Pattern Recognition Letters 100 (2017) 37–43

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Distance metric learning with the Universum

Bac Nguyen [a,*], Carlos Morell [b], Bernard De Baets [a]

[a] KERMIT, Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Coupure links 653, Ghent, 9000, Belgium
[b] Computer Science Department, Universidad Central de Las Villas, Santa Clara, Villa Clara, 54830, Cuba

ARTICLE INFO

Article history:
Received 15 December 2016
Available online 25 September 2017

MSC:
47N10

Keywords:
Metric learning
Universum learning
Nearest neighbor

ABSTRACT

Universum, a set of examples that do not belong to any class of interest for a classification problem, has been playing an important role in improving the performance of many machine learning methods. Since Universum examples are not required to have the same distribution as the training data, they can contain prior information for the possible classifiers. In this paper, we propose a novel distance metric learning method for nearest-neighbor (NN) classification, namely $\mathfrak{U}$-LMNN, that exploits prior information contained in the available Universum examples. Based on the large-margin nearest neighbor (LMNN) method, $\mathfrak{U}$-LMNN maximizes, for each training example, the margin between its nearest neighbor of the same class and the neighbors of different classes, while controlling the generalization capacity through the number of contradictions on Universum examples. Experimental results on synthetic as well as real-world data sets demonstrate a good performance of $\mathfrak{U}$-LMNN compared to the conventional LMNN method.

1. Introduction

Many machine learning applications involve high-dimensional data, where the number of input features is larger than the number of training examples, such as genomics [1], medical imaging [2], and document classification [3]. It is well known that classical machine learning methods can suffer from the "curse of dimensionality" on these high-dimensional settings and lead to an under-determined problem [4]. As a result, they may not perform well on unseen data. For instance, support vector machines (SVMs) [5] with linear kernels yield a very poor generalization due to the geometric properties of high-dimensional data: kernel matrices are close to a diagonal matrix, which is generally useless to learn [6].

One possible solution to this problem is to integrate more information into the training data [7]. In this paper, we are particularly interested in using Universum examples [8] as additional information to improve the classification performance. To be more specific, the Universum examples are collected from the same domain as the labeled data, but they do not belong to any class of interest. Instead of specifying information about the prior distribution of training examples, which is usually hard, we use the Universum examples, which also contain prior information related to the possible training and test examples [5,7]. For instance, in the classification of the handwritten digits 5 versus 8, we can introduce the additional information by including other handwritten digits that belong to any of the other classes in the database, e.g., images of digits 0, 1, 2, 3, 4, 6, 7, or 9. Those digits define the style of the handwritten digits problem, which are geometrically belonging to the same input space to which the training examples belong. The idea is to add prior information contained in examples from classes that are significantly different from the classes being separated. We will discuss this further in Section 2.

In recent years, researchers have realized the importance of employing an appropriate distance metric for distance-based learning methods such as nearest-neighbor (NN) classification [9]. Despite its simplicity, the NN classifier is well suited for multi-class classification problems and obtains competitive results compared to other learning methods. Another interesting property of NN is that its expected error is bounded by twice the Bayes error. Empirical evidence reported in the literature shows that the performance of NN depends crucially on the way of computing the closeness between examples [10–13]. A good distance metric mostly depends on the application domain and should yield small distances between examples of the same class and large distances between those of different classes. One of the most widely used distance metric learning methods for NN classification is the large-margin nearest neighbor (LMNN) proposed by Weinberger and Saul [14]. LMNN employs the principle of large margin to reduce the expected error bound for NN classification. Several aspects of LMNN have been discussed in the literature. For instance, Do et al. [15] highlighted a relation between LMNN and SVM in a quadratic

* Corresponding author.
E-mail address: bac.nguyencong@ugent.be (B. Nguyen).

https://doi.org/10.1016/j.patrec.2017.09.031
0167-8655/

space, where LMNN can be seen as a set of local SVMs. Yin and Li [11] showed that LMNN can be formulated as an eigenvalue optimization problem. In practice, LMNN has shown a good performance for several classification tasks, however, it can suffer from the overfitting problem caused by the lack of regularization.

Although the expected or generalization error of distance metric learning methods has been improved by employing the large-margin criterion, according to statistical learning theory [5], this kind of generalization error bound usually requires a large number of training examples to achieve a reasonable classification accuracy. Unfortunately, in the real world, collecting data is not always a quick and easy task. For this reason, we introduce the use of Universum examples for learning a distance metric because it is easier to collect Universum data for different kinds of problems. When the information of other classes (which are different from the classes of interest) is not available, Universum examples can be simply generated, for instance, by randomly selecting a pair of examples of different classes of interest and computing their average. This strategy is referred to as *random averaging* [8]. To the best of our knowledge, this is the first attempt to learn a distance metric using the Universum.

In short, the main contributions of this paper are summarized as follows:

i) We propose a strategy for maximizing the number of contradictions on the Universum examples, which is useful to measure the complexity of an equivalence class of functions, in the context of NN classification.
ii) The proposed strategy can be efficiently implemented in the LMNN method for learning a distance metric using the prior domain knowledge contained in the Universum examples.
iii) We conduct several experiments on synthetic as well as real-world data sets to verify the usefulness of Universum data for classification problems.

