Boosted Metric Learning for Efficient Identity-Based Face Retrieval

Romain Negrel
romain.negrel@unicaen.fr
Alexis Lechervy
alexis.lechervy@unicaen.fr
Frederic Jurie
frederic.jurie@unicaen.fr

GREYC, CNRS UMR 6072, ENSICAEN
Université de Caen Basse-Normandie
France

Abstract

This paper presents MLBoost, an efficient method for learning to compare face signatures, and shows its application to the hierarchical organization of large face databases. More precisely, the proposed metric learning (ML) algorithm is based on boosting so that the metric is learned iteratively by combining several weak metrics. Boosting allows our method to be free of any hyper-parameters (no cross-validation required) and to be robust with respect to overfitting. This MLBoost algorithm can be trained from constraints involving two pairs of vectors (quadruplets) with a quadratic complexity. The paper also shows how it can be included in a semi-supervised hierarchical clustering framework adapted to identity based face search. Our approach is validated on a benchmark relying on the Labelled Faces in the Wild (LFW) dataset supplemented with 1M face distractors.

1 Introduction and related Work

The performance of many visual recognition algorithms – *e.g.* face verification or person re-identification – heavily relies on the metric used to measure the similarity between input data. Recent works such as [5, 10, 12, 25, 31] have shown the interest of learning an optimal metric for a particular task using Metric Learning (ML). Most approaches learn a Mahalanobis metric based on an objective function whose constraints come either from a set of labeled examples or, more frequently, from sets of positive (in the same class) and negative (in different classes) pairs.

Presenting the exhaustive list of recent works that have used metric learning (ML) in such a context would be rather tedious, given their large number. However, we can mention a few of the most notable approaches such as DDML [14], RBML [18], Structural ML [32], PCCA [19], rPCCA [31], LMNN [30], LDML [13], ITML [7], KISSME [17], RS-KISSME [26], SML [9]. The reader interested in ML in general is referred to the recent survey by Bellet *et al.* [1]. The difference among these methods mainly lies in their objective functions which is task dependent or in the type of constraints of the objective function, (*e.g.* some require pairwise distances between points making large scale applications difficult).

The works of Shen *et al.* [24] and Bi *et al.* [3], which have introduced Boosting based approaches, deserve a particular attention due to the interesting properties they offer: i) they

are efficient and scalable, as no semidefinite programming is required and only the largest eigenvalue and corresponding eigenvectors are needed at each iteration; ii) like AdaBoost, they don't have any parameters to tune and they are easy to implement as no sophisticated optimization techniques are involved. These approaches hence contrast with most of the commonly used ML methods *e.g.* [4, 12, 19, 25, 31] for which hyper-parameters, often introduced for regularization purposes, have to be adjusted by cross-validation.

However, one strong limitation of these two approaches [3, 24] is that they require having triplets for learning the metric, *i.e.*, constraints expressed under the form $\mathrm{D}(x_a,x_b) < \mathrm{D}(x_a,x_c)$, where x_a, x_b, x_c are three input vectors for which the label is known: positive and negative pairs within a constraint have to share a common vector. This is a limitation as for most of the verification task only training pairs are available (*e.g.*, for person re-identification on the Viper dataset [11] only pairs of same/different persons are provided for training and, thus, using such triplets is not possible).

One of the key contributions of this paper is to propose a metric learning approach based on boosting allowing the use of use of pairs of points for training, *i.e.* which can use constraints such as $\mathrm{D}(x_a,x_b) < \mathrm{D}(x_c,x_d)$ while keeping the good properties of the approach in [24] (scalability, simplicity, no parameters, *etc.*). Our approach is based on the ranking algorithm RankBoost [9], known to offer three particularly interesting features: i) it requires no hyperparameters, ii) it has great robustness to overfitting (explained with a standard VC-dimensional analysis techniques [8, 27]), and iii) it uses a computational trick for reducing the complexity of the learning step. In the following, we show how to build on RankBoost [9] to efficiently address this metric learning problem.

