Prog Artif Intell (2013) 2:85–96
DOI 10.1007/s13748-012-0039-1

REGULAR PAPER

Passive–aggressive online distance metric learning and extensions

Adrián Pérez-Suay · Francesc J. Ferri · Miguel Arevalillo-Herráez

Received: 16 November 2012 / Accepted: 30 December 2012 / Published online: 24 January 2013

Abstract Online metric learning aims at constructing an appropriate dissimilarity measure from training data composed of labeled pairs of samples. Margin maximization has been widely used as an efficient approach to address this problem. This paper reviews various existing online metric learning formulations, and also introduces an alternative passive–aggressive scheme. In addition, the pros and cons of each alternative are analyzed in a comparative empirical study over several databases.

Keywords Distance metric learning · Online learning · Distance-based recognition

1 Introduction

Processing and analyzing data from different sources has become nowadays a problem of capital importance in many application domains e.g. images, video, multimedia information or biological data. Many of these give rise to data descriptors that keep growing due to better capacity of the corresponding sensor systems. In this context, distance-based methods are gaining importance over other classical feature-based methods, that use data descriptors or features.

Once an appropriate measure is given or has been learned, the well-known and studied family of distance-based methods can be used to tackle either classification, regression, estimation or clustering problems using only pairwise comparisons between objects.

The systematic search of an appropriate distance measure to be jointly used with either general or specific distance-based classification schemes (mainly Nearest Neighbor rules) has been studied almost as early as the corresponding classification methods [1,2]. Nevertheless, the so-called distance metric learning (DML) has suffered a kind of rebirth recently as optimization methods and formulations coming from other neighboring fields have been applied to this particular problem [3–5].

In the most basic case, DML aims at learning an appropriate Mahalanobis-like metric from data i.e. obtain an appropriate positive semidefinite (PSD) matrix that induces a quadratic norm in the original feature space. This basic approach can be generalized in multiple ways but many methods start by defining an appropriate (usually convex) criterion function and then use a convenient procedure to optimize it [6,7]. These criterion functions respond to the intention of keeping similar objects close and dissimilar objects apart at the same time. Metric learning implies optimization procedures whose computational burden is at least quadratic on the amount of training data. Moreover, the constraints that need to be imposed on both the relative object similarities and the metric matrix lead to computationally expensive optimization methods, specially when dealing with large-scale problems [8]. The need to overcome such computationally expensive tasks has raised the interest in other indirect ways of achieving the goals of DML [9,10]. In this context, it is worth mentioning the family of methods that use an online

Work partially funded by FEDER and Spanish Government through projects TIN2009-14205-C04-03, TIN2011-29221-C03-02 and Consolider Ingenio 2010 CSD07-00018.

A. Pérez-Suay · F. J. Ferri (✉) · M. Arevalillo-Herráez
Dept. Informàtica, Universitat de València, Burjassot 46100, Spain
e-mail: Francesc.Ferri@uv.es

A. Pérez-Suay
e-mail: Adrian.Perez@uv.es

M. Arevalillo-Herráez
e-mail: Miguel.Arevalillo@uv.es

approach [11]. These methods are specially relevant when data becomes available in an incremental form i.e. they are given sequentially or are obtained from a stream.

Incremental and online methods [5,10] usually aim at optimizing a convenient criterion over a single instance (usually pairs of objects) which is made available for learning at every time step in the corresponding algorithm. In this case, inherent problems to other metric learning techniques turn more complicated; usual problems related to positive semi-definiteness constraints that are common to batch methods still apply; and new problems arise. First, different ways of sequentially enforcing additional constraints may lead to algorithms that require different amounts of computation. Also, the performance of the final solution may deviate significantly from the ideal (global) goal depending on the particular instances used in the last few iterations. Finally, parameter tuning gets considerably harder than in the batch case, according to recent preliminary results carried out in this context [12].

2 The problem

2.1 Notation

Let us assume there are objects $x_i, i = 1, 2, \ldots$ conveniently represented in the d-dimensional vector feature space, that is $x_i \in \mathbb{R}^d$. Let us also assume that there is a similarity relation among these objects such that given any pair, (x_i, x_j), they can be considered either as similar or dissimilar. As we identify each pair of objects with indices i and j, we use the label y_{ij} to indicate whether this pair is similar ($y_{ij} = +1$) or dissimilar ($y_{ij} = -1$).

We could try to learn a predictor for this relation over the whole representation space, provided that some of these labels were available. Instead of this, a closely related problem is considered. This consists of learning a distance function that is compatible with the labeling.

2.2 Foundations

A real symmetric function $d : \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}$ that satisfies nonnegativity and triangle inequality is called a pseudometric. It is a distance if it also satisfies the identity property i.e. it gives zero if and only if the two objects are the same. A wide range of different pseudometrics (including the Euclidean distance) can be represented by using a PSD matrix M as follows:

$$d_{ij}^M = d^M(x_i, x_j) = (x_i - x_j)^\top M (x_i - x_j). \tag{1}$$

Note that the pseudometric function is $\sqrt{d^M}$, but by convenience and without loss of generality only its squared version will be explicitly considered. It must be also emphasized that in the context of the present paper and as in many other related works, we will refer to square pseudometric functions when we use the term distance function.

2.3 Problem formulation

A distance function gives us a way of comparing objects. Hence, the function should yield small values for similar objects; and larger ones for dissimilar pairs. Given a set of pairs of objects that are known to be either similar or dissimilar, we can pose the problem of obtaining the best distance function that is compatible with these given pairs. Ideally, the distance function obtained should keep all similar pairs strictly closer than all dissimilar pairs. This situation is illustrated in Fig. 1a. This will be referred to as the separable case. In a more realistic case in which samples cannot be fully separated in this way, it is still possible to aim for a distance function that makes most similar pairs be roughly closer than most dissimilar pairs. We refer to this as the non-separable case (see Fig. 1b).

3 Metric learning

3.1 Metric learning using all labelled instances at once

3.1.1 Separable case

When all labelled pairs are given at once and solutions separating the data exist, one can think of an ideal solution as the one that maximizes the separation margin according to the Structural Risk Minimization principle [3]. As the absolute values of the distance depend on multiplicative constants in the matrix M, this problem can be reformulated in a similar way as it is done with support vector machines [4]. In particular, a fixed value for this margin is set and the goal becomes to minimize the Frobenius norm [13,14] (or any other regularizer [5]) of the metric matrix.

If we denote by $b \in \mathbb{R}$ the threshold value that effectively separates the similar and dissimilar samples, and fix the value of the margin to 2, the following constraints apply:

$$d_{ij}^M \leq b - 1, \quad \text{if } y_{ij} = 1,$$
$$d_{ij}^M \geq b + 1, \quad \text{if } y_{ij} = -1,$$

or, in a more compact way

$$y_{ij}\left(b - d_{ij}^M\right) \geq 1. \tag{2}$$

These linear constraints, together with the regularization criterion and the PSD constraint lead to a quadratic optimization problem that can be tackled and solved in a number of ways [15,16] and results in the metric matrix M and the value b that effectively separates both kinds of distance values.

