About Combining Metric Learning and Prototype Generation*

Adrian Perez-Suay, Francesc J. Ferri,
Miguel Arevalillo-Herráez, and Jesús V. Albert

Dept. Informàtica, Universitat de València. Spain
{Adrian.Perez,Francesc.Ferri,Miguel.Arevalillo,Jesus.V.Albert}@uv.es

Abstract. Distance metric learning has been a major research topic in recent times. Usually, the problem is formulated as finding a Mahalanobis-like metric matrix that satisfies a set of constraints as much as possible. Different ways to introduce these constraints and to effectively formulate and solve the optimization problem have been proposed. In this work, we start with one of these formulations that leads to a convex optimization problem and generalize it in order to increase the efficiency by appropriately selecting the set of constraints. Moreover, the original criterion is expressed in terms of a reduced set of representatives that is learnt together with the metric. This leads to further improvements not only in efficiency but also in the quality of the obtained metrics.

1 Introduction

Classifying and/or conveniently representing high dimensional data has always been a very important goal in many different domains across the pattern recognition and image analysis fields. When the objects under study correspond to large collections of images or any other kind of visual information, this issue becomes even more critical due to the huge sizes usually involved. The classical approach for dealing with such high dimensional data is to apply some kind of dimensionality reduction in order to look for either numerical stability, performance improvement or simply to be able to get results in a reasonable amount of time [1, 2].

Dimensionality reduction has been largely studied from different points of view. In particular, linear methods such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are very well-known and commonly used in practice [1, 3]. Particular implementations and extensions of these have been proposed in particular domains such as face recognition [4–7].

The vast majority of approaches that propose using either linear or non linear dimensionality reduction to map the original problem into a (usually simplified) representation space, end up using a straightforward distance-based classification

* This work has been partially funded by FEDER and Spanish MEC through projects TIN2009-14205-C04-03, TIN2011-29221-C03-02 and Consolider Ingenio 2010 CSD2007-00018.

P. Fränti et al. (Eds.): S+SSPR 2014, LNCS 8621, pp. 323–332, 2014.

method in this space. The combination of the mapping and distance function can be seen as a composite (and possibly complex) metric in the original space. This puts forward the close relation that exists between dimensionality reduction and metric learning. Metric learning has received recent interest and has been tackled from very different viewpoints [8–11] using rather different methodologies to learn a convenient metric for a particular problem.

Basically, all methods that directly look for a (usually parameterized) distance function follow to some extent the same rationale that guides most (discriminant) dimensionality reduction approaches. This consists of increasing the effective distances between objects from different classes while decreasing the distances among objects of the same class. To this end, different approaches explicitly use distances either to define criteria or introduce constraints in the formulation along with different kinds of regularizers [8, 12, 13].

Regardless of the particular way of formulating the problem, one can distinguish between the criterion (and how exactly it relates to the ultimate goal of obtaining an appropriate metric), and the particular training information that is given to the algorithm (usually as sets of similar and dissimilar pairs). For example, different particular methods use different strategies to select this training information ranging from using all possible pairs [12] to pairs in the near vicinity of particular points [14]. More recently, it has been proposed to learn the best training pairs along with the metrics [15].

In this paper, we also propose a way of progressively adapting the training pairs along with the metric. Starting from the convex formulation used for MCML (Maximally Collapsing Metric Learning, [12]), we generalize it by introducing a reduced set of representative prototypes. With this generalization it is possible to obtain Mahalanobis like metrics that improve the results of the original algorithm, using a smaller amount of (selected) training pairs. Experimentation using several publicly available databases has been carried out to empirically validate the benefits of the proposed approach.

2 Metric Learning and Collapsing Classes

Given a collection of objects in a multidimensional vector space, $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$, let us consider distances parametrized by a positive semi-definite (PSD) matrix, A, as $d^A(x_i, x_j) = (x_i - x_j)^T A(x_i - x_j)$.

This quadratic distance (also referred to as Mahalanobis distance by analogy) is at the root of much recent work on metric learning in which the goal consists of appropriately estimating these matrices. As any PSD matrix can be decomposed as $A = W^T W$, using the above distance is equivalent to mapping the objects using W and then using the Euclidean distance on them.