The proposed approach is validated on the *identity face retrieval task*, such as defined by Bhattarai *et al.* [2]. This task consists in retrieving the face(s) of the same person to a query, from a large database of faces. The task is evaluated both in terms of scalability and accuracy. The idea presented in [2] bears similarity with the work of Verma *et al.* [29], who proposed a method for learning hierarchical similarity metrics using a taxonomy. However, unlike Verma *et al.*, the method proposed by Bhattarai *et al.* does not require any preexisting taxonomy. The experimented validation in [29] shows that using the proposed ML algorithm, allows to improve the performance by a large margin within the same framework as [2].

2 Proposed method : from RankBoost to boosted ML

2.1 Metric learning with boosting

Our approach, referred to as MLBoost, lies in the family of Mahalanobis ML. This means that the metric $\mathrm{D}_{\mathbf{W}}(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i-\mathbf{x}_j)^\top \mathbf{W}(\mathbf{x}_i-\mathbf{x}_j)$ is parametrized by the positive semi-definite (PSD) matrix $\mathbf{W}$ and $(\mathbf{x}_i,\mathbf{x}_j) \in \mathbb{R}^D \times \mathbb{R}^D$. This metric has two very interesting properties: (i) it is convex in $\mathbf{W}$ allowing to build convex objective functions; (ii) it can be used for the dimensionality reduction of the input data. Regarding this last point, we use the following decomposition $\mathbf{W} = \mathbf{L}\mathbf{L}^\top$, with $\mathbf{L} \in \mathbb{R}^{D\times d}$ and $d << D$ the dimension of reduced space. The data can be projected as: $\mathbf{y}_i = \mathbf{L}^\top \mathbf{x}_i$ and can be compared with the Euclidean distance: $\mathrm{D}_{\mathbf{L}\mathbf{L}^\top}(\mathbf{x}_i,\mathbf{x}_j) = (\mathbf{x}_i-\mathbf{x}_j)^\top \mathbf{L}\mathbf{L}^\top(\mathbf{x}_i-\mathbf{x}_j) = \|\mathbf{y}_i-\mathbf{y}_j\|_2^2$.

In the same way boosting builds *strong* (good) solutions to a problem by the combination of *weak* solutions [23], ML with boosting (*e.g.* [24]) consists in iteratively building a linear combination of weak Mahalanobis distances of rank 1, denoted by $\mathrm{D}_{\mathbf{Z}}^{(t)}$ (with $\mathrm{rank}(\mathbf{Z}^{(t)}) = 1$ and $\mathrm{tr}(\mathbf{Z}^{(t)}) = 1$): $\mathrm{D}_{\mathbf{W}}(\mathbf{x},\mathbf{y}) = \sum_{t=1}^{T} \alpha^{(t)} \mathrm{D}_{\mathbf{Z}^{(t)}}(\mathbf{x},\mathbf{y})$ in the following.

2.2 Metric Leaning with Boosting

We assume that the constraints consist of positive and negative pairs such as $\mathrm{D}_{\mathbf{W}}(\mathbf{p}_{1i},\mathbf{p}_{2i}) < \mathrm{D}_{\mathbf{W}}(\mathbf{n}_{1j},\mathbf{n}_{2j})$, with $(\mathbf{p}_{1i},\mathbf{p}_{2i}) \in \mathcal{P}$ the set of similar pairs and $(\mathbf{n}_{1j},\mathbf{n}_{2j}) \in \mathcal{N}$ the set of dissimilar pairs. This ML problem is closely related the following ranking problem : $\mathrm{H}(p_i) < \mathrm{H}(n_j), \forall (p_i,n_i)$, where $\mathrm{H}(\cdot)$ denotes a ranking function and $p_i = (\mathbf{p}_{1i},\mathbf{p}_{2i}) \in \mathcal{P}$ and $n_i = (\mathbf{n}_{1i},\mathbf{n}_{2i}) \in \mathcal{N}$. We address this problem by building on the RankBoost algorithm [9].