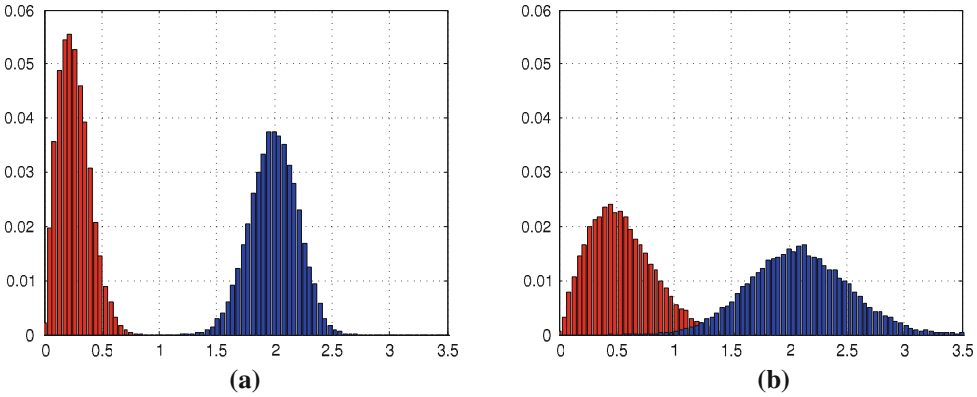

Fig. 1 Normalized histogram of distance values of similar (*red*) and dissimilar (*blue*) pairs of objects corresponding to bivariate Gaussians whose separation between means is 1. **a** separable case, $\sigma = 0.02$, **b** non-separable case, $\sigma = 0.1$

3.1.2 Non-separable case

Regardless of other considerations, the constraints in Eq. (2) can be modified to take into account the non-separable case by introducing a (nonnegative) slack variable for each labelled pair. The margin constraints then become

$$y_{ij}\left(b - d_{ij}^{M}\right) \geq 1 - \xi_{ij}. \tag{3}$$

This needs to be coupled with the inclusion of a new term in the optimization criterion to also minimize the values of the slack variables. This is usually referred to as establishing a soft margin to separate both kind of distance values and it is equivalent to minimizing the hinge loss associated with all pairs that violate the (hard) margin separation. Given a model (M, b) and a pair (i, j), the corresponding hinge loss is given as

$$\ell_{ij}^{(M,b)} = \max\left\{0, p_{ij}^{(M,b)}\right\} = \max\left\{0, 1 - y_{ij}\left(b - d_{ij}^{M}\right)\right\}$$

where $p_{ij}^{(M,b)}$ is the signed loss predicted for the pair (i, j) when using the model (M, b). Consequently, the constraint in Eq. (3) can be then expressed as $p_{ij}^{(M,b)} \leq \xi_{ij}$ or equivalently as $\ell_{ij}^{(M,b)} \leq \xi_{ij}$.

3.2 Online formulation using margins

Instead of considering constrained optimization using all available instances, the DML problem can be solved in a more convenient way both from the point of view of computation and robustness by using an online learning approach [10,17].

At each time step, k, a new optimization problem is formulated and solved using only a particular instance (a labeled pair (i, j)) that is made available to the system. The problem makes use of the model (M^k, b^k) learned at the previous step to produce a new model that takes also the new instance into account.

3.2.1 Separable case

In the separable case this is done by minimizing a convenient measure of the distance between the previous and the new model, subject to the restriction that the new instance must fall on the correct side of the (hard) margin [10]. In particular, this can be written as

$$\min_{M,b} \ \frac{1}{2}\left\|M - M^k\right\|_{\text{Fro}}^2 + \frac{1}{2}(b - b^k)^2 \tag{4}$$

$$s.t. \ \ \ell_k^{(M,b)} = 0, \tag{5}$$

where ℓ_k refers to the hinge loss on the kth pair, (x_i, x_j) and $\|\cdot\|_{\text{Fro}}$ is the Frobenius norm. An obvious consequence of this formulation is that the model only needs to be updated if the new instance yields a strictly positive loss when using the previous model.

3.2.2 Non-separable case

In the non-separable case, the new instance is allowed to violate the margin condition but it is penalized by using a parameter C that leads to the following optimization problem.

$$\min_{M,b,\xi} \ \frac{1}{2}\left\|M - M^k\right\|_{\text{Fro}}^2 + \frac{1}{2}(b - b^k)^2 + C\xi, \tag{6}$$

$$s.t. \ \ \ell_k^{(M,b)} \leq \xi, \tag{7}$$

$$\xi \geq 0. \tag{8}$$

In this formulation, the (nonnegative) slack variable, ξ, allows a particular instance to violate the constraint. An alternative way of posing the problem is to remove the nonnegativity constraint on the slack variable [Eq. (8)] and consider an squared penalty term in Eq. (6).

All the above formulations including both separable and inseparable cases are known as passive–aggressive (PA) learning formulations and were introduced in [11]. Following the taxonomy introduced by the authors, we will refer to these as PA (separable case), PAI (non-separable, linear penalty) or PAII (non-separable, quadratic penalty).

These formulations have been previously applied to DML or closely related problems [10,17,18].

It has been shown [10] that in all previous cases the solution of these problems has a closed-form solution that can be written as the following update rule

$$M^{k+1} = M^k - \tau y_{ij}(x_i - x_j)(x_i - x_j)^\top, \tag{9}$$

$$b^{k+1} = b^k + \tau y_{ij}, \tag{10}$$

where the value of τ is called the step length and depends on the particular formulation. The corresponding values of τ for the three formulations considered are as follows.

$$\text{PA}: \tau_0 = \frac{\ell_k}{1 + \|(x_i - x_j)(x_i - x_j)^\top\|_{\text{Fro}}^2}, \tag{11}$$

$$\text{PAI}: \tau_1 = \min\{C, \tau_0\}, \tag{12}$$

$$\text{PAII}: \tau_2 = \frac{\ell_k}{1 + \frac{1}{2C} + \|(x_i - x_j)(x_i - x_j)^\top\|_{\text{Fro}}^2}. \tag{13}$$

These step lengths are dependent on the hinge loss incurred by the new k-pair, according to the currently available kth predictive model $\ell_k = \ell_k^{(M^k, b^k)}$. If the pair similarity is consistent with the model at the previous iteration, we have $\ell_k = 0$ and then $\tau_n = 0$. This implies no training at this iteration (passiveness). On the other hand, when the prediction violates the (hard or soft) margin, the model gets updated either strictly (τ_0) or controlled by the aggressiveness parameter C. Hence, C controls how strongly the algorithm adapts the model to each example pair.

In the particular case of metric learning, two additional constraints are needed in the definition of the problem [10]. First, the matrix M must be PSD, $M \succeq 0$. Second, and as a consequence, the value of the threshold b must be above 1, $b \geq 1$.

In all previous works, these constraints are not considered in the formulation of the optimization problem. On the contrary, they are simultaneously enforced in a separate step, once the optimization problem has been solved and both matrix M and the threshold b have been computed. The final PSD matrix is the closest PSD approximation; and the final threshold is defined as $\max(1, b)$. Note that this computation only needs to be performed when $y_{ij} = 1$ (the new training pair is a similar one). Even in this case, the computational burden can be partially alleviated by considering that Eq. (9) is a rank-one update on a PSD matrix and using the method proposed in [10].