The MCML algorithm [12], works by looking for a matrix A whose corresponding mapping makes all classes collapse into a single target point per class (which means null distances), which are arbitrarily far away from each other.

To construct a criterion that measures goodness with regard to the above idealized mapping, the following probability of x_i being *similar* (i.e. from the

same class in the context of the present paper) to any other x_j is introduced as follows.

$$p^A(j|i) = \frac{1}{Z_i} e^{-d_{ij}^A} = \frac{e^{-d_{ij}^A}}{\sum_{k \neq i} e^{-d_{ik}^A}}$$

where $d_{ij}^A = d^A(x_i, x_j)$. For each i, this is a discrete probability density function ranging for all j such that $i \neq j$.

The more classes collapse into a single point and are far away from each other, the closer these probabilities will be to the following target probability:

$$p_0(j|i) \propto \begin{cases} 1 & \text{if } x_i \text{ and } x_j \text{ belong to the same class} \\ 0 & \text{otherwise} \end{cases}$$

The Kullback-Leibler divergence can be used as an objective measure of how far we are from the goal of having all classes maximally collapsed. The criterion to be minimized is the above mentioned divergence averaged for all objects i, which can be written as [12]:

$$J_M(A, X) = \frac{1}{n} \sum_{i=1}^{n} p_0(j|i) \log \frac{p_0(j|i)}{p^A(j|i)}$$

This criterion can be changed to an equivalent one after obviating constant terms:

$$J(A, X) = \frac{1}{n} \sum_{i,j=1}^{n} p_0(j|i) d_{ij}^A + \frac{1}{n} \sum_{i=1}^{n} \log \sum_{k \neq i} e^{-d_{ik}^A}$$

When minimizing this criterion with regard to A, the problem becomes convex. This adds some guarantees for applying optimization methods based on the gradient, which is given by:

$$\nabla_A J(A, X) = \frac{1}{n} \sum_{i,j=1}^{n} (p_0(j|i) - p^A(j|i)) \cdot \nabla_A d_{ij}^A$$

where $\nabla_A d_{ij}^A = (x_i - x_j)(x_i - x_j)^T$.

The corresponding gradient descent algorithm is guaranteed to converge to a global optimum but is extremely inefficient in practice as it needs to perform $O(n^2)$ operations involving $O(d^2)$ matrices. Moreover, the PSD constraint on A needs to be enforced at each iteration which implies a further $O(d^3)$ computational burden per iteration.

In the rest of the paper we will restrict ourselves to moderate dimensional problems and will concentrate only in reducing the $O(n^2)$ cost as much as possible at the same time that the quality of the learned metric gets improved.

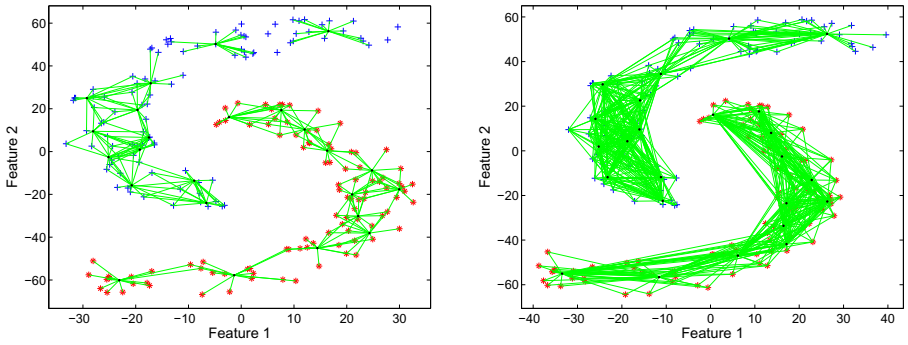

Fig. 1. Illustrative example displaying same-class neighborhood sets, $S_\beta(y_i)$, for a synthetic banana-shaped set for values $\beta = 0.15$(left) and $\beta = 0.4$(right). Different-class sets, $D_\beta(y_i)$ are not shown.

3 Maximally Collapsing Clusters around Representative Prototypes

One of the problems of MCML is related to the computational cost per iteration. A relatively straightforward way of alleviating this problem while maintaining the rationale of the method consists of considering a convenient set of *landmark* or *anchor* points to which distances are measured instead of the whole training set. The same idea has been largely used in the literature in different contexts [16, 17] and specifically for this very same problem [18].