Let us start by associating a weight to each constraint: $\mathcal{D}^{(t)} = \left\{ d_{ij}^{(t)} \in (0,1) \;\middle|\; c_{ij} \in \mathcal{C} \right\}$, with $\mathcal{C} = \left\{ \left((\mathbf{p}_{1i},\mathbf{p}_{2i}),(\mathbf{n}_{1j},\mathbf{n}_{2j})\right) \in \mathcal{P}\times\mathcal{N} \right\}$ the set of all quadruplets composed by a positive and a negative pair. These weights change over the iterations (denoted as t) of the algorithm. They can be interpreted as a measure of the importance of the correct ranking the pair (i,j) at the current iteration t. During the iterations, these weights are normalized such that:

$$\sum_{d_{ij}^{(t)} \in \mathcal{D}^{(t)}} d_{ij}^{(t)} = 1. \tag{1}$$

At the initialization of the algorithm, all quadruples have the same importance and the weights are uniform : $d_{ij}^{(1)} = \frac{1}{|\mathcal{P}||\mathcal{N}|}, \forall d_{ij} \in \mathcal{D}$. The loss function of MLBoost can be written as:

$$\mathrm{loss}(\mathbf{W}) = \sum_{\mathcal{C}} d_{ij}^{(1)} \left[\!\left[\mathrm{D}_{\mathbf{W}}(\mathbf{p}_{1i},\mathbf{p}_{2i}) \geq \mathrm{D}_{\mathbf{W}}(\mathbf{n}_{1j},\mathbf{n}_{2j}) \right]\!\right], \tag{2}$$

with the function $[\![\pi]\!]$ that returns 1 if predicate π holds and 0 otherwise. At each iteration, the weights are updated according to:

$$d_{ij}^{(t+1)} = \frac{d_{ij}^{(t)} \, \mathrm{e}^{\alpha^{(t)}\left(\mathrm{D}_{\mathbf{Z}^{(t)}}(\mathbf{p}_{1i},\mathbf{p}_{2i}) - \mathrm{D}_{\mathbf{Z}^{(t)}}(\mathbf{n}_{1j},\mathbf{n}_{2j})\right)}}{w^{(t)}}, \tag{3}$$

with $\mathrm{D}_{\mathbf{Z}^{(t)}}$ the weak metric computed at iteration t, $\alpha^{(t)}$ its weight and $w^{(t)}$ the normalization factor chosen so that the constraint (1) is met. Equation (3) and the inequality $[\![x \geq 0]\!] \leq \mathrm{e}^{x}$, allow us to show that:

$$\mathrm{loss}(\mathbf{W}) \leq \prod_{t=1}^{T} w^{(t)}. \tag{4}$$

The objective at each iteration is then to compute the weak metric $\mathrm{D}_{\mathbf{Z}^{(t)}}$ and the weight $\alpha^{(t)}$ which minimizes the following function:

$$w^{(t)} = \sum_{\mathcal{C}} d_{ij}^{(t)} \, \mathrm{e}^{\alpha^{(t)}\left(\mathrm{D}_{\mathbf{Z}^{(t)}}(\mathbf{p}_{1i},\mathbf{p}_{2i}) - \mathrm{D}_{\mathbf{Z}^{(t)}}(\mathbf{n}_{1j},\mathbf{n}_{2j})\right)}, \tag{5}$$

with $\mathrm{rank}(\mathbf{Z}^{(t)}) = 1$, $\mathrm{tr}(\mathbf{Z}^{(t)}) = 1$ and $\alpha^{(t)} > 0$. We note that for increasing the performance of the final metric, we only need to have $w^{(t)} < 1$.

Weak learning algorithm: The weak metric $\mathrm{D}_{\mathbf{Z}^{(t)}}$ is obtained by computing the solution of an approximation of Equation (5). Freund *et al.* [9] propose to approximate the exponential by the following lower bound:

$$\mathrm{e}^{\alpha x} \leq \left(\frac{1+x}{2}\right)\mathrm{e}^{\alpha} + \left(\frac{1-x}{2}\right)\mathrm{e}^{-\alpha}, \tag{6}$$

for $x \in [-1,1]$ and $\alpha \in \mathbb{R}$. Using, this idea, obtain the following optimization problem :

$$\arg\min_{\alpha,\mathbf{Z}} \left(\left(\frac{1-\mathrm{r}(\mathbf{Z})}{2} \right) \mathrm{e}^{\alpha} + \left(\frac{1+\mathrm{r}(\mathbf{Z})}{2} \right) \mathrm{e}^{-\alpha} \right), \tag{7}$$

where

$$\mathrm{r}(\mathbf{Z}) = \sum_{\mathcal{C}} d_{ij}^{(t)} \left(\mathrm{D}_{\mathbf{Z}} \left(\mathbf{n}_{1j}, \mathbf{n}_{2j} \right) - \mathrm{D}_{\mathbf{Z}} \left(\mathbf{p}_{1i}, \mathbf{p}_{2i} \right) \right). \tag{8}$$

