Online Metric Learning Methods Using Soft Margins and Least Squares Formulations*

Adrian Perez-Suay and Francesc J. Ferri

Dept. Informàtica, Universitat de València, Burjassot 46100, Spain
{Adrian.Perez,Francesc.Ferri}@uv.es

Abstract. Online metric learning using margin maximization has been introduced as a way to learn appropriate dissimilarity measures in an efficient way when information as pairs of examples is given to the learning system in a progressive way. These schemes have several practical advantages with regard to global ones in which a training set needs to be processed. On the other hand, they may suffer from a poor performance depending on the quality of the examples and the particular tuning or other implementation details. This paper formulates several online metric learning alternatives using a passive-aggressive schema. A new formulation of the online problem using least squares is also introduced. The relative behavior of the different alternatives is studied and comparative experimentation is carried out to put forward the benefits and weaknesses of each alternative.

1 Introduction

Organizing, classifying and/or representing sets of data is of key importance in many different application domains from fields like image analysis, pattern recognition or data mining. Distance-based methods form a well established group of approaches to tackle classification, regression, estimation and clustering problems. The performance of such methods, depends on the metric that relates input instances which is intimately tied to the way objects are represented.

In recent years, Distance Metric Learning (DML) has been an active area of research. In most cases, DML aims at learning an appropriate Mahalanobis-like distance matrix. Although there are many approaches, the most common is to define a (usually convex) criterion function which expresses the desired goal [1,2] which basically consists of keeping similar objects close and dissimilar ones far away at the same time. Determining the solution of such global optimization problems can be computationally expensive specially when dealing with large-scale problems [3]. Consequently, the need of effective and efficient learning methods has led to the emergence of sequential methods [4,5], mainly based on optimizing a convenient criterion over only one instance (pair of objects) that is made available for learning at every time step.

* Work partially funded by FEDER and Spanish Government through projects TIN2009-14205-C04-03, TIN2012-38604-C05-02 and Consolider Ingenio 2010 CSD07-00018.

G.L. Gimel' farb et al. (Eds.): SSPR & SPR 2012, LNCS 7626, pp. 373–381, 2012.

Unfortunately, many practical and theoretical problems arise. On one hand, different ways of sequentially enforcing additional constraints may lead to different solutions requiring different amount of computation. On the other hand, the performance of the final solution may deviate significantly from the ideal (global) goal depending of the particular instances used in the last iterations.

The present work jointly introduces a family of online metric learning algorithms which use margin maximization [4]. A novel formulation of the same online optimization problem is proposed using a least square formulation instead of the passive-aggressive schema [6]. Exhaustive comparative experimentation is carried out in order to fully characterize the advantages and drawbacks of each online algorithm with regard to other state of the art alternatives.

2 Online Metric Learning

Assume $\mathcal{R}^d$ is a real $d-$dimensional feature space and consider a set of points $\{x_i\}_{i=1}^N \in \mathcal{R}^d$, and a labeling function y_{ij} which indicates when a pair of points x_i, x_j is similar ($y_{ij} = 1$), or dissimilar ($y_{ij} = -1$). This labeling can come from a user or an appropriate oracle according to the practical problem at hand. A distance function (or simply distance) is a real function defined on $\mathcal{R}^d \times \mathcal{R}^d$ satisfying nonnegativity, identity and triangle equality. This function is a pseudo-metric if identity (it can be zero even for different objects) is not enforced. It is possible to represent a vast family of pseudo-metrics (including the Euclidean distance) by using a Positive Semi Definite (PSD) matrix M as:[1]

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

The main goal of metric learning is to obtain a matrix M that reflects the (dis)similarity between pairs of points, x_i, x_j, leading to appropriately different distance values depending on whether these points are really (dis)similar or not. An idealized situation can be visualized as having a convenient threshold value, b, in such a way that all similar distance values are under b and all dissimilar distance values are above b.