The remainder of this paper is organized as follows. Section 2 introduces notations and some preliminary background in order to develop our proposal. Then, we extend the LMNN method by incorporating the use of Universum examples, which is referred to as $\mathfrak{U}$-LMNN. Section 3 discusses the computational complexity of $\mathfrak{U}$-LMNN. In Section 4, we briefly review some related works that employ the idea of learning from unlabeled examples. Section 5 presents several experiments to assess the performance of the proposed method, followed by some concluding remarks in Section 6.

2. Proposed method

In this section, we present the main idea of learning a distance metric using the Universum examples. For the sake of convenience, we first introduce some notations and background on statistical learning theory, then describe in detail the proposed method.

2.1. Preliminaries

The following notations are used throughout this paper. Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \subset \mathbb{R}^D \times \mathcal{Y}$ be the training set containing n labeled examples, where $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^D$ denotes the input points and $\mathcal{Y}$ denotes the finite set of classes. For simplicity, we will restrict our attention to two-class classification problems, i.e., $\mathcal{Y} = \{-1, 1\}$. Along with the training set $\mathcal{D}$, we also have a set of m unlabeled examples, which do not belong to either class of the training data, called the Universum set, $\mathcal{U} = \{\mathbf{x}_1^*, \ldots, \mathbf{x}_m^*\} \subset \mathbb{R}^D$. Next, we recall some results from statistical learning theory, which motivate the idea of learning from the Universum.

The goal of a learning algorithm is to find an appropriate function $f : \mathbb{R}^D \to \mathcal{Y}$, where $f \in \mathcal{H}$, a set of admissible functions, that achieves the best results in class prediction. According to Vapnik [5], with a probability at least $1 - \sigma$, the expected error of any $f \in \mathcal{H}$ is bounded by

$$R(f) \leq R_{\text{emp}}(f) + \sqrt{\frac{h\left(\log \frac{2n}{h} + 1\right) - \log \frac{\sigma}{4}}{n}}, \tag{1}$$

where R_{emp} is the empirical error over the training examples and h is the Vapnik–Chervonenkis (VC)-dimension of the function space $\mathcal{H}$, which measures the capacity of $\mathcal{H}$. Clearly, a small value of the empirical error does not necessarily imply a small value of the expected error. The generalization-error bound in (1) suggests that, disregarding the logarithmic factors, in order to achieve a good generalization performance, we need to minimize both the empirical error and the ratio between the VC-dimension and the number of training examples at the same time. In other words, it is very important to choose an appropriate value for the VC-dimension of the function space $\mathcal{H}$.

One solution to this problem is based on the Structural Risk Minimization (SRM) principle [5]. This principle tries to find an optimal relationship between the empirical error estimated by the function chosen from a set of functions and the capacity of that set of functions. Let $\Gamma_1, \ldots, \Gamma_l$ be the equivalence classes of all admissible functions in $\mathcal{H}$, where two functions belong to the same equivalence class if they separate the training examples in the same way. In doing this, we split our set containing an infinite number of admissible functions $\mathcal{H}$ into a set containing a finite number of equivalence classes Γ_i, where $i \in \{1, \ldots, l\}$. In order to perform the SRM principle, we first create a structure on these equivalence classes, which is a set of nested subsets of functions

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \ldots \subset \mathcal{H},$$

such that $h_1 \leq h_2 \leq \ldots \leq h$, where h_j is the VC-dimension of $\mathcal{H}_j$. The minimization of the right-hand side of (1) can be performed as follows: we first choose an element of the structure to control the VC-dimension, then choose a function in this element that minimizes the empirical error.

In order to build this structure, we need to associate with each equivalence class some value characterizing the capacity of the class [8]. For instance, in the case of SVM, each equivalence class is associated with the largest margin of the function belonging to this class, since a high value of the margin corresponds to a low value of the VC-dimension. Next, we will discuss the idea of computing this value through contradictions.

2.2. Maximal contradiction on Universum examples

Vapnik [7] argues that when trying to solve a problem, one should try to avoid additional generalizations. An Universum example $\mathbf{x}_i^*$ is considered as a contradiction for an equivalence class Γ_j when there exist two functions f_1 and f_2 belonging to Γ_j such that $f_1(\mathbf{x}_i^*) f_2(\mathbf{x}_i^*) < 0$. We associate to each equivalence class Γ_j the number of contradictions on Universum examples. A large number of contradictions usually implies a small capacity of the function class, which is denoted by a small value of the VC-dimension. Similar to the large-margin principle employed in SVM, we create a structure on the equivalence classes based on these numbers of contradictions. Here, our goal is to find an equivalence class that has a large number of contradictions on Universum examples for training the distance metric used in NN classifications. In the next subsection, we will describe how to apply this principle.