The optimization problem (7) is minimized when $\alpha = \frac{1}{2} \ln \left(\frac{1+\mathrm{r}(\mathbf{Z})}{1-\mathrm{r}(\mathbf{Z})} \right)$, showing that:

$$\prod_{t'=1}^{t} w^{(t')} \leq \sqrt{1-\mathrm{r}(\mathbf{Z})^2}, \tag{9}$$

Shen *et al.* [24] proposed to use eq. (8) to compute the weak metric by solving the following optimization problem:

$$\begin{aligned} \mathbf{Z}^{(t)} = & \arg\max_{\mathbf{Z}} \ \mathrm{r}(\mathbf{Z}) \\ & \text{s.t. } \mathrm{rank}(\mathbf{Z}) = 1, \ \mathrm{tr}(\mathbf{Z}) = 1 \end{aligned} \tag{10}$$

By making the following variable change: $\mathbf{Z} = \mathbf{z}\mathbf{z}^{\top}$, the previous problem is equivalent to:

$$\begin{aligned} \mathbf{z}^{(t)} = & \arg\max_{\mathbf{z}} \mathbf{z}^{\top} \mathbf{A}^{(t)} \mathbf{z} \\ & \text{s.t. } \|\mathbf{z}\|_2 = 1 \end{aligned} \tag{11}$$

with $\mathbf{A}^{(t)} = \sum_{\mathcal{C}} d_{ij}^{(t)} \left((\mathbf{n}_{1j} - \mathbf{n}_{2j})(\mathbf{n}_{1j} - \mathbf{n}_{2j})^{\top} - (\mathbf{p}_{1i} - \mathbf{p}_{2i})(\mathbf{p}_{1i} - \mathbf{p}_{2i})^{\top} \right)$. The optimization problem (11) is simply solved by the computation of the eigenvector corresponding to the largest eigenvalue of the $\mathbf{A}^{(t)}$ matrix.

Choosing $\alpha^{(t)}$: once the weak metric $\mathrm{D}_{\mathbf{Z}}^{(t)}$ is computed with the aforementioned method, the best weight $\alpha^{(t)}$ of this weak metric is computed by solving the following problem:

$$\alpha^{(t)} = \arg\min_{\alpha} \sum_{\mathcal{C}} d_{ij}^{(t)} \, \mathrm{e}^{\alpha \left(\mathrm{D}_{\mathbf{Z}^{(t)}} (\mathbf{p}_{1i}, \mathbf{p}_{2i}) - \mathrm{D}_{\mathbf{Z}^{(t)}} \left(\mathbf{n}_{1j}, \mathbf{n}_{2j} \right) \right)}. \tag{12}$$

This optimization problem is strictly convex and therefore has a unique minimum. We find the solution numerically via the golden section search method. We note that if $\alpha^{(t)}$ is negative or zero, we stop the algorithm. Indeed, we want the parameter matrix of the final metric to meet the positive-semidefinite constraint.

2.3 Reduction of the algorithmic complexity

A naive implementation of the presented MLBoost algorithm would induce a high computational complexity. Indeed, the complexity depends on the cardinality of $\mathcal{C}$, the set quadruplets. Under the assumption that the images are uniformly distributed across identities, we can approximate the cardinality of $\mathcal{C}$ by:

$$|\mathcal{C}| = |\mathcal{P}||\mathcal{N}| \approx \frac{|\mathcal{I}|^4}{2c}, \tag{13}$$