The above expressions need to be rewritten in terms of the given training set, X, and a (reduced) landmark set $Y = \{y_i\}_{i=1}^m$. In both criterion and gradient expressions all terms $(x_i - x_j)$ must be substituted by $(y_i - x_j)$ and the i index must range now over the set Y. If the set Y is small but representative, the new criterion obtained is a good approximation of the original one.

Regardless of the way in which landmark points are obtained, the corresponding algorithm will be referred here to as MCMLA(α), where the A suffix refers to the use of anchors and the proportion $\alpha = \frac{m}{n}$ is the only parameter that controls the size of Y while maintaining the relative sizes of the classes as in X. In the particular case of $X = Y$, we have MCMLA(1) which matches the original MCML algorithm. For high proportion values, the behavior of the algorithm is very similar to MCML. On the other hand, the smaller its value, the more efficient the algorithm will be. Below a particular value of α which is problem dependent, the MCMLA algorithm usually deteriorates due to the poor representativity of the landmarks used with regard to the whole set X.

The first step of the new proposal consists of restricting the probability functions only to objects, x_j, in the close neighborhood of y_i. In fact, for each (fixed) landmark, y_i, we define the $(\beta \cdot n)$-nearest same-class neighbors, $S_\beta(y_i)$, and the

$(\beta \cdot n)$-nearest different-class neighbors, $D_\beta(y_i)$, and then redefine the probabilities as

$$p^A(j|i) = \frac{1}{Z_i} e^{-d^A(y_i, x_j)}, \quad \forall x_j \in N_\beta(y_i)$$

where Z_i must be redefined accordingly and $N_\beta(y_i) = S_\beta(y_i) \cup D_\beta(y_i)$. The parameter $\beta \in (0, 1]$ is a proportion over the size of X that controls the size of the neighborhoods around each landmark. As far as the size of the neighborhoods is fixed, it is straightforward to redefine the above criteria and gradient which will be written now as $J_M(A, Y, X, N_\beta)$, $J(A, Y, X, N_\beta)$ and $\nabla_A J(A, Y, X, N_\beta)$, respectively.

Figure 1 illustrates the S_β sets for two different values of β (0.15 and 0.4) for a fixed number of landmarks generated using k-means clustering on a synthetic banana-shaped two dimensional set. Note that with the proposed modification, probabilities still represent true similarities according to class labels but only in a neighborhood of the landmarks. Note also that the optimization problem is mathematically equivalent but it will lead to a different solution. In addition, the effective size of the set of constraints (pairs of objects) taken into account has been reduced to a proportion which is $\alpha \cdot \beta$ of the original MCML one while the same reduction when using MCMLA is only α.

It is possible to generalize the problem further by considering the set Y as a variable and then try to learn it. To this end, we first write the corresponding gradient as

$$\nabla_Y J(A, Y, X, N_\beta) = \frac{1}{m} \sum_{\substack{i\,:\,y_i \in Y \\ j\,:\,x_j \in N_\beta(x_i)}} (p_0(j|i) - p^A(j|i)) \cdot \nabla_Y d_{ij}^A$$

with $\nabla_Y d_{ij}^A = 2A(y_i - x_j)$. This expression can be plugged into the same gradient based optimization algorithm along with $\nabla_A J(A, Y, X, N_\beta)$ in order to learn both metric and landmarks at the same time. It is important to note that the problem is no longer convex in general. Nevertheless, with reasonable initializations and in a wide range of experiments it is possible to obtain appropriately good results that approximate the MCML ones as both parameters approach one.

The new algorithm will be referred to as MCMLC(α, β) where the C suffix stands for changing anchors. In the following section, several experiments with different kinds of data are carried out in order to put forward the main benefits of the proposal with regard to previous algorithms.