3.3 Online metric learning using least squares

A slightly different alternative formulation using least squares [19] is also possible for the previous metric learning problem [12]. Instead of forcing a soft margin by penalizing the deviation from the ideal conditions, it is possible to force similar and dissimilar distance values to fall close to the "representative" values $b - 1$ and $b + 1$, respectively. To this end, one can sequentially minimize the corresponding squared error. This corresponds to reformulating the previous (PAII version) optimization problem as:

$$\min_{M,b,\xi_k} \frac{1}{2}\|M - M^k\|_{\text{Fro}}^2 + \frac{1}{2}(b - b^k)^2 + C\xi_k^2, \tag{14}$$

$$s.t. \quad p_{ij}^{(M,b)} = \xi_k. \tag{15}$$

The main change in this formulation is that the inequality constraint in Eq. (7) has been changed to an equality and the signed loss function, $p_{ij}^{(M,b)} = 1 - y_{ij}\left(b - d_{ij}^M\right)$, now measures how far the distance value is from its corresponding ideal value ($b - 1$ or $b + 1$).

The corresponding online optimization problem can now be tackled in a similar way as the previous ones. This also leads to a closed-form solution (see Appendix 5), that consists of the same update rule expressed in Eqs. (9) and (10), but with a different step length given by

$$\tau_3 = \frac{p_k}{1 + \frac{1}{2C} + \|(x_i - x_j)(x_i - x_j)^\top\|_{\text{Fro}}^2}. \tag{16}$$

where $p_k = p_{ij}^{(M^k, b^k)} = 1 - y_{ij}\left(b^k - d_{ij}^{M^k}\right)$ is the signed loss predicted using the kth learned model. Note that τ_3 can now take negative values and it holds that $\tau_2 = \max\{0, \tau_3\}$. Consequently, the corresponding algorithm can be considered as more aggressive because it implies a larger number of updates. As the approach shares the structure and part of the aims of the PA approaches, it will be referred to as PALS (passive–aggressive least squares) in this work. Only when it holds that $p_k = 0$, a passive step is performed. This only occurs when the distance value takes exactly its desired value. This is in contrast to pure passive-aggressive approaches, which perform a passive step when $\ell_k = \max(0, p_k) = 0$, i.e. $p_k \leq 0$.

One can see that all previous learning algorithms and in particular the corresponding step lengths, τ_n, are closely related to each other. In particular, we can write these step lengths as a function of C and p_k to put forward their interdependences.

$$\text{PALS}: \tau_3(C, p_k) = \frac{p_k}{1 + \frac{1}{2C} + \|X_{ij}\|_{\text{Fro}}^2} \in]-\infty, +\infty[,$$

$$\text{PAII}: \tau_2(C, p_k) = \max(0, \tau_3(C, p_k)) \in [0, +\infty[,$$

$$\text{PA}: \tau_0(p_k) = \lim_{C \to \infty} \tau_2(C, p_k) \in [0, +\infty[,$$

$$\text{PAI}: \tau_1(C, p_k) = \min(C, \tau_0(p_k)) \in [0, +\infty[,$$

where $X_{ij} = (x_i - x_j)(x_i - x_j)^\top$.

It is possible to show a graphical illustration to clarify how the different step lengths relate to each other. In Fig. 2a, the values of τ_n are plotted as a function of p_k for a given value of C. It can be clearly seen that τ_1 is a saturated version of τ_o while τ_2 corresponds to a milder version of the

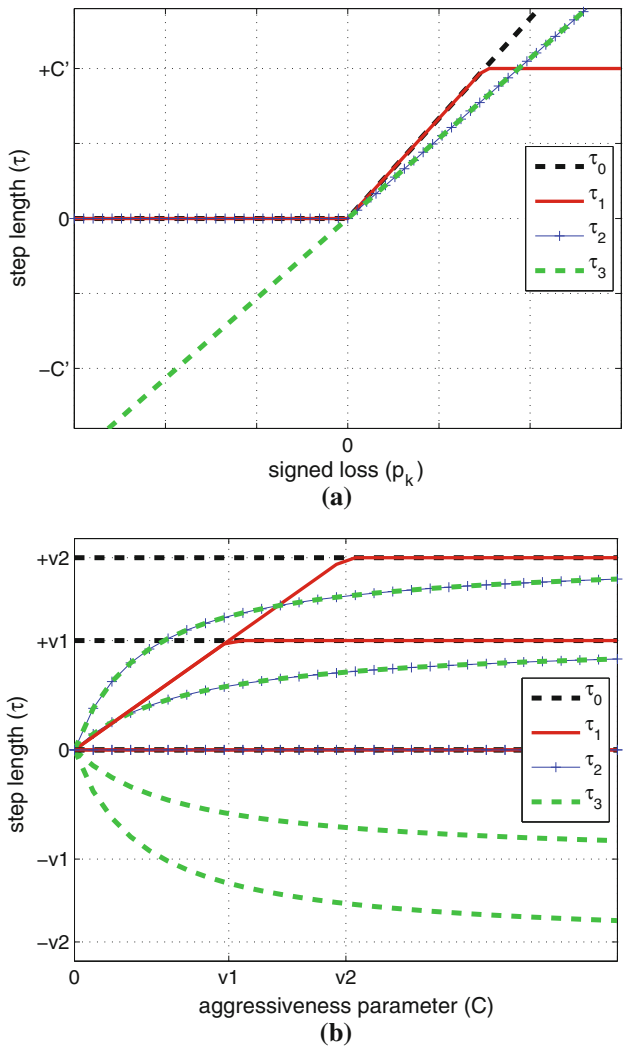

Fig. 2 Different step lengths corresponding to separable PA, PAI, PAII and PALS. **a** As a function of the signed loss for the current pair, p_k for a given value of C. The value C' shown corresponds to $C \cdot (1+\|(x_i - x_j)(x_i - x_j)^\top\|^2_{\text{Fro}})$. **b** As a function of C for two different values of p_k that lead to two different asymptotic values (v_1 and v_2, see *text*). Both the positive and negative cases ($p_k > 0$ and $p_k \leq 0$) are shown for each value

(unsaturated) τ_1. On the other hand, τ_3 equals τ_2 in the positive case but keeps leading to corrections on the model also in the negative case. In Fig. 2b, the different values of τ_n are plotted for two different values of p_k as a function of C. Note that these two values of p_k lead to two different asymptotic values of the form $v = \frac{p_k}{1+\|(x_i - x_j)(x_i - x_j)^\top\|^2_{\text{Fro}}}$. It can be observed that τ_3 is an approximation of τ_0 for large values of C. In both illustrations it can be observed that τ_1 and τ_2 may lead to either stronger or milder corrections depending on the prediction values and the parameter C.