Algorithm 1 MLBoost with efficient learning implementation

procedure MLBOOST($\mathbf{X}, \mathcal{P}, \mathcal{N}, itersMax$)
  $t \leftarrow 1$
  $\mathbf{L}^{(t)} \leftarrow \varnothing$
  Initialize weights: $u_i^{(t)} = 1/|\mathcal{P}|,\ \forall i \in 1 \dots |\mathcal{P}|,\ v_j^{(t)} = 1/|\mathcal{N}|,\ \forall j \in 1 \dots |\mathcal{N}|$.
  repeat
    Compute weak metric: $\mathbf{z}^{(t)}$ by solving of eq. (11) + (16).
    Choose the best: $\alpha^{(t)}$ by solving of eq. (17).
    if $\alpha^{(t)} \leq 0$ **then**
      break
    $\mathbf{L}^{(t+1)} \leftarrow \left[\mathbf{L}^{(t)}, \sqrt{\alpha^{(t)}}\mathbf{z}^{(t)}\right]$
    Update weights : $u_i^{(t+1)}$ and $v_j^{(t)}$ with the eq. (15).
  until $t < itersMax$
  return $\mathbf{L}$

with $|\mathcal{I}|$ the total number of images and c the number of identities. The computational complexity, as a function of the number of images, is thus $\mathcal{O}(|\mathcal{I}|^4)$.

Interestingly, Freund *et al*. [9] have introduced a way to rewrite the equations of RankBoost in order to reduce the computational complexity. It turns out these principles extend to our MLBoost algorithm. For doing this, we decompose $d_{ij}^{(t)}$, the weights of quadruplets in $\mathcal{C}$, as follows:

$$d_{ij}^{(t)} = u_i^{(t)} v_i^{(t)}, \tag{14}$$

with $u_i^{(1)} = 1/|\mathcal{P}|,\ \forall i$ and $v_j^{(1)} = 1/|\mathcal{N}|,\ \forall j$. Equation (3) used to update the weights can be rewritten as :

$$u_i^{(t+1)} = \frac{u_i^{(t)} \, \mathrm{e}^{\alpha^{(t)} \mathrm{D}_{\mathbf{z}^{(t)}}(\mathbf{p}_{1i}, \mathbf{p}_{2i})}}{w_{\mathcal{P}}^{(t)}},\ \forall i \qquad v_j^{(t+1)} = \frac{v_j^{(t)} \, \mathrm{e}^{-\alpha^{(t)} \mathrm{D}_{\mathbf{z}^{(t)}}(\mathbf{n}_{1j}, \mathbf{n}_{2j})}}{w_{\mathcal{N}}^{(t)}},\ \forall j \tag{15}$$

with $w_{\mathcal{P}}^{(t)}$ and $w_{\mathcal{N}}^{(t)}$ the normalization factors chosen such that $\sum u_i^{(t+1)} = 1$ and $\sum v_i^{(t+1)} = 1$. The expression of matrix $\mathbf{A}^{(t)}$ used to compute the weak metric (11) is rewritten as follows:

$$\mathbf{A}^{(t)} = \sum_{\mathcal{N}} v_j^{(t)} \left((\mathbf{n}_{1j} - \mathbf{n}_{2j})(\mathbf{n}_{1j} - \mathbf{n}_{2j})^\top \right) - \sum_{\mathcal{P}} u_i^{(t)} \left((\mathbf{p}_{1i} - \mathbf{p}_{2i})(\mathbf{p}_{1i} - \mathbf{p}_{2i})^\top \right). \tag{16}$$

The optimization problem (12) allowing to compute $\alpha^{(t)}$, which are the weights of this weak metric, is rewritten as:

$$\alpha^{(t)} = \arg\min_{\alpha} \left(\sum_{\mathcal{P}} u_i^{(t)} \, \mathrm{e}^{\alpha \left(\mathrm{D}_{\mathbf{z}^{(t)}}(\mathbf{p}_{1i}, \mathbf{p}_{2i}) \right)} \right) \left(\sum_{\mathcal{N}} v_j^{(t)} \, \mathrm{e}^{-\alpha \left(\mathrm{D}_{\mathbf{z}^{(t)}}(\mathbf{n}_{1j}, \mathbf{n}_{2j}) \right)} \right). \tag{17}$$

In this form, the equations have a much lower computational complexity which it depends on the sum of the number of positive and negative pairs : $|\mathcal{P}| + |\mathcal{N}|$. The computational complexity is then of the order of $\mathcal{O}(|\mathcal{I}|^2)$. The different steps of MLBoost are described in Algorithm 1.