2.3. Problem formulation

Recently, a number of approaches have been proposed to learn a distance metric from training data (e.g., generalized cosine distance, Bregman divergence, and Mahalanobis distance) [16]. We

aim at finding a Mahalanobis distance metric, where the distance between two examples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is computed as

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

where $\mathbf{M} \in \mathbb{R}^{D\times D}$ is a symmetric positive semidefinite matrix, i.e., $\boldsymbol{M} \succcurlyeq 0$. The Mahalanobis distance metric can be seen as a generalization of the Euclidean distance metric, where the correlation between features is taken into account [14]. Due to its computational simplicity, the Mahalanobis distance metric has been widely used in many pattern recognition problems (see [16] for a recent survey). Next, we recall some intuitions behind LMNN, which make it one of the most popular distance metric learning methods in order to improve the performance of NN classification.

First, the learned distance metric should minimize distances between each training example $\boldsymbol{x}_i$ and its nearest neighbor that shares the same class label, named *target neighbor*. To make the optimization problem convex, LMNN predefines all target neighbors as nearest neighbors of the same class using the Euclidean distance metric. Let η_{ij} be an indicator variable, which is given by

$$\eta_{ij} = \begin{cases} 1 & \text{, if } \mathbf{x}_j \text{ is the target neighbor of } \mathbf{x}_i\,; \\ 0 & \text{, otherwise}\,. \end{cases}$$

LMNN aims to *pull* target neighbors close together by using a function g_{pull} that penalizes large distances between each training example and its target neighbor:

$$g_{\text{pull}}(\mathbf{M}) = \sum_{ij} \eta_{ij} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)\,.$$

Second, the learned distance metric should maximize the distances between each training example and its neighbors of different classes. For each training example $\boldsymbol{x}_i$, a local perimeter surrounding its target neighbor is established. Examples that invade this perimeter are called *impostor neighbors*. LMNN aims to *push* impostor neighbors far away by using a function g_{push} that penalizes the small distances between each training example and its impostor neighbors:

$$g_{\text{push}}(\mathbf{M}) = \sum_{ijk} \eta_{ij} y_{jk} \Big[1 + d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k)\Big]_+\,,$$

where $[z]_+ = \max(z, 0)$ corresponds to the hinge loss function with margin one and y_{jk} is an indicator variable that takes value one if $y_j \neq y_k$, otherwise it takes value zero.

Therefore, we combine two terms $g_{\text{pull}}(\boldsymbol{M})$ and $g_{\text{push}}(\boldsymbol{M})$ using a weight parameter $\mu \in]0,\, 1[$ into an optimization problem for learning the Mahalanobis distance metric. Hence, LMNN can be expressed as the following problem,

$$\begin{aligned} \min \quad & (1-\mu) \sum_{ij} \eta_{ij} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \mu \sum_{ijk} \eta_{ij} y_{jk} \xi_{ijk} \\ \text{s.t.} \quad & d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijk}\,, \\ & \xi_{ijk} \geq 0, \\ & \mathbf{M} \succcurlyeq 0\,, \end{aligned} \tag{2}$$

where ξ_{ijk} are slack variables.

Our goal is to find an optimal distance metric induced by $\boldsymbol{M}$ that maximizes local margins for each training example and simultaneously minimizes margins for each Universum example. In other words, the decision function of the NN classifier on Universum examples should yield a value close to zero since only a small change in $\boldsymbol{M}$ will make a contradiction. We facilitate this goal by defining *dummy neighbors* for a Universum example $\mathbf{x}^*_i$ as two labeled examples, each one belonging to a class of interest. Assume that dummy neighbors are formed by the nearest neighbors in each class for each Universum example in the transformed space induced by the resulting Mahalanobis distance metric. These dummy neighbors can be selected by simply searching the nearest neighbor in the labeled data using the Euclidean distance metric. Note that the dummy neighbors do not change during the learning process. For minimizing the margins on Universum examples, each Universum example should have the same distances to its dummy neighbors. This is mainly due to the fact that the NN classifier is likely to increase the risk of misclassifying when distances to the nearest neighbors of different classes are equal, which renders it difficult to make an accurate prediction. If we try to keep the Universum examples having the same distances to their dummy neighbors, we implicitly try to maximize the number of contradictions. Let θ_{ijk} be an indicator variable, which is given by

$$\theta_{ijk} = \begin{cases} 1 & \text{, if } \mathbf{x}_j \text{ and } \mathbf{x}_k \text{ are dummy neighbors of } \mathbf{x}^*_i\,; \\ 0 & \text{, otherwise}\,. \end{cases}$$

We aim to make more *contradictions* on the Universum examples by using a function that penalizes large differences between the distances of a Universum example to its dummy neighbors, which is defined as

$$g_{\text{contra}}(\mathbf{M}) = \sum_{ijk} \theta_{ijk} \ell\Big(d^2_{\mathbf{M}}(\mathbf{x}^*_i, \mathbf{x}_j) - d^2_{\mathbf{M}}(\mathbf{x}^*_i, \mathbf{x}_k)\Big)\,,$$

where ℓ is a loss function penalizing outputs that are far away from zero. The function g_{contra} can be seen as a regularizer to reduce the risk of overfitting. Possible choices for the loss function are, for instance, quadratic loss and ϵ-insensitive loss. Similar as in [8], we choose the ϵ-insensitive loss function, which is given by

$$\ell(z) = \max(0, z - \epsilon) + \max(0, -z - \epsilon) = \max(0, |z| - \epsilon)\,.$$