2.1 The Separable Case

A pseudo-metric function is better if the corresponding separation between (dis)similar distance values is bigger. This condition can be expressed as maximizing the margin around the threshold value, b. In the separable case and following a similar approach as with support vector machines [7], maximizing the margin can be turned into fixing a margin value and minimizing the (Frobenius) norm of the matrix M. Setting a fixed value of 2 between both kind of distance values can be compactly expressed as

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

[1] By convenience, we consider in this work squared versions of the distances.

Instead of considering constrained optimization using all information (examples) available, the above 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 [4,8] . Under this sequential scheme, at each step k, a particular model formed by the pair (M^k, b^k) is available to make a prediction over the labeled pair $t_k = (x_i, x_j, y_{ij})$ which is revealed to the system at this step. First, a prediction with the previous model (M^k, b^k) is made and a loss corresponding to the violation of this particular constraint is measured. In particular, the hinge loss is used

$$\ell_H(M, b, t_k) = \max\left\{0, 1 - y_{ij}(b - d_{ij}^M)\right\}. \tag{3}$$

Only in the case when the predictor (M^k, b^k) fails, i.e. the hinge-loss is greater than zero, $\ell_H(M^k, b^k, t_k) > 0$, the system is forced to retrain their current model. The aim is to find the nearest model $(\hat{M}^{k+1}, \hat{b}^{k+1})$ to the previous one that attains zero loss in the received pair (provided that it exists). This can be written as the following (online) optimization problem

$$(\hat{M}^{k+1}, \hat{b}^{k+1}) = \arg\min_{M,b} \frac{1}{2}\|M - M^k\|_{\mathsf{Fro}}^2 + \frac{1}{2}(b - b^k)^2, \tag{4}$$

$$s.t. \quad \ell_H(M, b, t_k) = 0, \tag{5}$$

where $\|\ .\ \|_{\mathsf{Fro}}$ is the Frobenius norm. As was shown in [4] the corresponding update becomes:

$$\hat{M}^{k+1} = M^k - \tau y_{ij}(x_i - x_j)(x_i - x_j)^\top, \tag{6}$$

$$\hat{b}^{k+1} = b^k + \tau y_{ij}, \tag{7}$$

where

$$\tau = \frac{\ell_H(M^k, b^k, t_k)}{1 + \|(x_i - x_j)(x_i - x_j)^\top\|_{\mathsf{Fro}}^2}. \tag{8}$$

Two additional constraints are needed in the definition of the problem [4]. The first one is that the matrix M must be Positive Semi Definite (PSD),that is, $M \succeq 0$ and consequently the threshold b must be above 1, $b \geq 1$. Since τ is nonnegative, the constraint $M \succeq 0$ is straightforwardly taken taken into account if $y_{ij} = -1$ when using the rank-one update in Eq. (6). In the case $y_{ij} = 1$, this update may introduce one negative eigenvalue in M at most. The closest PSD matrix can be found by setting this negative eigenvalue to zero after eigendecomposing M. Alternatively and equivalently, both the eigenvalue along with its corresponding eigenvector can be added to M after computing them using the Lanczos method [4].

2.2 Soft Margin Formulation

The previous formulation only makes sense if feasible solutions exist. Which means that similar/dissimilar pairs can be strictly separated. As in [6], different

alternatives for the inseparable case can be considered. In particular, the inseparable case can be solved by adding the slack variable $\xi_{ij} \in \mathcal{R}$ to the original formulation and enforcing its positiveness by adding a new constraint to the problem. This slack variable ξ_{ij} is weighted by a hyper-parameter $C \in [0, +\infty[$, that preserves the trade off between closeness to the previous model and loss minimization. Following the passive-aggressive approach [6] the corresponding online optimization problem can be stated as:

$$(M^{k+1}, b^{k+1}) = \underset{M,b,\xi_{ij}}{\arg\min} \frac{1}{2}\|M - M^k\|^2_{\mathsf{Fro}} + \frac{1}{2}(b - b^k)^2 + C\xi_{ij}, \tag{9}$$