3.4 Constraint satisfaction and tuning

The particular performance of the above algorithms may depend on two very important practical considerations related to PSD constraint satisfaction and tuning. It has been commented in previous sections that PSD is enforced after the optimization step is performed. In particular, the only negative eigenvalue is neglected as soon as it appears at each iteration. This leads to a reasonable approximation to the original problem [10]. Nevertheless, it is also possible to postpone the PSD corrections for a fixed number of steps or even until the end of the iterative process [18]. This strategy has an obvious computational benefit since even with specialized incremental algorithms, recomputing the eigendecomposition of the metric matrix is a computationally expensive operation. Apart from this, allowing indefinite matrices in the iterative optimization process leads in practice to a kind of concentration of information in the negative eigenvectors that are implicitly neglected. We have experimented here with two possibilities for each online metric learning algorithm. We call these the positive (marked with symbol +) and negative (denoted by symbol -) versions. In the former case, PSD corrections are applied at each iteration. In the latter, indefinite matrices are left as such until the end of the whole process. Indeed, much sparser metric matrices are produced in the negative versions, with a prediction power slightly below their positive counterparts. Other options (whose behavior is not reported in this paper) lead to performance results and computational burdens that lie somewhere in between the two previous ones.

Another very important question involves the appropriate tuning of the aggressiveness parameter, C. The optimal value of C may depend on many different aspects, such as how separable the problem is. In contrast to the same concept in binary classification, this separability in distance values is difficult to analyze and manage. All these considerations get worse as the optimal value of C may depend on the particular pair because in the online learning case we have a different optimization problem at each iteration. In the present work, the optimal value of parameter C has been fixed for all iterations but in a different way for each algorithm. To this end, a quick round on a validation set has been performed, and the parameter that leads to the best results has been selected. Two alternative criteria have been considered, namely a) the value that minimizes the accumulated loss and b) the one that maximizes classification performance of the learned metric matrix when used in the k-nearest neighbor rule using the best k in the range from 1 to 25.

It is also worth mentioning the dependence of the result on the initial model given to the online algorithms. In general, one can think of starting from a random model, a previous one (e.g. obtained using batch learning on a reduced sample) or use the empty (zero) model. Using a previously learned model may accelerate finding a good solution but may seriously bias the behavior of the online algorithm. A random model may force the algorithm to start far away from optimal regions of the solution space but conditioning the sparseness

Algorithm 1 Passive aggressive online metric learning

Require: $\{x_i \mid i \in \mathbb{N}\}$, $P \subseteq \mathbb{N} \times \mathbb{N}$ //set of objets and pairs of indices
Require: $\{(x_i, x_j; y_{ij}) \mid (i, j) \in P\}$ //training pairs with similarity labels
Require: M^0, b^0,C. //initial model and aggressiveness parameter
Require: version, $tolerance$. //version (+/−) and tolerance on aggressiveness
Ensure: M, b. //learned metric model

$k = 0$
repeat
  $(i, j) \leftarrow$ GetNewPairFrom(P)
  Set τ according to Eqs. (12), (13) or (16).
  if $|\tau| \geq tolerance$ **then**
    $M^{k+1} \leftarrow M^k - \tau y_{ij}(x_i - x_j)(x_i - x_j)^\top$
    $b^{k+1} \leftarrow b^k + \tau y_{ij}$
    $k \leftarrow k + 1$
    if version = (+) **then**
      $(M^k, b^k) \leftarrow$ApplyConstraints(M^k, b^k)
    end if
  else
    $M^{k+1} \leftarrow M^k$
    $b^{k+1} \leftarrow b^k$
    $k \leftarrow k + 1$
  end if
until convergence

if version = (−) **then**
  $(M^k, b^k) \leftarrow$ApplyConstraints(M^k, b^k)
end if

$(M, b) \leftarrow (M^k, b^k)$

of the model. Despite the obvious disadvantages of using the empty model, this scheme offers the advantage of starting from a neutral, sparse and unbiased model. Moreover, in this case, convenient bounds on the accumulated loss of the model can be proved [10].

Finally, the algorithmic scheme which is common to the three online algorithms considered both in their positive or negative versions is shown in Algorithm 1.

4 Experiments and results

In order to compare and assess the relative benefits and disadvantages of the online algorithms considered in this paper, an exhaustive experimentation has been carried out. To this end, a number of experiments focused on showing their behavior as online processes have been designed. Also, different metrics have been used in combination with k-nearest neighbor rules to evaluate the goodness of the feature space in the classification task. In addition, the computational load of each method has been stored in order to show and compare their execution in a fair setup.

4.1 Datasets

To conduct the experimental evaluation, a total of 15 databases have been used. In particular, 12 different databases from [20]; 2 databases (spam, balance) from [21]; and one more realistic repository previously used in CBIR tasks [22] have been used. To clarify some abbreviations, databases soybean are represented as soyS and soyL. These refer to soybean Small and soybean Large, respectively. Also the morphological features in the digits dataset is called mor. Art100 refers to a commercial collection called "Art Explosion", distributed by the company Nova Development[1]. Images in this collection were manually classified according to subjective image similarity, according to judgments issued by real users. To perform more meaningful classification experiments, only classes with more than 100 elements have been considered in this collection. In all databases, objects are considered similar only if they share the same class label. A summary of the particular characteristics of each dataset used in the comparative study is shown in Table 1. Repositories have been approximately sorted in order of relative complexity, by considering both the number of samples and the dimensionality. For consistency reasons, this order has been respected in all figures and tables that refer to a same experiment on different datasets.

In the experiments, all databases have been split randomly into two equally sized disjoint subsets. These are used for training and testing, respectively.

4.2 Experimental settings

The experimentation setup in this work has been fixed as suggested in [5] and other previous preliminary studies [12]. Also, the Information theoretic metric learning algorithm (ITML) [5] has been considered in this work as a baseline for comparison purposes. The ITML algorithm has been used as suggested in [5], using the software made available by the authors that has its own parameter tuning mechanism. This algorithm has been shown to competitively compare to many other recent metric learning algorithms and can be considered as a good representative of the state of the art.

To ensure that all the methods considered (including ITML) use the same amount of information, a set P composed of $r = 40c(c-1)$ non-repeated, random pairs has been selected for training (with c the number of classes in the dataset). To cope with the diversity in the datasets and guarantee a fair comparison, a simple method to stop the online algorithms has been used. This method simultaneously accounts for the number of classes and the total number of samples of each specific dataset. In particular, the set P is shuffled and all pairs are sequentially provided to the algorithm. The set is given to the algorithm at least twice and then this process is repeated until a maximum number of iterations has been reached. This maximum number of iterations has been established as the minimum between

[1] http://www.novadevelopment.com.

Table 1 Characteristics of databases: size (n), dimension (d), number of classes (c) and number of training pairs used (r)

	soyS	wine	glass	ecoli	malaysia	iono	balance	breast	chromo	mor	spam	satellite	soyL	Art100	nist16
n	136	178	214	272	291	351	625	683	1143	2000	4601	6435	266	1710	2000
d	35	13	9	7	8	34	4	9	8	6	57	36	35	104	256
c	4	3	4	3	20	2	3	2	24	10	2	6	15	10	10
r	480	240	480	240	15200	80	240	80	22080	3600	80	1200	8400	3600	3600

20 % of all possible pairs of training samples (as in previous studies [17]) and 50 times the number of pairs in P (which is by far more than enough for larger databases with small number of classes). This means that at least t steps are executed, with $t = \min(\lfloor \frac{n(n-2)}{40} \rfloor, 50r)$. This value has been proved as a good trade-off between computational cost and performance.