Algorithm 2

procedure HIERARCHICAL_TREE_CLUSTERING($\mathbf{X}, \mathbf{y}, nbPairs, nbCluster, treeDepth$)
  $rootNode$.**idxC** $\leftarrow [1 \dots N]$ and $rootNode$.**idxML** $\leftarrow [1 \dots N]$
  $WorkQueue$.push($rootNode$)
  while $WorkQueue$.empty() == false **do**
    $node \leftarrow WorkQueue$.pop()
    $(\mathcal{P}, \mathcal{N}) \leftarrow$ build_pairs($\mathbf{y}[node.idxML]$, $nbPairs$)
    $\mathbf{L} \leftarrow RankBoost(\mathbf{X}[:, node.idxML], \mathcal{P}, \mathcal{N}, 1024)$
    $node.\mathbf{L} \leftarrow \mathbf{L}$
    if $node.depth < treeDepth$ **then**
      $\mathbf{X}' \leftarrow \mathbf{L}^\top \mathbf{X}[:, node.\mathbf{idxC}]$
      $\mathbf{C} \leftarrow clustering(\mathbf{X}', nbCluster)$
      for i = 1 ... nbCluster **do**
        $childNode.\mathbf{idxC} \leftarrow$ cluster_assign($node.\mathbf{idxC}$, $\mathbf{X}'$, $\mathbf{C}$, i)
        $childNode.\mathbf{idxML} \leftarrow$ ml_selection($\mathbf{y}$, $childNode.\mathbf{idxC}$)
        $node$.addChild($childNode$)
        $WorkQueue$.push($childNode$)
  return $rootNode$

3 MLBoost for Semi-Supervised Hierarchical Clustering

In [2], Bhattarai *et al*. proposed a method for building hierarchical face representations for efficient identity-based face retrieval. The main idea is to use a ML algorithm for building a tree structured representation of a large set of faces, such that identical-identity faces belongs to the same cluster at each tree level. The hierarchical clustering is built recursively: at each level, a new metric is learned using the faces belonging to a particular tree node which is then used to locally cluster the faces and to create the sub-branches.

The retrieval for a new query face is done by visiting the tree from top to bottom until a leaf is reached; if the current node is a non-leaf node, the face is projected into the corresponding subspace and compared with its centroids, which allows to move to the closest child node. When a leaf is reached, the face is compared with all the faces contained in the leaf using the corresponding local metric.

The main drawback of this method is that, when a local metric is learned within a particular node of the tree, no information is retained from the parent nodes. By doing this, the metrics become quickly too local (over specialized). Furthermore, misclassified faces (those going in to the wrong branches) can strongly impact the local metrics.

We therefore propose to improve [2] by combining the metrics learnt at different levels of the tree. A simple method for doing this would be to add an elastic term between the metric from the parent branch and the metric learned. However, the addition of these terms would add hyper-parameters to the metric learning problem and would require some additional tuning. We propose instead a simple method based on the selection of training samples: if one or more examples of an identity belongs to a given node, we use all the examples of this identity to learn the metric as described in Algo. 2. We see that, in each branch, we have two vectors of example indices: the one of the faces actually belonging to the branch (denoted as **idxC** in Algo. 2 this vector is the same as that used in the algorithm proposed by [2]); the example indices used to learn the metric in the branch, denoted as **idxML** in Algo. 2.

4 Experiments

Performance evaluation. In the considered scenario: a person has to be identified given an image of his face by querying a dataset containing faces of different identities. The performance should reflect the ability of the system to place the correct identity at the beginning of a list of candidate identities. The criterion used to evaluate the performance is the one used in [2], *i.e.*, the mean of k-call@n (such as defined in [6]), with $k = 1$. We measure the performance for several values of n, *i.e.*, $n \in \{1, 10, 20, 50, 100\}$.