Finally, we add $g_{\text{contra}}(\boldsymbol{M})$ to the objective function in (2) and obtain the following optimization problem

$$\begin{aligned} \min \quad & (1-\mu) \sum_{ij} \eta_{ij} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \mu \sum_{ijk} \eta_{ij} y_{jk} \xi_{ijk} + \frac{\lambda}{2} \sum_{ijk} \theta_{ijk} \xi^*_{ijk} \\ \text{s.t.} \quad & d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijk}\,, \\ & |d^2_{\mathbf{M}}(\mathbf{x}^*_i, \mathbf{x}_j) - d^2_{\mathbf{M}}(\mathbf{x}^*_i, \mathbf{x}_k)| \leq \epsilon + \xi^*_{ijk}\,, \\ & \xi_{ijk} \geq 0,\ \xi^*_{ijk} \geq 0\,, \\ & \mathbf{M} \succcurlyeq 0\,, \end{aligned} \tag{3}$$

where λ controls the trade-off between minimization of errors and maximization of contradictions. We refer to the proposed method as $\mathfrak{U}$-LMNN. Note that problem (3) is a convex semidefinite program, which can be solved using standard semidefinite programming.

Fig. 1 illustrates how the Universum examples affect the Mahalanobis distance metric learned by LMNN. In the absence of the Universum examples, distances between the considered Universum example and its dummy neighbors are large. However, these distances turn to be the same when using the distance metric learned by $\mathfrak{U}$-LMNN, and as a consequence, a small change in this distance metric will cause a contradiction on the Universum example.

2.4. Solver for $\mathfrak{U}$-LMNN

Since the number of constraints in (3) is likely to grow as the number of examples increases, standard semidefinite programmings tend to scale poorly. To further accelerate the solver for $\mathfrak{U}$-LMNN, we adopt a subgradient descent method which was designed to make LMNN more efficient [14] on large-scale data sets. To be more specific, it maintains an active list of those examples with margin violations and performs a full re-check only after a certain number of iterations.

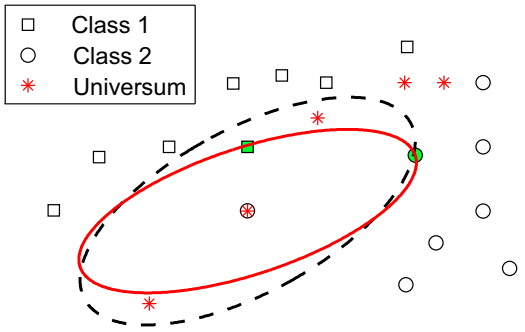

Fig. 1. Two separating ellipses represent unit circles produced by two different Mahalanobis distance metrics centered at a Universum example. The dashed black ellipse is the unit circle produced by LMNN and the solid red ellipse is the unit circle produced by $\mathfrak{U}$-LMNN. The green points denote the dummy neighbors of the considered Universum example (which is denoted by a black circle filled with a red asterisk). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

At the t-th iteration, $\mathfrak{U}$-LMNN operates as follows. Let $\boldsymbol{M}_t$ be the current solution, then the Mahalanobis distance metric can be rewritten as

$$d_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}_t (\mathbf{x}_i - \mathbf{x}_j)} = \sqrt{\mathrm{tr}\left(\mathbf{M}_t (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top\right)}.$$

For simplicity of notation, let $\mathbf{X}_{ij} = (\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\top$ and $\mathbf{U}_{ij} = (\mathbf{x}_i^* - \mathbf{x}_j)(\mathbf{x}_i^* - \mathbf{x}_j)^\top$, then we rewrite the objective function in (3) at the t-th iteration as

$$g(\mathbf{M}_t) = (1-\mu)\sum_{ij}\eta_{ij}\mathrm{tr}(\mathbf{M}_t\mathbf{X}_{ij}) + \mu\sum_{ijk}\eta_{ij}y_{jk}\Big[1 + \mathrm{tr}(\mathbf{M}_t\mathbf{X}_{ij}) - \mathrm{tr}(\mathbf{M}_t\mathbf{X}_{ik})\Big]_+ + \frac{\lambda}{2}\sum_{ijk}\theta_{ijk}\Big[\big|\mathrm{tr}(\mathbf{M}_t\mathbf{U}_{ij}) - \mathrm{tr}(\mathbf{M}_t\mathbf{U}_{ik})\big| - \epsilon\Big]_+ .$$

Let us define the following sets of active constraints, for labeled examples,

$$\mathcal{A}_t = \Big\{ (i,j,k) \mid \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in \mathcal{X}, \eta_{ij} = 1, y_{ik} = 1, \text{ and } d_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_k) - d_{\mathbf{M}_t}(\mathbf{x}_i, \mathbf{x}_j) < 1 \Big\},$$

and for Universum examples with their dummy neighbors,

$$\mathcal{A}_t^* = \Big\{ (i,j,k) \mid \mathbf{x}_i^* \in \mathcal{U}, \mathbf{x}_j, \mathbf{x}_k \in \mathcal{X}, \theta_{ijk} = 1, \text{ and } d_{\mathbf{M}_t}(\mathbf{x}_i^*, \mathbf{x}_j) - d_{\mathbf{M}_t}(\mathbf{x}_i^*, \mathbf{x}_k) > \epsilon \Big\}.$$