In order to automatically tune an appropriate value for C, each fixed set of pairs is used to train a model for each version of each online method. This model is trained by fixing $t = 40c(c-1)$ which corresponds to feeding each pair only once. Besides, the final model is validated over the whole training set.

An appropriate range of exponentially spaced values in the range $[10^{-4}, 10^{2}]$ has been considered as the parameter space. Both criteria described in Sect. 3.4 have been attempted, and the two criteria led to very similar results. Hence, only results obtained by maximizing classification performance are reported in the present work.

All online models have been initialized with an empty model, that is $b = 0$ and a zero metric matrix as suggested in [11]. All results presented are the average of ten independent runs with different random initializations to obtain different training and test sets, but taking care of using exactly the same data for each one of the algorithms considered.

4.3 Performance evaluation

4.3.1 Online predictive comparison

Apart from using the algorithms to obtain a metric matrix for distance-based classification, their behavior as online learning methods has been assessed in the experiments. To this end, several loss and performance measures have been considered throughout the learning process. First, Fig. 3 shows the averaged predictive 0-1 loss defined as

$$\ell = \frac{1}{t}\sum_{k=1}^{t} \mathrm{sgn}(\ell_k) = \frac{1}{2t}\sum_{k=1}^{t} \left| y_k - \hat{y}_k \right|,$$

where t is the learning sequence length, y_k is the true label of the pair supplied at the kth step, and $\hat{y}_k$ is the label predicted by using the $(k-1)$th model.

This measure illustrates the behavior of the different online algorithms throughout time when discriminating between similar and dissimilar objects. Only 6 out of the 15 databases considered are shown but they are representative of all different behaviors observed in the whole set of experiments. In particular, the behavior on the databases nist16 and spam are very similar to the one shown for ionosphere. This similarity is also observed for ecoli, malaysia, mor and satellite with regard to Art100, which is shown. Databases chromo and breast show the same behavior as that of wine and glass, respectively. Finally, soybean small (soyS) and soybean large (soyL) also exhibit a very similar behavior.

Summarizing, all online learning algorithms have lead to reasonably good behavior in the experiments, according to the loss measure specified above. In 12 out of 15 databases, the negative versions of the methods have led to better results than their positive counterparts. And in five of them the worst negative version is better than all positive ones (glass, ionosphere, breast, spam and nist16). On the other hand, LS methods have exhibited a significantly worse behavior in the wine, chromo, balance, ecoli, mor, satellite, malaysia,soybean and Art100 databases. This difference in behavior is even more evident in the positive versions of the algorithms.

4.3.2 Dimensionality behavior evolution

To better understand how the online algorithms behave, the averaged effective dimension of the metric matrix as the optimization goes on is shown in Fig. 4 for the particular case of the ionosphere database. It can be observed that all online algorithms (which start from a null matrix) very quickly incorporate new dimensions and rapidly converge to a particular dimensionality.

The difference between positive and negative versions rely on the fact that negative versions converge to a sparser matrix because (hopefully unimportant) information concentrates in the negative eigenvectors of the corresponding metric matrix. Figure 5a and b display how the final metric matrices behave when used in a 2-NN classifier using train and test data, respectively. These two figures illustrate that the behavior on training data is representative of what happens with test data. In both cases, the matrices obtained with the positive and negative versions still contain uninformative dimen-

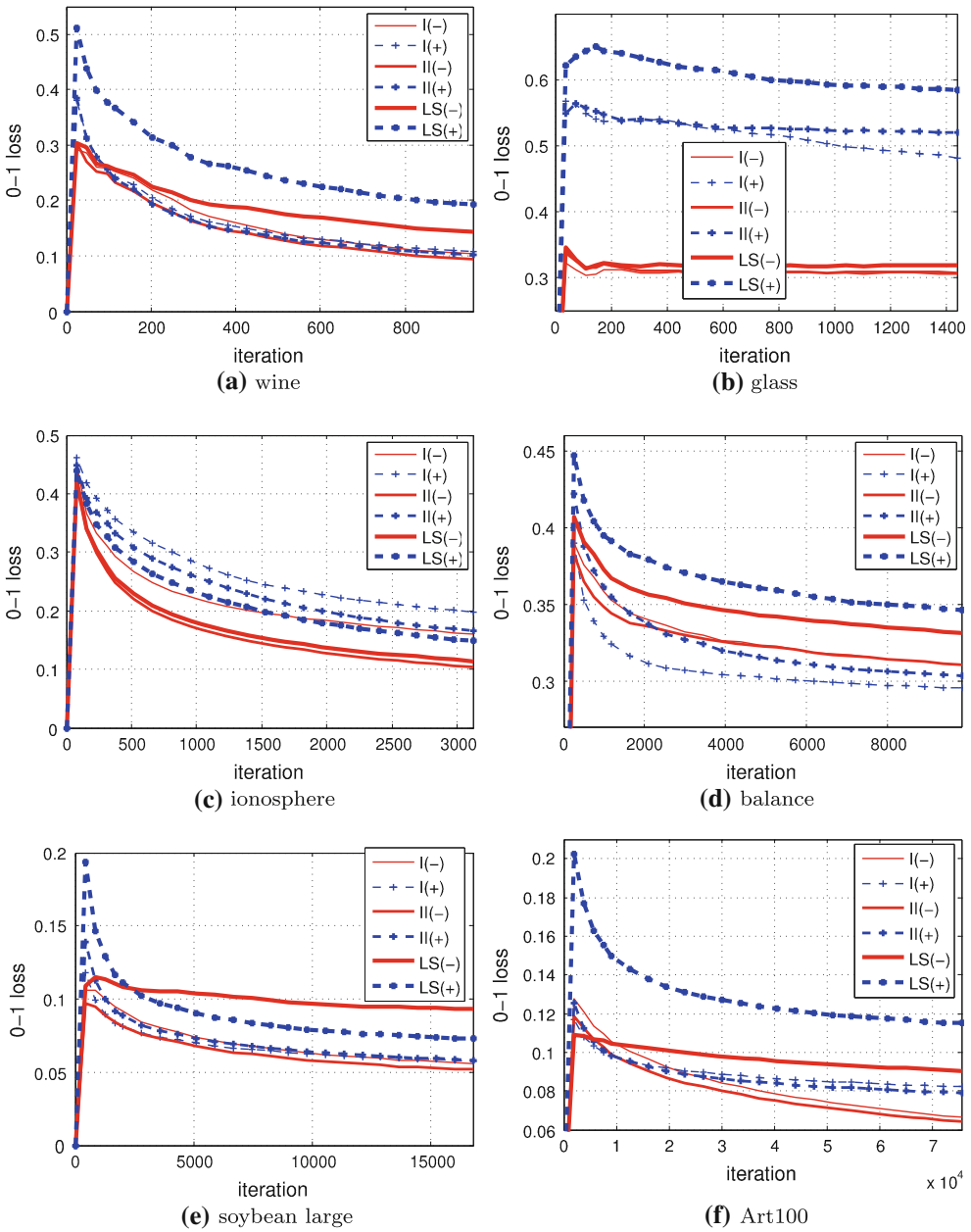

Fig. 3 Predictive error of the online algorithms

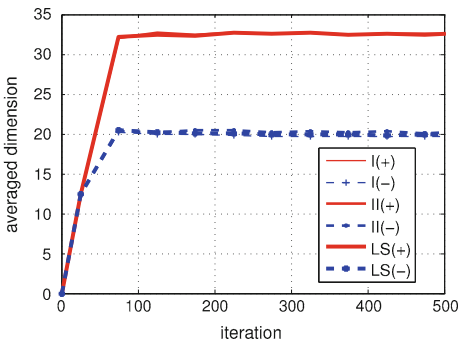

Fig. 4 Averaged effective dimension obtained using online algorithms at each iteration on the ionosphere database