4 Experiments and Results

Several different publicly available databases have been adopted in order to compare the different methods and extensions described in this work. Firstly, some small size databases from the UCI repository [19] as in previous works have been considered. Moreover, databases involving handwritten digits from the Multiple Features Database [20] and the well-known AR face database [21]

Table 1. Details of the databases used in the experimental validation

Name	Size	Dimension	Classes	Objects/class
Iris	150	4	3	50
Wine	178	13	3	48–71
Balance	625	4	3	49–288
Ionosphere	351	34	2	126–225
Mfeat-kar	1000	20	10	100
AR	532	30	38	14

have also been used. For the purposes of this work, the dimensionality of these two databases has been reduced to 20 and 30, respectively, by using PCA. Table 1 shows the details of the databases. In the particular case of the AR database, only 14 images (the ones without occlusions: scarf, glasses, etc.) per individual (20 men and 20 women) have been taken into account.

All data has been used to learn a metric matrix which has been evaluated by computing the leaving one out error of the nearest neighbor classifier in the corresponding mapped space. Although this is well known to be an optimistic measure of (classification) performance, we have found it well suited to make relative comparisons about the quality of the different metrics and mappings. All the presented results correspond to the average of 5 independent runs.

Landmark points for MCMLA(α) have been selected by running a standard k-means algorithm with $k = \alpha \cdot n$ (the number of desired landmark points given by the proportion α). The initial set of prototypes for MCMLC(α, β) has been computed in exactly the same way using α. Moreover, the size of the subsampled set of neighbors for each prototype has been selected as a proportion, β, of the total size of the available training set.

For the experiments in the present work, typical α values in $\{\frac{1}{8}, \frac{1}{4}, \frac{1}{2}\}$ for the MCMLA have ben set [18]. The proportion of prototypes for MCMLC has been set to smaller values, $\alpha \in \{\frac{1}{20}, \frac{1}{10}, \frac{1}{5}\}$ while the proportion of neighbors per prototype, β, has been set to 5 equally spaced values between $\frac{1}{10}$ and $\frac{1}{2}$. In the particular case of the AR database in which the number of objects per class is only 14, the 3 values of α and 5 values of β have been set as equally spaced between $\frac{1}{5}$ and $\frac{2}{5}$, and $\frac{1}{10}$ and $\frac{1}{2}$, respectively.

The methods considered in this work and corresponding extensions have been implemented using the toolbox drtools [22]. All the other parameters of the different methods have been tuned as in the above toolbox and taking into account appropriate ranges. All databases were centered and normalized (to a fixed and common domain) prior to using the algorithms.

For illustration purposes, some measures on the MCMLC algorithm as it iterates using the small database Wine are shown in Figure 2. These are representative of the behavior of the algorithm in all the databases considered in this work. In particular, the value of the modified criterion, $J_M(A, Y, X, N_\beta)$, along with the corresponding original MCML criterion using all pairs, $J_M(A, X)$, are shown for two different settings for (α, β) on the left hand side of this figure.

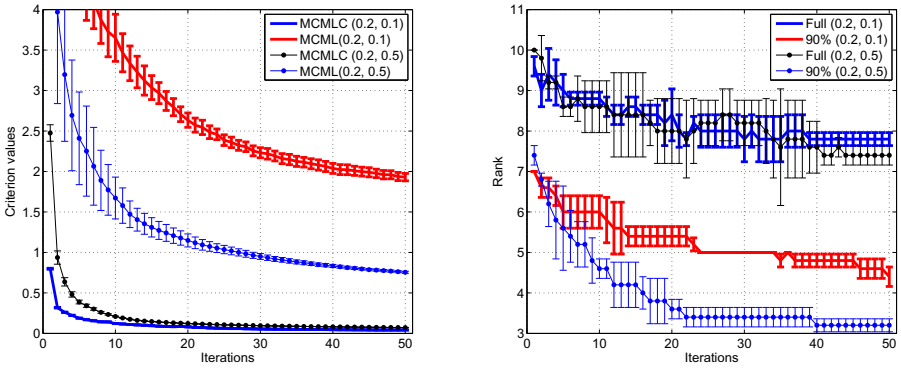

Fig. 2. Criterion (left) and rank (right) values obtained at each iteration when using algorithm MCMLC on the wine dataset. The original MCML criterion is shown together with the MCMLC one for two different (α, β) settings. Strict (or full) rank is shown together with the rank when keeping only the largest eigenvalues up to 90% of the total.

Even noting that absolute values are not directly comparable, we see that the new criteria closely follow the behavior of the original one although it does not necessarily optimize it. As expected, we also see that higher values of β lead to (significantly) smaller values of the original criterion. A subproduct of the new proposal is a reduced variability and consequently the possibility of faster convergence. However, this advantage has not been fully exploited in the present work.