$$s.t. \quad \ell_H(M, b, t_k) \leq \xi_{ij}, \quad \xi_{ij} \geq 0 \tag{10}$$

Alternatively, the objective function can be scaled quadratically with respect to ξ_{ij}. This fact avoids the need to restrict ξ_{ij} to be nonnegative. That is,

$$(M^{k+1}, b^{k+1}) = \underset{M,b,\xi_{ij}}{\arg\min} \frac{1}{2}\|M - M^k\|^2_{\mathsf{Fro}} + \frac{1}{2}(b - b^k)^2 + C\xi^2_{ij}, \tag{11}$$

$$s.t. \quad \ell_H(M, b, t_k) \leq \xi_{ij}. \tag{12}$$

These two formulations lead to different update rules that will be referred to as PA-I and PA-II, respectively as in [6]. These update rules are the ones in Equations (6) and (7) but changing the rate τ by τ_1 and τ_2, respectively.

$$\tau_1 = \min\left\{C, \frac{\ell_H(M^k, b^k, t_k)}{1 + \|(x_i - x_j)(x_i - x_j)^\top\|^2_{\mathsf{Fro}}}\right\}, \tau_2 = \frac{\ell_H(M^k, b^k, t_k)}{1 + \frac{1}{2C} + \|(x_i - x_j)(x_i - x_j)^\top\|^2_{\mathsf{Fro}}}. \tag{13}$$

2.3 Least Squares Formulation

An alternative formulation inspired on a Least-Squares approach [9] is also possible. Instead of forcing a soft margin and minimize the amount of violation on this condition, 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 can be formulated as:

$$(M^{k+1}, b^{k+1}) = \underset{M,b,\xi_{ij}}{\arg\min} \frac{1}{2}\|M - M^k\|^2_{\mathsf{Fro}} + \frac{1}{2}(b - b^k)^2 + C\xi^2_{ij}, \tag{14}$$

$$s.t. \quad 1 - y_{ij}(b - d^M_{ij}) = \xi_{ij}. \tag{15}$$

The main change in this formulation is that the inequality in the restriction (12) has been changed by an equality at restriction (15) and, the loss function is now a function which measures how far is the distance value from its corresponding idealized one ($b-1$ or $b+1$). This fact brings more aggressiveness to the problem formulation and the final number of updates will consequently expected to be greater. In fact, the update rule can be derived in a very similar way as in the previous cases leading to a different rate given by

$$\tau_3 = \frac{1 - y_{ij}(b^k - d_{ij}^{M^k})}{1 + \frac{1}{2C} + \|(x_i - x_j)(x_i - x_j)^\top\|_{\mathsf{Fro}}^2}. \tag{16}$$

Note that τ_3 can take negative values and it holds that $\tau_2 = \max\{0, \tau_3\}$.

The corresponding algorithm will be referred to as PA-LS here, although it cannot be properly considered as passive-aggressive because only in the particular case when it holds $1 - y_{ij}(b - d_{ij}^M) = 0$, the system does a passive step (do nothing) and this only occurs when the distance value takes exactly its desired value. This is in contrast with the above PA-I and PA-II approaches which perform a passive step in the case when $\ell_H = 0$ that corresponds to $1 - y_{ij}(b - d_{ij}^M) \leq 0$.

2.4 Tuning and Implementation Details

The performance of the different online learning algorithms using the above update rules strongly depend on the value of the parameter C (which needs to be adapted for each database) and also on the initial model given by M and b (that have been set to the zero matrix and 0 following considerations in [4]).

On the other hand, the PSD constraint needs to be enforced at each learning step but it is possible to relax this by allowing negative models during a fixed amount of learning steps after enforcing positiveness [3]. In fact, in our experiments we have obtained better results in general if the PSD constraint is not enforced until the end of the online learning process. Consequently, for each one of the above algorithms we consider a positive (+) version in which the PSD constraint (and $b \geq 1$) are enforced at each iteration, and a negative (-) one in which these constraints are only enforced at the end of the process.