sions. It can be seen that roughly the same results could be obtained if the matrices were cut to the best ten eigenvectors. The differences among methods are not significant and the absolute minimum seen on the training data does not have an exact correspondence in the test data. Even so, the behavior with training data can be considered as a good indicator about the intrinsic dimensionality for this problem.

4.3.3 Classification task

To measure the quality of the final outcome of the different algorithms, the corresponding matrices, M, have been used to construct a k-NN classifier. The classification error using

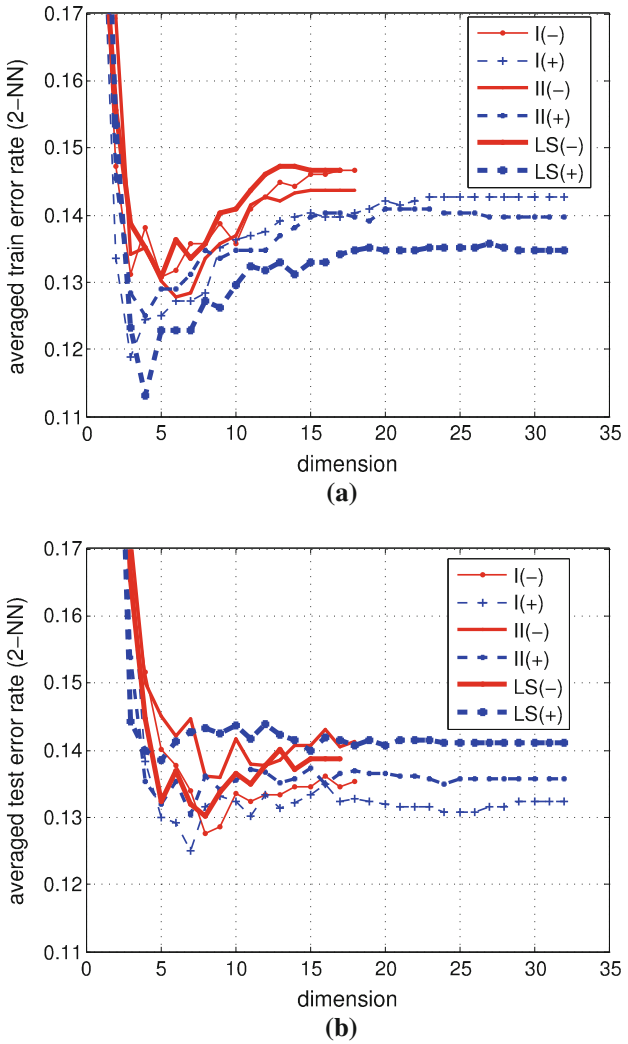

Fig. 5 Averaged 2-NN classification errors obtained as more eigenvectors from the obtained metric matrices are considered in order of importance for the experiments in Fig. 4

up to the first 25 neighbors has been computed and the best results for each database and method are shown in Table 2.

The classification errors obtained with all algorithms including ITML indicate a good classification performance in the context of the experimentation carried out in this work. A multiple comparison Friedman test [23] has not revealed any significant differences between the classification results obtained with ITML and any of the six other online alternatives proposed. Nevertheless, it can be concluded that all algorithms lead to very competitive results. It is also worth noting the fact that the combination of the LS approach with indefinite matrices has led to the best results in three out of fifteen databases. These are precisely the three largest ones, according to both size and dimensionality. In addition, using the Euclidean distance was the best option for one of the databases, ecoli, which provides an example where DML does not help improving classification results.

4.4 Computational burden measures

All experiments on all databases have been run on the same computer. In particular, an AMD Athlon(tm) 64 X2 Dual Core Processor 4200+ has been used. All different runs have been restricted to using a single CPU to obtain more accurate and machine-independent measurements.

Averaged CPU times in seconds for all databases, along with their corresponding standard deviations are shown in Table 3. The same running times have also been plotted in Fig. 6, using a logarithmic scale in the time axis to graphically illustrate the relative merits of each algorithm on each database. For completeness, and to better evaluate how the CPU time of the different algorithms compare to each other in all databases, a multiple comparison Friedman test has also been performed in this case. The resulting average ranks (from faster to slower) are shown in Table 4. The more important pairwise comparisons among the methods and their adjusted p values according to the Holm post-hoc test are given in Table 5.

From the running times shown in Table 4, one can conclude that all online algorithms are quite computationally competitive with regard to ITML. The lowest running times are always obtained when using the negative versions of the online algorithms. Significant differences have been found between the positive and negative versions of a same method. This occurs even in the case of PALS that needs about twice the number of updates than the other online algorithms. Significant differences also exist between all negative versions and the ITML. These differences increase with the size of the database. It is worth noting that in all cases, the CPU time of the online negative version is below 40% of that taken by the ITML algorithm. Finally, the negative version of the PALS algorithm is considerably slower than the other negative online algorithms in the two largest databases. In particular, it is about 3 times slower for nist16 and twice as slow in the case of Art100. On the contrary, CPU times for all negative online algorithms were very similar in all other databases.

5 Concluding remarks and further work

Several online learning algorithms adapted to solve the problem of learning a Mahalanobis-like distance matrix have been considered in this work. These algorithms are derived from the well-known passive-aggressive schema in which a term that measures closeness to the current learned model gets mixed with a term that enforces the particular constraints. All of them have been formulated under a common framework that has been extended to substitute the soft margin criterion by a least square condition.

