDaniel, Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning, arXiv preprint arXiv:1803.04765 (2018).
[97] D. Yi, Z. Lei, S. Liao, S.Z. Li, Deep metric learning for person re-identification, in: 2014 22nd International Conference on Pattern Recognition, IEEE, 2014, pp. 34–39.
[98] X. Zhe, S. Chen, H. Yan, Directional statistics-based deep metric learning for image classification and retrieval, Pattern Recognition 93 (2019) 113–123.
[99] F. Cakir, K. He, X. Xia, B. Kulis, S. Sclaroff, Deep metric learning to rank, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1861–1870.
[100] X. Cao, Y. Ge, R. Li, J. Zhao, L. Jiao, Hyperspectral imagery classification with deep metric learning, Neurocomputing 356 (2019) 217–227.
[101] B. Nguyen, B. De Baets, Improved deep embedding learning based on stochastic symmetric triplet loss and local sampling, Neurocomputing 402 (2020) 209–219.
[102] J. Pan, H. Le Capitaine, Metric learning with submodular functions, Neurocomputing (2020).
[103] H. Shindo, M. Nishino, Y. Kobayashi, A. Yamamoto, Metric learning for ordered labeled trees with pq -grams, in: 24th European Conference of Artificial Intelligence, 2020.
[104] D. Kedem, S. Tyree, F. Sha, G.R. Lanckriet, K.Q. Weinberger, Non-linear metric learning, in: Advances in Neural Information Processing Systems, 2012, pp. 2573–2581.
[105] D. Cai, X. He, J. Han, D. Cai, X. He, J. Han, Subspace learning based on tensor analysis, Technical Report (2005).
[106] O. Laiadi, A. Ouamane, A. Benakcha, A. Taleb-Ahmed, A. Hadid, Tensor cross-view quadratic discriminant analysis for kinship verification in the wild, Neurocomputing 377 (2020) 286–300.
[107] S. Sun, Z. Cao, H. Zhu, J. Zhao, A survey of optimization methods from a machine learning perspective, IEEE Transactions on Cybernetics (2019).
[108] P.I. Frazier, A tutorial on bayesian optimization, arXiv preprint arXiv:1807.02811 (2018).
[109] L.M. Rios, N.V. Sahinidis, Derivative-free optimization: a review of algorithms and comparison of software implementations, Journal of Global Optimization 56 (2013) 1247–1293.
[110] G. Morse, K.O. Stanley, Simple evolutionary optimization can rival stochastic gradient descent in neural networks, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2016, 2016,, pp. 477–484.
[111] W. Kalintha, S. Ono, M. Numao, K.-I. Fukui, Kernelized evolutionary distance metric learning for semi-supervised clustering., in: 31st AAAI Conference on Artificial Intelligence, 2017, pp. 4945–4946.
[112] B. Ali, W. Kalintha, K. Moriyama, M. Numao, K.-I. Fukui, Reinforcement learning for evolutionary distance metric learning systems improvement, in, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, 2018, pp. 155–156.

Juan Luis Suárez received the B.Sc. degrees in Mathematics and Computer Science and the M.Sc. degree in Computer Science from the University of Granada, Granada, Spain, in 2018 and 2019, respectively. He is currently a Ph.D. student in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. His research interests include data mining, data preprocessing, no standard classification and distance metric learning.

Salvador García received the B.S. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively. He is currently a Full Professor in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. Dr. Garc?a has published more than 100 papers in international journals (more than 70 in Q1), H-index 51. As edited activities, he is the associate editor in chief of "Information Fusion" (Elsevier), and an associate editor of "Swarm and Evolutionary Computation" (Elsevier) and "AI Communications" (IOS Press) journals. He is a co-author of the books entitled "Data Preprocessing in Data Mining", "Learning from Imbalanced Data Sets" and "Big Data Preprocessing: Enabling Smart Data" published by Springer. His research interests include Data Science, data preprocessing, Big Data, evolutionary learning, Deep Learning, metaheuristics and biometrics. He belongs to the list of the Highly Cited Researchers in the area of Computer Sciences (2014–2019):http://highlycited.com/ (Clarivate Analytics).

Francisco Herrera (SM'15) received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada and Director of the Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI). He's an academician in the Royal Academy of Engineering (Spain). He has been the supervisor of 51 Ph.D. students. He has published more than 500 journal papers, receiving more than 86000 citations (Scholar Google, H-index 144). He has been nominated as a Highly Cited Researcher (in the fields of Computer Science and Engineering, respectively, 2014 to present, Clarivate Analytics). He currently acts as Editor in Chief of the international journal "Information Fusion" (Elsevier). He acts as editorial member of a dozen of journals. He received the several honors and awards, among others: ECCAI Fellow 2009, IFSA Fellow 2013, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 and 2012 Papers, 2011 Lotfi A. Zadeh Prize Best paper Award (IFSA Association), 2013 AEPIA Award to a scientific career in Artificial Intelligence, 2014 XV Andalucía Research Prize Maimónides, 2017 Andalucía Medal (by the regional government of Andaluc?a), 2018 "Granada: Science and Innovation City" award. His current research interests include among others, Computational Intelligence (including fuzzy modeling, computing with words, evolutionary algorithms and deep learning), information fusion and decision making, and data science (including data preprocessing, prediction, non-standard classification problems, and big data).