Based on these definitions, the subgradient of $g(\boldsymbol{M}_t)$ can be computed as

$$\mathbf{G}_t = (1-\mu)\sum_{ij}\eta_{ij}\mathbf{X}_{ij} + \mu\sum_{(i,j,k)\in\mathcal{A}_t}(\mathbf{X}_{ij} - \mathbf{X}_{ik}) + \lambda\sum_{(i,j,k)\in\mathcal{A}_t^*}(\mathbf{U}_{ij} - \mathbf{U}_{ik}).$$

Hence, $\mathbf{M}_{t+1}$ can be obtained by moving a step in the reverse direction of $\boldsymbol{G}_t$, that is,

$$\mathbf{M}_{t+1} = \mathbf{M}_t - s\mathbf{G}_t,$$

where s denotes the step size. To enforce the positive semidefiniteness constraint on the matrix $\mathbf{M}_{t+1}$, we project it onto the positive semidefinite cone after each gradient step. This procedure continues until all constraints are satisfied or a predetermined number of iterations is reached.

3. Computational complexity

In this section, we analyze the computational complexity of $\mathfrak{U}$-LMNN. The computational complexity of the standard LMNN on each iteration scales as $O(nD^2 + n^2D)$. This is mainly due to the computation of the subgradient. Our method $\mathfrak{U}$-LMNN, in addition, requires to compute the subgradient contribution from dummy neighbors, which scales as $O(mD^2)$. After updating the distance metric, we need to project it onto the cone of positive semidefinite matrices, which scales as $O(D^3)$. Summarizing, the overall computational complexity of $\mathfrak{U}$-LMNN in each iteration is $O(nD^2 + n^2D + mD^2 + D^3)$. In order to further reduce this computational burden in large-scale settings, we use the special solver proposed by Weinberger and Saul [14]. More specifically, it is based on the active set technique and only computes the full subgradient after a certain number of iterations. We have slightly modified the Matlab implementation of LMNN for $\mathfrak{U}$-LNNN.

4. Related work

Our work is closely related to two groups of research works. The first group is Universum learning and the second group is distance metric learning. We briefly review some representative work in both groups.

The idea of using Universum data to make inferences was firstly introduced in [5] and further developed in [7,8] with a new support vector machine framework, namely $\mathfrak{U}$-SVM. After Vapnik [7] several authors [17,18] focused on the algorithmic implementation of $\mathfrak{U}$-SVM and its empirical validation. For instance, Chapelle et al. [19] analyzed the geometrical relation between the hyperplane learned by $\mathfrak{U}$-SVM and the Universum set and extended $\mathfrak{U}$-SVM to a least-squares version. Cherkassky and Dai [17] studied the effectiveness of the RA strategy for generating the Universum examples on $\mathfrak{U}$-SVM in high-dimensional settings. Learning with Universum data has also been applied to many other machine learning methods [20–22]. It was used, for instance, to adapt the objective function of a boosting method in a recent development namely $\mathfrak{U}$-Boost [20]. Experimental results confirmed that $\mathfrak{U}$-Boost outperforms those traditional AdaBoost methods without considering Universum data. In a clustering setting, Zhang et al. [21] demonstrated the advantages of using Universum data for document clustering. Our method was inspired by the success of $\mathfrak{U}$-SVM and $\mathfrak{U}$-Boost.

Distance metric learning has been extensively studied in the literature. Based on the availability information of the training data, distance metric learning can be divided into two categories, namely unsupervised and supervised. Unsupervised distance metric learning methods try to preserve the geometric relationships between training examples as much as possible. These methods are usually used to find a low-dimensional manifold from the high-dimensional input data, including principal component analysis (PCA) [23], Isomap [24], and locally linear embedding (LLE) [25]. On the other hand, supervised distance metric learning methods try to estimate an appropriate distance metric for classification problems from the training data with explicit class labels. Some representative methods in this group include information-theoretic metric learning (ITML) [26], large-margin nearest neighbor (LMNN) [14], distance metric learning with eigenvalue optimization (DML-eig) [11], and distance metric learning through maximization of the Jeffrey divergence (DMLMJ) [13]. Most of the existing supervised methods only consider the labeled examples that need to be classified and neglect the useful information that can be obtained from some other sources. A few methods tried to incorporate unlabeled data, which are referred to as semi-supervised distance metric learning. For instance, Yeung and Chang [27] used a kernel to describe the relationship between labeled and unlabeled

data. In a recent work, Wang et al. [28] proposed a regularization on unlabeled data based on local topology (i.e., region density and manifold information). Although these methods exploit the information about the domain distribution contained in the unlabeled data, none of them performs the principle of maximal number of contradictions. Compared to these semi-supervised distance metric learning methods, $\mathfrak{U}$-LMNN is simpler and easier to implement.