In addition, both positive and negative versions of the online algorithms have been exhaustively tested. In the first

Table 2 Average classification errors and best number of neighbors (in brackets)

	Euclidean	ITML	PAI+	PAI-	PAII+	PAII-	PALS+	PALS-
soyS	0.158 (1)	**0.119** (2)	0.133 (14)	0.125 (13)	0.136 (12)	0.121 (15)	0.125 (9)	0.126 (12)
wine	0.027 (13)	0.033 (3)	0.018 (5)	0.017 (6)	**0.016** (3)	0.017 (9)	0.024 (7)	0.019 (3)
glass	0.510 (1)	0.513 (1)	0.523 (1)	**0.506** (1)	0.518 (1)	0.527 (1)	0.507 (1)	0.518 (1)
ecoli	**0.064** (3)	0.077 (5)	0.075 (3)	0.075 (17)	0.076 (15)	0.075 (7)	0.076 (17)	0.070 (9)
malaysia	0.301 (1)	0.298 (1)	0.281 (1)	0.289 (1)	**0.281** (1)	0.289 (1)	0.327 (1)	0.318 (1)
iono	0.153 (2)	0.155 (2)	**0.129** (2)	0.136 (2)	0.14 (2)	0.139 (2)	0.143 (2)	0.138 (2)
balance	0.335 (6)	0.263 (3)	**0.219** (3)	0.219 (2)	0.258 (3)	0.234 (2)	0.312 (6)	0.258 (6)
breast	0.033 (3)	0.037 (3)	**0.025** (9)	0.026 (11)	0.027 (17)	**0.025** (11)	0.028 (9)	0.030 (9)
chromo	**0.467** (1)	0.489 (1)	0.470 (1)	0.489 (1)	**0.467** (1)	0.488 (1)	0.476 (1)	0.481 (1)
mor	0.277 (9)	0.273 (12)	0.296 (12)	0.299 (10)	0.294 (10)	0.297 (8)	**0.269** (11)	0.279 (10)
spam	0.115 (1)	**0.109** (3)	0.118 (5)	0.12 (3)	0.119 (5)	0.121 (3)	0.12 (3)	0.121 (1)
satellite	0.119 (3)	**0.118** (3)	0.137 (4)	0.132 (3)	0.132 (3)	0.127 (3)	0.132 (3)	0.127 (3)
soyL	0.136 (1)	0.086 (1)	0.107 (1)	0.1 (1)	0.099 (1)	0.097 (1)	0.098 (1)	**0.083** (1)
Art100	0.219 (6)	0.203 (6)	0.209 (7)	0.217 (6)	0.207 (7)	0.214 (7)	0.203 (9)	**0.202** (9)
nist16	0.057 (1)	0.057 (1)	0.057 (1)	0.059 (1)	0.057 (1)	0.060 (1)	0.066 (1)	**0.056** (1)

Best result for each database is shown in bold

Table 3 Averaged CPU running times in seconds along with standard deviations (in brackets)

	ITML	PAI+	P0AII+	PALS+	PAI-	PAII-	PALS-
soyS	10.56 (12.08)	0.30 (0.01)	0.30 (0.04)	0.49 (0.05)	0.11 (0.02)	**0.10** (0.05)	0.11 (0.04)
wine	0.72 (1.17)	0.18 (0.03)	0.15 (0.04)	0.29 (0.02)	0.10 (0.00)	**0.10** (0.00)	0.10 (0.00)
glass	0.53 (1.28)	0.26 (0.04)	0.25 (0.04)	0.32 (0.03)	**0.08** (0.06)	0.09 (0.07)	0.11 (0.08)
ecoli	0.85 (1.08)	0.24 (0.07)	0.28 (0.08)	0.32 (0.03)	**0.05** (0.01)	0.05 (0.01)	0.06 (0.01)
malaysia	4.02 (0.12)	1.25 (0.06)	1.24 (0.06)	2.06 (0.04)	1.15 (0.10)	1.10 (0.10)	**1.09** (0.22)
iono	2.15 (0.69)	0.80 (0.06)	0.84 (0.13)	1.37 (0.11)	**0.28** (0.03)	0.29 (0.04)	0.34 (0.04)
balance	2.11 (1.17)	0.46 (0.02)	0.46 (0.02)	0.58 (0.01)	**0.28** (0.01)	0.28 (0.01)	0.30 (0.00)
breast	0.29 (0.45)	0.57 (0.12)	0.63 (0.14)	0.83 (0.12)	**0.26** (0.10)	0.27 (0.12)	0.35 (0.09)
chromo	6.39 (0.30)	1.99 (0.11)	1.96 (0.09)	2.65 (0.28)	**1.71** (0.07)	1.80 (0.07)	1.86 (0.15)
mor	4.08 (2.30)	4.77 (0.23)	4.72 (0.24)	8.00 (0.29)	**3.76** (0.19)	3.81 (0.18)	4.23 (0.14)
spam	0.73 (0.57)	12.51 (1.08)	12.68 (1.31)	17.69 (1.17)	0.54 (0.07)	**0.50** (0.07)	0.60 (0.01)
satellite	3.99 (7.65)	39.86 (2.40)	34.93 (5.20)	99.60 (0.42)	2.12 (0.09)	**2.07** (0.05)	2.46 (0.06)
soyL	26.93 (0.97)	4.48 (1.03)	4.55 (1.01)	26.47 (2.37)	0.86 (0.12)	**0.85** (0.09)	1.07 (0.04)
Art100	18.76 (0.12)	39.00 (1.30)	43.07 (3.20)	177.42 (18.85)	**4.32** (0.06)	4.58 (0.32)	6.55 (0.06)
nist16	86.09 (46.08)	480.23 (17.15)	500.55 (19.73)	2007.20 (14.47)	**10.13** (1.02)	10.81 (1.67)	34.06 (1.09)

Best results for each database are shown in bold

case, the PSD constraint on the learned matrix has been enforced at each iteration; in the second, the PSD constraint has not been considered when learning the matrix. From the classification results obtained, it can be safely stated that all the online algorithms considered in this work have the potential to arrive at very similar and competitive results with regard to the state of the art. In fact, no significant differences in classification performance where found between any of the online algorithms tested. On the other hand, the running times needed by the different algorithms vary depending on the size of the database. For small sizes, all online algorithms are very efficient. For larger databases, the positive versions lead to a relatively high computational effort. Differences in running time between the positive version of an algorithm and its negative counterpart have been shown significant in all cases.

A major contribution of this work is the proposal to use indefinite matrices in distance learning, as an efficient alternative to the so called alternating projection methods. Apart from the computational benefits obtained, results suggest that the use of indefinite matrices yield more sparse solutions. This leads to a recommendation to use the negative

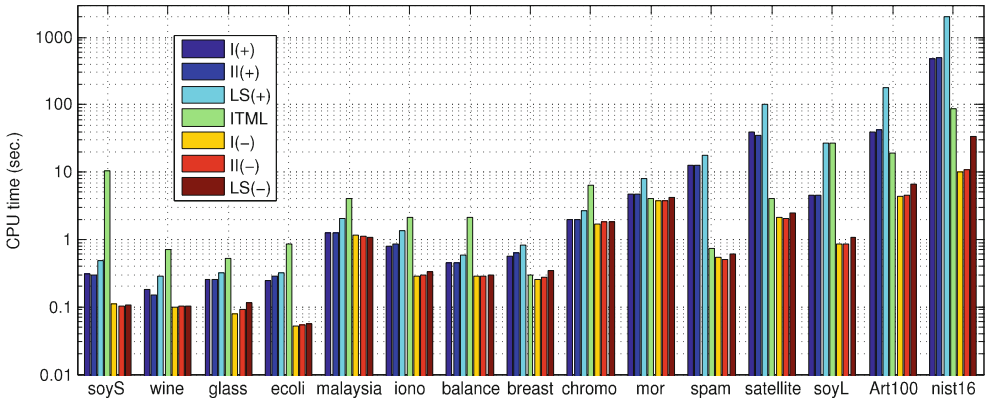

Fig. 6 Averaged CPU running times for each algorithm in all databases

Table 4 Average ranks of the methods according to the Friedman test ($\alpha = 0.05$)

Methods	Average rank
PAI-	1.49
PAII-	1.77
PALS-	2.87
PAII+	4.86
PAI+	4.93
ITML	5.67
PALS+	6.40

Table 5 Results of the Holm post hoc test ($\alpha = 0.05$)

	ITML	PAI+	PAII+	PALS+
PAI-	$<\mathbf{10^{-5}}$	$<\mathbf{10^{-3}}$	$<\mathbf{10^{-3}}$	$<\mathbf{10^{-7}}$
PAII-	$<\mathbf{10^{-4}}$	$<\mathbf{10^{-3}}$	**0.0011**	$<\mathbf{10^{-7}}$
PALS-	**0.0046**	0.0967	0.1112	$<\mathbf{10^{-3}}$

Rejected null hypotheses at the level α are marked in bold

versions of the algorithms that offer significantly lower running times while exhibiting roughly the same performance.