On the right hand side of Figure 2, the values of the ranks of the (PSD version of the) metric matrix with iterations is plotted. The same rank after keeping only the directions that correspond to 90% of the eigenvalues is also shown. As was previously put forward for the MCML algorithm [12, 18], the (full) ranks slowly decrease to arrive at the optimum. Moreover, this decrease does not depend much on the parameter β. On the other hand, a larger sparseness of the metric matrices is observed for greater values of β. If we restrict the ranks in the same way, we observe a significant decrease for higher values of β.

Comparative experiments using MCMLA and MCMLC with the different settings mentioned have also been carried out. The corresponding performance measures on four of the databases are shown in Figure 3. In these plots, the averaged leaving one out error rate estimate corresponding to the nearest neighbor classifier is displayed with regard to the relative number of constraints effectively processed by each algorithm at each iteration (in a logarithmic scale). That is, α for MCMLA(α) and the product $\alpha \cdot \beta$ for MCMLC(α, β). This relative number is an accurate estimate of the computational burden of each algorithm.

The results for the original MCML algorithm are not shown but they are in all cases indistinguishable from the ones obtained for MCMLA($\frac{1}{2}$). Looking at the curves in Figure 3, we see that it is possible to reproduce the behavior of

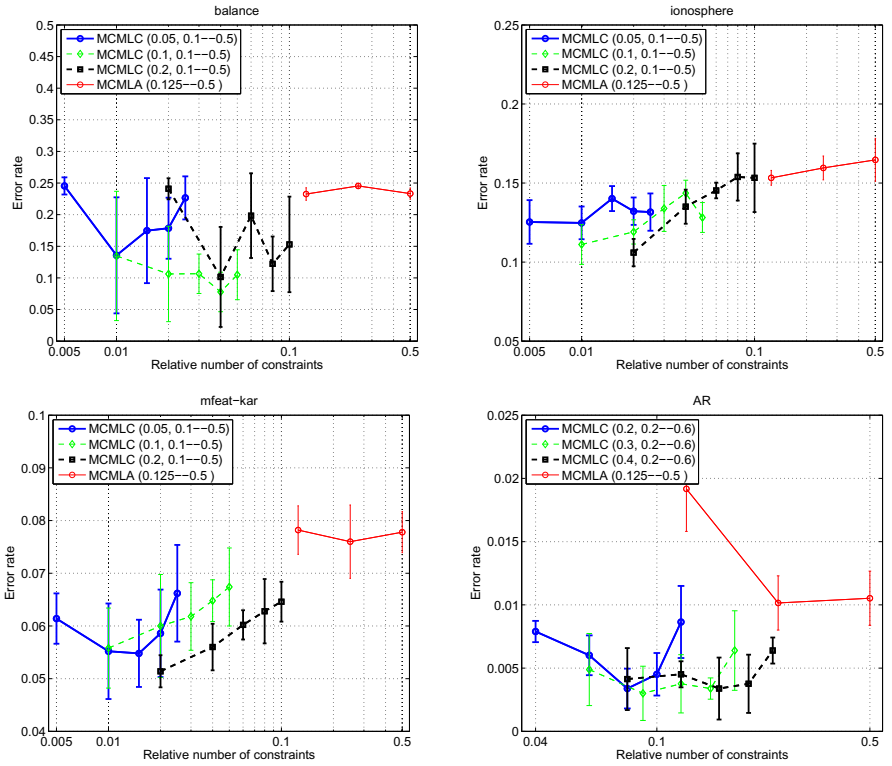

Fig. 3. Performance obtained for the metric learning algorithm with different number of anchors, MCMLA(α), and the proposed generalization with different number of prototypes, MCMLC (α, β). The leaving one out estimate of the 1-NN classifier is shown against the relative number of constraints each method uses (that is, α and $\alpha \cdot \beta$, respectively).

the original MCML algorithm at 50 and 25% of its cost. When using $\alpha = \frac{1}{c}$ (12% in computational cost), the behavior of MCMLA begins to deteriorate in the two larger databases and very significantly in the case of AR. On the other hand, MCMLC with different settings is not only able to reproduce the original behavior but also to improve it. It can be seen that in most of the cases there is a tradeoff between reducing the two parameters and improving the result. In general, we have observed very good results when using from 1% to 5% of the original constraints (around 10% in the case of AR).