Database: We use the well known Labeled Faces in the Wild (LFW) database by Huang *et al.* [16]. This database has more than 13000 images of over 4000 different identity. We use its aligned version with deep funneling [15]. In our experiments, we use the same set of images and queries than the ones used by Bhattarai *et al.* [2]. We keep only the images of identities with at least five examples each, all other images are not used during the experiments; it results in a set of 5985 images of 423 persons. Regarding the query set, we select one query image per identity (the same as Bhattarai *et al.*). The training set contains the selected images, excluding the query set. To evaluate the robustness with respect to large-scale databases, we use the set of million distractor faces kindly provided by Bhattarai *et al.* [2].

Image description: We evaluate the methods with two types of image signatures: (i) Local Binary Pattern (LBP) [20] and (ii) Fisher Vector (FV) [21, 22]. To extract the LBP signature, we use the same protocol as proposed by Bhattarai *et al.*, each image being described by a vector of 9860 dimensions [2, 28]. To extract the FV signature, we use the implementation proposed by Simonyan *et al.* [25]. We use the publicly available source code[1] with the default parameters and with our own training set. Each image is then described by a vector of 67584 dimensions. The two types of signatures (LPB and FV) are subsequently ℓ_2-normalized.

Traing pairs: We select as many positive pairs as negative ones, uniformly distributed across the identities.

Dimension of the output space: The proposed algorithm does not allow to control explicitly the dimension of the output space, which is required for fair comparisons (*i.e.*, for comparing two approaches for a given dimensionality of the projected features). Thus, we select the d best projectors by applying a singular value decomposition of the projectors resulting in a d-dimensional signature.

Baseline (PCA, no metric learning): These experiments are intended to evaluate the performance obtained by the raw signatures (without metric learning), possibly projected in a low dimensional space by PCA. They are reported in Table 1, which compares the performance of the two signatures (LBP and FV), without distractors or with 100.000 and 1M distractors, and for different values of recall n. The performances are given in the mean 1-call@n (%). Without distractors, the FV signatures perform better than LBP, while with distractors LBP outperforms FV. Projecting the signatures on a subspace of 4096 dimensions or more does not alter the performance significantly. In the following experiments, we prereduce the signatures at 4096 components with the PCA projectors.

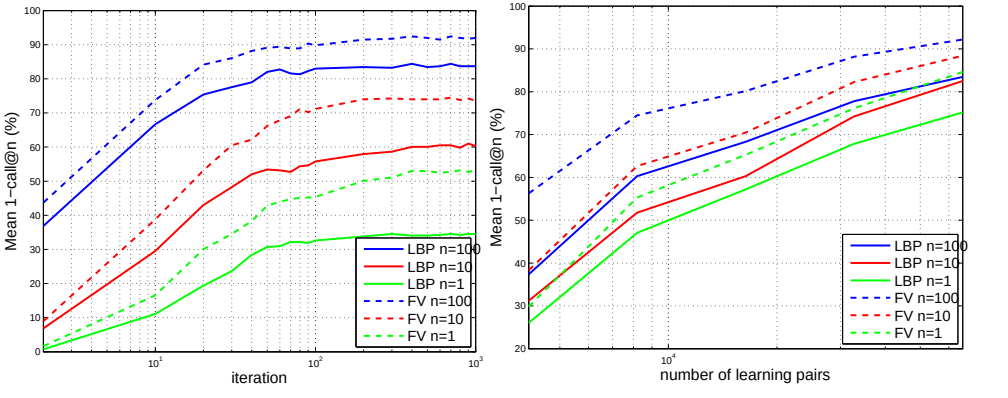
Experimental validation of the proposed algorithm (exhaustive search)

Figure 1(a) illustrates some results on convergence of the algorithm (without distractors, using 2×32768 pairs for training). Interestingly, we do not see any overfitting (this is confirmed by adding even more iterations). Figure 1(b) shows the performance of the algorithm