5. Experiments

In this section, we describe some experiments to evaluate the performance of $\mathfrak{U}$-LMNN against other distance metric learning methods, including the baseline Euclidean distance metric and LMNN. All of the experiments are performed in the context of NN classification. We use the source codes implemented in Matlab of LMNN[1] supplied by the authors. The implementation of $\mathfrak{U}$-LMNN is also available online[2]. All hyper-parameters are selected using cross-validation. Following Weinberger and Saul [14], we tune the hyper-parameter μ for LMNN and $\mathfrak{U}$-LMNN considering as set of values {0.125, 0.25, 0.5}. For the hyper-parameters of $\mathfrak{U}$-LMNN, we consider as possible values $\lambda \in \{0.001, 0.25, 0.5, 1, 2\}$ and $\epsilon \in \{0, 0.01, 0.1, 1, 1.5, 2\}$. The narrow range for λ (as for ϵ above) is motivated by the fact that if λ is too small, $\mathfrak{U}$-LMNN performs similarly as LMNN and if λ is too large, $\mathfrak{U}$-LMNN performs rather poorly.

5.1. Synthetic data

We carry out an experiment on a two-dimensional data set to highlight the influence of using Universum examples. The data set consists of 50 positive examples, 50 negative examples, and 100 Universum examples. Both negative examples and positive examples follow a Gaussian distribution with means $\boldsymbol{\mu}_1 = (-0.70; -0.70)$ and $\boldsymbol{\mu}_2 = (0.70; 0.70)$ respectively, and the same covariance matrix $\boldsymbol{\Sigma} = (2.00, -0.35; -0.35, 0.08)$. The Universum examples follow a Gaussian distribution with mean $\boldsymbol{\mu} = (0.00; 0.00)$ and covariance matrix $\boldsymbol{\Sigma} = (2.50, -0.60; -0.60, 0.20)$.

Fig. 2 shows the decision boundaries separating positive and negatives examples induced by LMNN and $\mathfrak{U}$-LMNN. From the results, we can see that the decision boundary induced by LMNN is close to the horizontal line, which separates the labeled examples well. In addiction to separating the labeled examples well, $\mathfrak{U}$-LMNN also tries to induce a decision boundary that easily makes contradictions on the Universum examples. As a result, the Universum examples are closer to the decision boundary induced by $\mathfrak{U}$-LMNN than that induced by LMNN. Clearly, $\mathfrak{U}$-LMNN induces a more accurate and reasonable classifier with the help of the Universum examples. In other words, $\mathfrak{U}$-LMNN can achieve a better generalization capacity than LMNN.

5.2. USPS

In this experiment, we evaluate the performance of our method on the USPS ZIP code[3] handwritten digits. The data set consists of 9,298 handwritten digit images, from 0 to 9, of size 16×16 (7291 examples for training and 2007 examples for test). In particular, we carry out experiments on two-class classification problems, digits 5 versus 8, using digit 3 as Universum data due to its usefulness illustrated in [8,20]. The test examples are extracted from the original test set, which contains 160 images of digit 5 and 166 images of digit 8. Universum examples are extracted from the original training set, which contains 658 images of digit 3. We randomly select subsets of sizes 100, 300, 400, 500, 700 for training and a subset of size 326 for validation from the original training set.

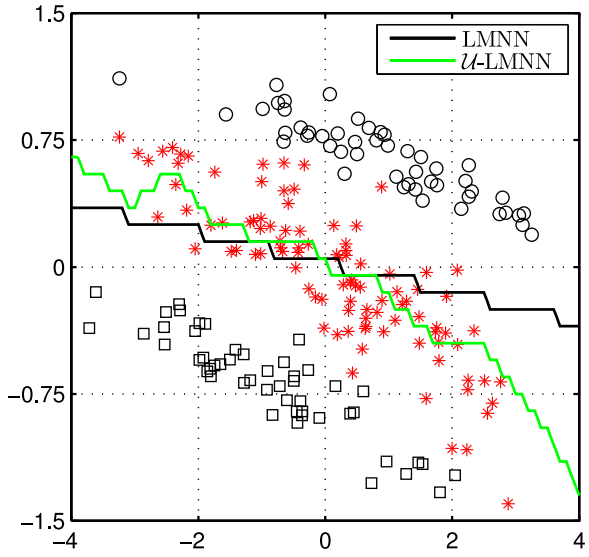

Fig. 2. A synthetic data set illustrating the difference between the decision boundaries induced by LMNN and $\mathfrak{U}$-LMNN. The data set consists of 50 positive examples denoted by black circles, 50 negative examples denoted by black squares, and 100 Universum examples denoted by red asterisks. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1 presents the overall training and test errors of NN classifier obtained over five runs using different distance metric methods. The training errors reported are leave-one-out estimates. The experimental results indicate that $\mathfrak{U}$-LMNN obtains the most accurate classification results. It achieves an error rate of 4.17% for a training data size of 100 and 2.19% for a training data size of 700. Both $\mathfrak{U}$-LMNN and LMNN consistently outperform the Euclidean distance metric. Note that $\mathfrak{U}$-LMNN has three tunable hyper-parameters, which can make the selection difficult in practice. In contrast, LMNN has only one tunable hyper-parameter. Consequently, LMNN may yield better performance than $\mathfrak{U}$-LMNN due to the robust hyper-parameter selection.