A number improvements on the presented algorithms are still possible. First, the convergence criterion could be further improved to reduce the training times of all online algorithms. Moreover, introducing an adaptive tolerance on the online process could also lead to a significantly lower number of matrix updates. This could have a significant impact, specially in the case of PALS algorithms. More importantly, we are considering adding specific constraints to control the sparseness of the matrix in order to both reduce computation time and improve the quality of the learned metrics.

Appendix: Derivation of the PA-LS update

As stated in Eqs. (14) and (15), the least squares formulation of the original PA problem leads to an optimization problem with only an equality constraint.

$$\min_{M,b,\xi} \frac{1}{2}\|M - M^k\|_{\text{Fro}}^2 + \frac{1}{2}\left(b - b^k\right)^2 + C\xi^2,$$
$$s.t. \quad 1 - y_{ij}\left(b - d_{ij}^M\right) = \xi.$$

The corresponding unconstrained minimization problem is obtained by introducing a Lagrange multiplier, τ, to arrive at the following Lagrangian

$$\mathcal{L}(M, b, \xi, \tau) = \frac{1}{2}\|M - M^k\|_{\text{Fro}}^2 + \frac{1}{2}(b - b^k)^2 + C\xi^2 + \tau\left(1 - y_{ij}(b - d_{ij}^M) - \xi\right). \quad (17)$$

Differentiating the Lagrangian with respect to M, b, ξ and setting these partial derivatives to zero leads to

$$M = M^k - \tau y_{ij}(x_i - x_j)(x_i - x_j)^\top, \quad (18)$$
$$b = b^k + \tau y_{ij}, \quad (19)$$
$$2C\xi = \tau \Longrightarrow \xi = \frac{\tau}{2C}. \quad (20)$$

Now we can use (18), (19) and (20) in the Lagrangian (17) to obtain

$$\mathcal{L}(\tau) = -\frac{\tau^2}{2}\|(x_i - x_j)(x_i - x_j)^\top\|_{\text{Fro}}^2 - \frac{\tau^2}{2} - \frac{\tau^2}{4C} + \tau\left(1 - y_{ij}\left(b^k - d_{ij}^{M^k}\right)\right). \quad (21)$$

Taking the derivative of $\mathcal{L}(\tau)$ with respect to τ and setting it to zero will give us its only critical point that corresponds to the minimum of the above problems.

$$\tau = \frac{1 - y_{ij}\left(b^k - d_{ij}^{M^k}\right)}{1 + \frac{1}{2C} + \|(x_i - x_j)(x_i - x_j)^\top\|_{\text{Fro}}^2}.$$

Note that this expression has been introduced in Eq. (16) as τ_3.

Equations (18) and (19) with this value of τ lead to the solution of the PALS problem.

References

1. Ii, R.D.S., Fukunaga, K.: The optimal distance measure for nearest neighbor classification. IEEE Trans. Info. Theory **27**(5), 622–626 (1981)
2. Fukunaga, K., Flick, T.E.: An optimal global nearest neighbor metric. IEEE Trans. Pattern Anal. Mach. Intell. **6**(3), 314–318 (1984)
3. Vapnik, V.N.: The nature of statistical learning theory. Springer-Verlag New York, Inc., New York (1995)
4. Scholkopf, B., Smola, A.J.: Learning with Kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge (2001)
5. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Ghahramani, Z. (ed.) ICML. Volume 227 of ACM international conference proceeding series., http://www.cs.utexas.edu/users/pjain/itml/, ACM pp. 209–216 (2007)
6. Globerson, A., Roweis, S.T.: Metric learning by collapsing classes. In: Weiss, Y., Scholkopf, B., Platt, J. (eds) Advances in neural information processing systems 18, MIT Press, Cambridge (2006)
7. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Weiss, Y., Scholkopf, B., Platt, J. (eds) Advances in neural information processing systems 18, MIT Press, Cambridge (2006)
8. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge University Press, New York (2004)
9. Shen, C., Kim, J., Wang, L.: Scalable large-margin Mahalanobis distance metric learning. Trans. Neur. Netw. **21**, 1524–1530 (2010)
10. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Brodley, C.E. (ed.) ICML. Volume 69 of ACM international conference proceeding series, ACM (2004)
11. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. J. Mach. Learn. Res. **7**, 551–585 (2006)
12. Perez-Suay, A., Ferri, F.: Online metric learning methods using soft margins and least squares formulations. In: Georgy Gimel'farb et al. (eds.) Structural, syntactic and statistical pattern recognition. Lecture Notes in Computer Science. vol. 7626, pp. 373–381. Springer, Verlag (2012)
13. Kwok, J., Tsang, I.: Learning with idealized kernels. Proceedings of the twentieth international conference on, machine learning, pp. 400–407 (2003)
14. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in neural information processing systems 16. MIT Press, Cambridge (2004)
15. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. J. Mach. Learn. Res. **10**, 207–244 (2009)
16. Pérez-Suay, A., Ferri, F.J.: Algunas consideraciones sobre aprendizaje de distancias mediante maximizacin del margen. In: Actas del V Taller de Minera de Datos y Aprendizaje (TAMIDA'2010), Valencia, CEDI 2010 pp. 83–91 (2010)
17. Perez-Suay, A., Ferri, F.J., Albert, J.V.: An online metric learning approach through margin maximization. In: Vitrià, J., Sanches, J.M.R., Hernández, M. (eds.) IbPRIA. Volume 6669 of Lecture Notes in Computer Science, Springer pp. 500–507 (2011)
18. Chechik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. J. Mach. Learn. Res. **11**, 1109–1135 (2010)
19. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Process Lett. **9**, 293–300 (1999)
20. Duin, R.P.W.: Prtools version 3.0: A matlab toolbox for pattern recognition. In: Proceedings of SPIE, p. 1331 (2000)
21. Asuncion, D.N.: UCI machine learning repository (2007)
22. Arevalillo-Herráez, M., Ferri, F.J., Domingo, J.: A naive relevance feedback model for content-based image retrieval using multiple similarity measures. Pattern Recogn. **43**(3), 619–629 (2010)
23. Garca, S., Herrera, F.: An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J. Mach. Learn. Res. **9**, 2677–2694 (2008)