5 Concluding Remarks

An empirical evaluation of an extension of a metric learning algorithm that includes prototype generation and adaptation has been considered. The proposed

approach is able to improve both the quality of the metrics obtained and the computational efficiency of the method by significantly reducing the effective number of constraints effectively taken into account at each gradient step.

Some interesting facts and also some critical points have been discovered in this work. Amongst the bad news, the tuning of these methods is not trivial and it is not easy to automate. In fact, more experimentation is needed prior to establishing whether an optimal tradeoff between the two parameters introduced can be found. On the other hand, we have observed that the proposed method leads to good results for a relatively wide range of its parameters.

Apart from consolidating some of the findings of the present work, efforts are also being currently directed towards improving the behavior of the algorithm by forcing and maintaining the sparseness of the metric matrix. A more ambitious line of research tries to formulate some of the ideas in the present work in a more general way. This would make possible to apply them to other metric learning algorithms.

References

1. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley and Sons (2001)
2. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: A review. IEEE Trans. Pattern Anal. Mach. Intell. 22, 4–37 (2000)
3. Fukunaga, K.: Introduction to Statistical Pattern Recognition. Academic Press (1990)
4. Turk, M., Pentland, A.: Eigenfaces for recognition. Journal of Cognitive Neuroscience 3, 71–86 (1991)
5. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Trans. Pattern Anal. Mach. Intell. 19, 711–720 (1997)
6. Chen, L., Liao, H.M., Ko, M., Lin, J., Yu, G.: A new LDA-based face recognition system which can solve the small sample size problem. Pattern Recognition 33, 1713–1726 (2000)
7. Cevikalp, H., Neamtu, M., Wilkes, M., Barkana, A.: Discriminative common vectors for face recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 4–13 (2005)
8. Xing, E.P., Ng, A.Y., Jordan, M.I., Russell, S.J.: Distance metric learning with application to clustering with side-information. In: NIPS, pp. 505–512 (2002)
9. Paredes, R., Vidal, E.: Learning weighted metrics to minimize nearest-neighbor classification error. IEEE Trans. Pattern Anal. Mach. Intell. 28, 1100–1110 (2006)
10. Yu, J., Amores, J., Sebe, N., Radeva, P., Tian, Q.: Distance learning for similarity estimation. IEEE Trans. Pattern Anal. Mach. Intell. 30, 451–462 (2008)
11. Kulis, B.: Metric learning: A survey. Foundations and Trends in Machine Learning 5, 287–364 (2013)
12. Globerson, A., Roweis, S.: Metric learning by collapsing classes. Neural Information Processing Systems (NIPS 2005) 18, 451–458 (2005)
13. Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S.: Information-theoretic metric learning. In: ICML, pp. 209–216 (2007)

14. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research 10, 207–244 (2009)
15. Wang, J., Woznica, A., Kalousis, A.: Learning neighborhoods for metric learning. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part I. LNCS, vol. 7523, pp. 223–236. Springer, Heidelberg (2012)
16. Micó, L., Oncina, J., Vidal, E.: A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. Pattern Recognition Letters 15, 9–17 (1994)
17. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)
18. Perez-Suay, A., Ferri, F.: Scaling up a metric learning algorithm for image recognition and representation. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Remagnino, P., Porikli, F., Peters, J., Klosowski, J., Arns, L., Chun, Y.K., Rhyne, T.-M., Monroe, L. (eds.) ISVC 2008, Part II. LNCS, vol. 5359, pp. 592–601. Springer, Heidelberg (2008)
19. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
20. Duin, R.P.W.: Prtools - version 3.0 - a matlab toolbox for pattern recognition. In: Proc. of SPIE, p. 1331 (2000)
21. Martinez, A., Benavente, R.: The AR face database. Technical Report 24, Computer Vision Center, Barcelona (1998)
22. van der Maaten, L., Postma, E., , van den Herik, H.: Dimensionality reduction: A comparative review. Technical report, Tilburg University, TiCC-TR 2009-005 (2009)