3 Experiments and Results

In order to compare all the above online methods, an exhaustive experimentation has been designed. It has been mainly focused on classification, time execution and convergence-optimality trade off. Several different publicly available databases from [10,11] are taken into account. Moreover, a more realistic database previously used in CBIR tasks [12] has also been selected. This database is extracted from a commercial collection called "Art Explosion", distributed by the company Nova Development (`http://www.novadevelopment.com`). To perform more meaningful classification experiments, only classes with more than 100 elements have been selected. In all cases, objects are considered similar only if they share the same class label. Datasets used in the comparative study and their particular characteristics are shown in Table 1.

Table 1. Characteristics of Databases

	wine	ionosphere	balance	soybean	BDG100	nist16
Size	178	351	625	266	1710	2000
Dimension	13	34	4	35	104	256
no. of classes	3	2	3	15	10	10

Experimentation setup has been fixed as suggested in the work [5], where one of the most competitive metric learning algorithms has been introduced and studied. Precisely this algorithm, the Information Theoretic Metric Learning (ITML), has been adopted in the present study as a baseline. The ITML algorithm has been used as suggested in [5] using the software made available by the authors that has its own tuning mechanism which assigns appropriate parameters to the algorithm.

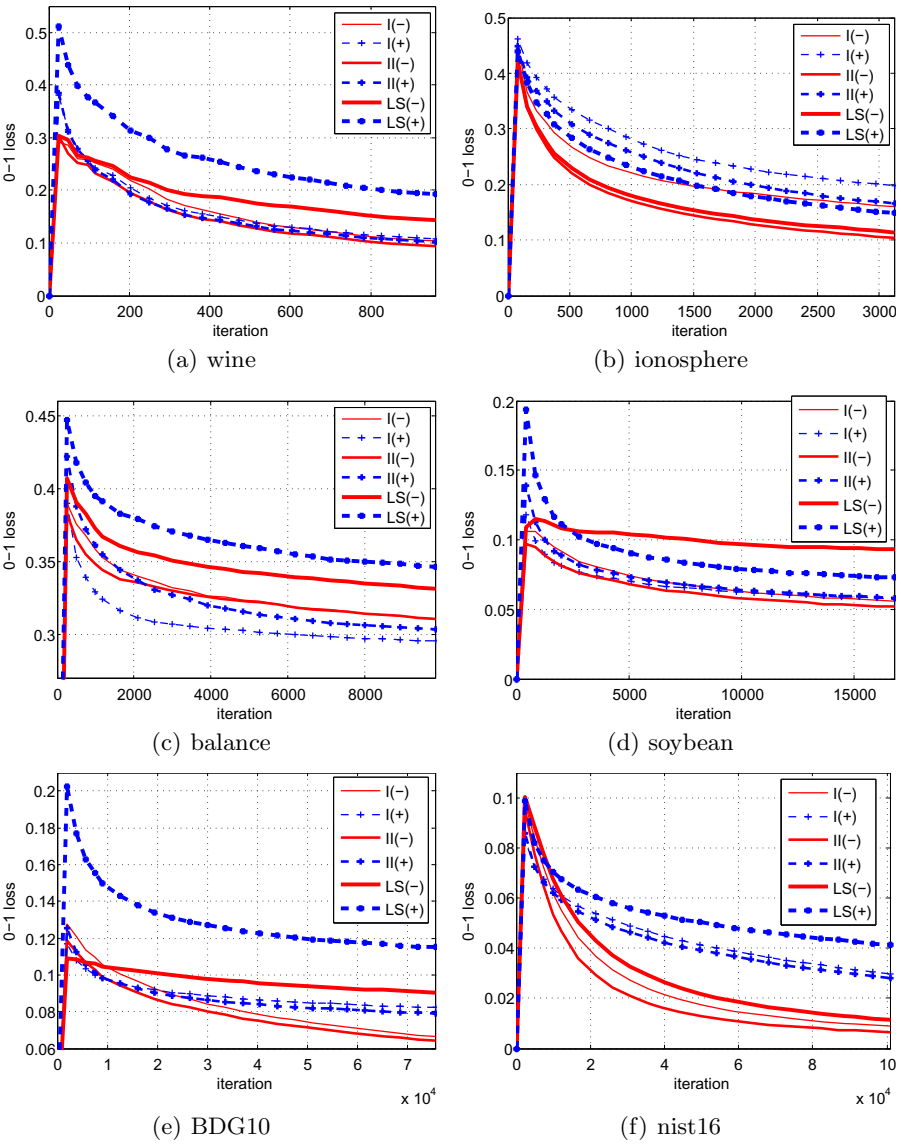
To study the behavior of all the online methods, initial values have been fixed as $(M^0, b^0) = 0$. In all cases, C is tuned using a validation set taking from the available training set. In particular, exponentially spaced values between $[10^{-4}, 10^2]$, have been considered. In order to feed all the methods considered with the same amount of information, a subset of $40c(c-1)$ pairs has been randomly selected for each run on each of the databases. The online algorithms are feed with random pairs from this subset. The subset of pairs is presented several times with different random order until the total number of time steps exceeds 20% of total number of pairs in each training set. As will be shown in the following, this amount of iterations has been proved as a good trade off between computational cost and performance. All results presented are the average of 10 independent runs with different random initializations but with exactly the same data for each one of the alternative algorithms.