5.3. MNIST

In the second experiment, we perform an evaluation on the MNIST[4] handwritten digits. The original data set contains 70,000 handwritten digit images of size 28×28 (60,000 examples for training and 10,000 examples for test). Similar to the USPS data set, we perform experiments on two-class classification problems, digits 5 versus 8. We report the results by varying the size of the training set as 200, 300, 500, 700, 1,000, and 1,500. The training and validation subsets are randomly selected from the original training set, which contains 5421 images of digit 5 and 5851 images of digit 8. We use the original test set, which contains 892 images of digit 5 and 974 images of digit 8, to estimate the test performance. The Universum set consists of 1000 images of digit 3 randomly selected from the original training data.

Table 2 presents the overall training and test errors of the NN classifier using different distance metric learning methods over five runs. Due to the high-dimensional nature and relatively small training data size, both LMNN and $\mathfrak{U}$-LMNN obtain a good performance on training. However, for all cases, $\mathfrak{U}$-LMNN outperforms

[1] http://www.cse.wustl.edu/~kilian/code/.
[2] http://users.ugent.be/~bacnguye/ulmnn.zip.
[3] http://www-stat.stanford.edu/~tibs/ElemStatLearn/data.html.
[4] http://yann.lecun.com/exdb/mnist/.

Table 1
Training and test errors (standard deviations) on the USPS data set.

	Method	Training data size				
		100	300	400	500	700
Training	Euclidean	6.00 (2.00)	1.40 (0.55)	2.30 (0.60)	1.92 (0.81)	1.51 (0.24)
	LMNN	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)
	$\mathfrak{U}$-LMNN	0.60 (1.34)	0.07 (0.15)	**0.00** (0.00)	**0.00** (0.00)	0.34 (0.77)
Test	Euclidean	5.15 (1.18)	3.74 (0.90)	3.01 (0.66)	3.74 (0.55)	3.07 (0.22)
	LMNN	4.42 (1.12)	3.13 (1.05)	2.64 (0.67)	2.88 (0.27)	**2.09** (0.59)
	$\mathfrak{U}$-LMNN	**4.17** (0.91)	**2.96** (1.13)	**2.52** (0.76)	**2.76** (0.22)	2.19 (0.37)

Table 2
Training and test errors (standard deviations) on the MNIST data set.

	Method	Training data size				
		300	500	700	1000	1500
Training	Euclidean	4.27 (0.86)	3.56 (0.70)	4.01 (0.58)	3.74 (0.33)	2.47 (0.13)
	LMNN	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	0.27 (0.75)
	$\mathfrak{U}$-LMNN	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)	**0.00** (0.00)
Test	Euclidean	4.59 (0.34)	3.91 (0.39)	3.39 (0.20)	3.16 (0.21)	2.93 (0.28)
	LMNN	3.92 (0.68)	2.83 (0.55)	2.43 (0.33)	2.08 (0.30)	1.61 (0.28)
	$\mathfrak{U}$-LMNN	**3.44** (0.50)	**2.63** (0.53)	**2.30** (0.31)	**1.92** (0.27)	**1.36** (0.18)

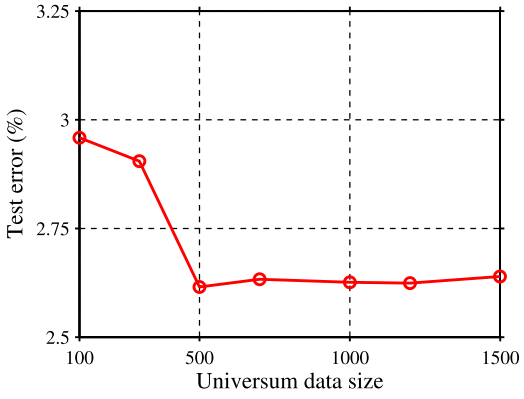

Fig. 3. Test error versus Universum data size of the NN classifier using $\mathfrak{U}$-LMNN on the MNIST data set (digits 5 versus 8).

LMNN and the Euclidean distance metric on the test set. These experimental results again confirm the validity of Universum data.

Fig. 3 shows the performance of $\mathfrak{U}$-LMNN with a varying number of Universum examples (100, 500, 1,000, and 1,500) and a training set of 500 examples. In the absence of Universum data, LMNN achieves an error rate of 2.83%. From the results shown in Fig. 3, we can observe that increasing the Universum data size yields an increased classification performance for $\mathfrak{U}$-LMNN. Once a certain number of Universum examples is reached, the performance remains more or less the same.

6. Conclusions

We have proposed a distance metric learning method, named $\mathfrak{U}$-LMNN, for NN classification by exploiting the available information contained in the Universum examples. Our method employs the principle of maximal number of contradictions on Universum examples to improve the classification performance of the NN classifier. Since $\mathfrak{U}$-LMNN optimization is convex, we expect that adding "bad" Universum examples will not affect the convergence rate as in other nonconvex semi-supervised distance metric learning methods. Experimental results have shown that $\mathfrak{U}$-LMNN outperforms the conventional LMNN without using Universum examples, especially on data sets with small sizes. Experiments on the digit recognition data sets also showed that $\mathfrak{U}$-LMNN is competitive with other Universum learning methods such us $\mathfrak{U}$-SVM [8] and $\mathfrak{U}$-Boost [20]. Moreover, the loss on Universum examples used by $\mathfrak{U}$-LMNN can be seen as an additional regularizer for the decision boundary, therefore, it could be easily adapted to other distance metric learning methods.

Acknowledgment

The authors would like to acknowledge the financial support for this project from the Special Research Fund – Doctoral Scholarships, Ghent University, Belgium, under grant No. BOF15/DOS/039.

References

[1] L.M. Cannas, N. Dessì, B. Pes, Assessing similarity of feature selection techniques in high-dimensional domains, Pattern Recognit. Lett. 34 (12) (2013) 1446–1453.
[2] Y. Wang, Y. Fan, P. Bhatt, C. Davatzikos, High-dimensional pattern regression using machine learning: from medical images to continuous clinical variables, Neuroimage 50 (4) (2010) 1519–1535.
[3] M. Sun, C.E. Priebe, Efficiency investigation of manifold matching for text document classification, Pattern Recognit. Lett. 34 (11) (2013) 1263–1269.
[4] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, second edition, Wiley-Interscience, 2012.
[5] V.N. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.
[6] P.F. Evangelista, M.J. Embrechts, B.K. Szymanski, Taming the curse of dimensionality in kernels and novelty detection, in: Applied Soft Computing Technologies: The Challenge of Complexity, Springer, Berlin, Heidelberg, 2006, pp. 425–438.
[7] V.N. Vapnik, Estimation of Dependences Based on Empirical Data, Springer-Verlag, New York, 2006.
[8] J. Weston, R. Collobert, F. Sinz, L. Bottou, V.N. Vapnik, Inference with the Universum, in: Proceedings of the 23rd International Conference on Machine Learning, New York, NY, USA, 2006, pp. 1009–1016.
[9] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[10] C. Shen, J. Kim, L. Wang, A. Van Den Hengel, Positive semidefinite metric learning using boosting-like algorithms, J. Mach. Learn. Res. 13 (1) (2012) 1007–1036.
[11] Y. Ying, P. Li, Distance metric learning with eigenvalue optimization, J. Mach. Learn. Res. 13 (1) (2012) 1–26.
[12] V.E. Liong, J. Lu, Y. Ge, Regularized local metric learning for person re-identification, Pattern Recognit. Lett. 68, Part 2 (2015) 288–296.
[13] B. Nguyen, C. Morell, B. De Baets, Supervised distance metric learning through maximization of the Jeffrey divergence, Pattern Recognit. 64 (2017) 215–225.
[14] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (2009) 207–244.
[15] H. Do, A. Kalousis, J. Wang, A. Woznica, A metric learning perspective of SVM: on the relation of LMNN and SVM, in: Proceedings of the 15th International Conference on Artificial Intelligence and Statistics, 2012, pp. 308–317.
[16] A. Bellet, A. Habrard, M. Sebban, Metric Learning, Morgan & Claypool Publishers, 2015.

[17] V. Cherkassky, W. Dai, Empirical study of the Universum SVM learning for high-dimensional data, in: Proceedings of 19th International Conference Artificial on Neural Networks, 2009, pp. 932–941.
[18] Z. Qi, Y. Tian, Y. Shi, Twin support vector machine with Universum data, Neural Netw. 36 (2012) 112–119.
[19] O. Chapelle, A. Agarwal, F.H. Sinz, B. Schölkopf, An analysis of inference with the Universum, in: Advances in Neural Information Processing Systems 20, 2008, pp. 1369–1376.
[20] C. Shen, P. Wang, F. Shen, H. Wang, $\mathcal{U}$Boost: boosting with the Universum, IEEE Trans. Pattern Anal. Mach. Intell. 34 (4) (2012) 825–832.
[21] D. Zhang, J. Wang, L. Si, Document clustering with Universum, in: Proceedings of the 34th International Conference on Research and Development in Information Retrieval, 2011, pp. 873–882.
[22] B. Peng, G. Qian, Y. Ma, Recognizing body poses using multilinear analysis and semi-supervised learning, Pattern Recognit. Lett. 30 (14) (2009) 1289–1294.
[23] I. Jolliffe, Principal Component Analysis, Wiley Online Library, 2005.
[24] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[25] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[26] 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, 2007, pp. 209–216.
[27] D.Y. Yeung, H. Chang, A kernel approach for semisupervised metric learning, IEEE Trans. Neural Netw. 18 (1) (2007) 141–149.
[28] Q. Wang, P.C. Yuen, G. Feng, Semi-supervised metric learning via topology preserving multiple semi-supervised assumptions, Pattern Recognit. 46 (9) (2013) 2576–2587.