To illustrate the behavior of the different approaches throughout the learning process, several loss and performance measures have been taken during the learning process. First, Figure 1 shows the predictive 0-1 loss defined as $\ell = \frac{1}{2T}\sum_{k=1}^{T} |y_k - \hat{y}_k|$, where T is the learning sequence length, k represents the corresponding pair supplied at $(k+1)$-th step and y_k and $\hat{y}_k$ are the true and predicted labels using the model (M^k, b^k). This measure illustrates the behavior of the different online algorithms throughout time when discriminating between similar and dissimilar objects.

All online learning algorithms have lead to reasonably good behavior in the experiments carried out according to this loss measure. In 5 out of 6 databases, negative methods have led to better results compared to positive versions of the same methods. The case of the two biggest databases is specially remarkable. On the other hand, the LS methods exhibit significantly worse behavior in wine, balance, soybean and BDG10 databases.

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 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, are all good classification results in the context of the experimentation carried out in this work. It must be noted that the differences among different classification results are not significant in most of the cases. Nevertheless, it can be concluded that all algorithms lead to very competitive results. Also worth noting is the fact that the combination of LS approach with negative matrices lead to the best results in the three last databases.



(a) wine (b) ionosphere

(c) balance (d) soybean

(e) BDG10 (f) nist16

Fig. 1. Predictive error of the online algorithms

Table 2. Average classification errors and best number of neighbors (in brackets). Best result for each database is shown in bold.

	ITML	PA-I$^+$	PA-I$^-$	PA-II$^+$	PA-II$^-$	PA-LS$^+$	PA-LS$^-$
wine	3.33(3)	1.76(5)	1.68(6)	**1.59(3)**	1.68(9)	2.42(7)	1.87(3)
ionosphere	15.46(2)	**12.87(2)**	13.59(2)	14.00(2)	13.87(2)	14.26(2)	13.84(2)
balance	26.27(3)	**21.88(3)**	21.92(2)	25.81(3)	23.40(2)	31.23(6)	25.79(6)
soybean	8.63(1)	10.73(1)	10.00(1)	9.87(1)	9.70(1)	9.77(1)	**8.27(1)**
BDG10	20.27(6)	20.93(7)	21.74(6)	20.72(7)	21.40(7)	20.25(9)	**20.20(9)**
nist16	5.65(1)	5.67(1)	5.86(1)	5.67(1)	6.03(1)	6.64(1)	**5.62(1)**

All experiments on all databases have been run using the same computer. In particular, an AMD Athlon(tm) 64 X2 Dual Core Processor 4200+ has been used but restricting the code to use only one CPU to obtain more accurate measurements (except for the two biggest databases). Figure 2 shows the relative averaged running time spent by each online learning algorithm with regard to the time used by the ITML algorithm.

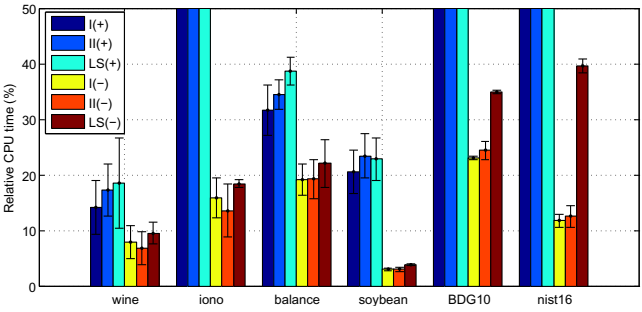


Fig. 2. Relative averaged execution CPU time with regard to ITML

From the running times shown, it can be seen that all online algorithms are very efficient compared to ITML. The case of positive versions on ionosphere, BDG10 and nist16 databases that get close to 200% is an exception. But more importantly, we see that the negative online algorithms reduce dramatically the time spent by their corresponding positive versions while preserving good performance results. The running times are kepts relatively low even in the case of PA-LS(-) that need about twice the number of updates with regard to the other negative online algorithms.

4 Concluding Remarks and Further Work

A family of online metric learning algorithms has been considered. The formulation using a soft margin and a passive-aggressive scheme has been extended by considering a least squares formulation. The algorithms have been implemented as positive versions in which the PSD constraint is enforced at each iteration,

and negative ones in which the constraint is enforced only at the end. Performance results show that under an appropriate implementation and tuning, all online methods are able to arrive at good results, but the negative versions are appealing due to their low running times. All online methods presented in this work still have room for improvement. In particular, the number of iterations can be adapted by introducing convergence criteria that are now under study. Also, with regard to LS methods, we are currently improving its running time by adding an updating tolerance when the value of τ_3 is close enough to zero.

References

1. Globerson, A., Roweis, S.T.: Metric learning by collapsing classes. [13]
2. Weinberger, K.Q., Blitzer, J., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. [13]
3. Chechik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. Journal of Machine Learning Research 11, 1109–1135 (2010)
4. Shalev-Shwartz, S., Singer, Y., Ng, A.Y.: Online and batch learning of pseudo-metrics. In: Brodley, C.E. (ed.) ACM International Conference Proceeding Series, ICML, vol. 69, ACM (2004)
5. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: Ghahramani, Z. (ed.) ICML. ACM International Conference Proceeding Series, vol. 227, pp. 209–216. ACM (2007), `http://www.cs.utexas.edu/users/pjain/itml/`
6. Shalev-Shwartz, S., Crammer, K., Dekel, O., Singer, Y.: Online passive-aggressive algorithms. In: Thrun, S., Saul, L.K., Schölkopf, B. (eds.) NIPS. MIT Press (2003)
7. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
8. Perez-Suay, A., Ferri, F.J., Albert, J.V.: An Online Metric Learning Approach through Margin Maximization. In: Vitrià, J., Sanches, J.M., Hernández, M. (eds.) IbPRIA 2011. LNCS, vol. 6669, pp. 500–507. Springer, Heidelberg (2011)
9. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Process. Lett. 9, 293–300 (1999)
10. Duin, R.P.W.: Prtools version 3.0: A matlab toolbox for pattern recognition. In: Proc. of SPIE, p. 1331 (2000)
11. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
12. Arevalillo-Herrez, M., Ferri, F.J., Domingo, J.: A naive relevance feedback model for content-based image retrieval using multiple similarity measures. Pattern Recognition 43(3), 619–629 (2010)
13. Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]. In: NIPS (2005)