[1] http://www.robots.ox.ac.uk/~vgg/software/face_desc/

Sign.	Dim.	No distractor					+100.000 distractors					+1.000.000 distractors				
		n= 1	n= 10	n= 20	n= 50	n= 100	n= 1	n= 10	n= 20	n= 50	n= 100	n= 1	n= 10	n= 20	n= 50	n= 100
LBP	9860	31.9	53.7	60.5	68.8	74.7	31.4	52.7	59.3	67.8	73.3	31	50.6	57.2	65.0	70.4
LBP-PCA	8192	31.9	53.7	60.5	68.8	74.7	31.4	52.2	58.4	66.7	72.1	30.3	49.4	55.8	61.9	69
LBP-PCA	4096	31.7	52.7	60.3	68.1	74.2	30.5	49.6	56.5	62.6	69.3	28.1	44.7	48.9	57.2	62.4
LBP-PCA	32	16.1	33.3	43.3	55.8	66.7	15.1	31.4	39.5	51.3	60.3	13.2	29.1	33.3	42.3	50.1
FV	67584	57.9	77.5	83.5	86.6	92.8	53.2	69.3	73	78	72.7	-	-	-	-	-
FV-PCA	8192	55.8	75.2	81.3	88.4	92.9	12.5	17.3	18	19.6	20.8	11.1	13.7	14.4	16.3	17.0
FV-PCA	4096	56.5	75.2	81.8	90.1	92	17.49	21.5	22.9	26.5	30	13.9	18.4	19.6	20.3	21
FV-PCA	32	21.7	42.1	49.4	62.9	72.8	16.5	25.5	29.8	35.7	40	10.4	18.2	20.6	24.8	28.1

Table 1: Mean 1-call@n performance given by raw LBP and FV signatures (no ML), with/without PCA, with/without distractors.



(a) Performance vs number of boosting rounds

(b) Performance vs number of training pairs

Figure 1: Performance of the proposed ML approach

as a function of the number of pairs used to learn the metric (no distractors, 1000 rounds of boosting). We observe that the evolution of performance is sub-logarithmic in the number of pairs.

Table 2 compares the performance of the proposed MLBoost algorithm to the performance of the state-of-the art PCCA (Pairwise Constrained Component Analysis) [2, 19] with and without distractors for different dimensions of the final signature. For fair comparison, we use the signatures kindly given by the authors of [2]. We apply 1000 rounds of boosting. The performance of the proposed approach improves over PCCA by about 20% (LBP, $n = 10$). We observe, in our experiments, that PCCA tends to overfit and that the performance could certainly be improved by adding a regularization term, such as proposed by Xiong *et al.* [31]. However, adding a regularization term would add a second hyper-parameter to PCCA, whereas our method has no hyper-parameter at all.

Sign.	Dim.	No distractor					+100.000 distractors					+1.000.000 distractors				
		n= 1	n= 10	n= 20	n= 50	n= 100	n= 1	n= 10	n= 20	n= 50	n= 100	n= 1	n= 10	n= 20	n= 50	n= 100
FV-MLBoost	32	44.2	70	78.7	85.8	91	28.4	47.3	52.7	60	63.6	20.3	32.6	38.1	44	50.1
LBP-MLBoost	32	31	54.1	63.4	74	83.2	26.2	43.7	49.2	57.7	64.1	22.9	34.3	39.5	46.1	51.8
LBP-PCCA [2]	32	-	32.1	41.3	53.4	63.5	-	23.6	28.3	41.1	48.9	-	13.4	18.2	26.7	33.8

Table 2: Comparison with PCCA [2].

Sign.	Method	Dim.	No distractor			+100.000 distractors			+1.000.000 distractors		
			n= 1	n= 10	n= 20	n= 1	n= 10	n= 20	n= 1	n= 10	n= 20
LBP	MLBoost	32	30.7	53.7	62.2	26.2	43.0	48.7	21.5	34.5	38.5
LBP	MLBoost-Tree-d3-Global	32	27.9	50.4	58.6	24.1	39.5	46.1	20.1	31.4	36.2
LBP	MLBoost-Tree-d3	32	29.6	54.6	62.2	23.6	39.5	45.2	17.7	31.0	35.2
LBP	PCCA-Tree-d3 [2]	32	-	33.3	40.9	-	27.4	32.2	-	18.5	23.9
LBP	MLBoost-Tree-d4-Global	32	27.0	49.2	56.7	22.7	37.4	42.3	18.9	30.5	33.6
LBP	MLBoost-Tree-d4	32	28.6	52.0	59.3	21.0	35.5	40.9	17.3	26.7	30.5
LBP	PCCA-Tree-d4 [2]	32	-	36.2	41.6	-	32.4	36.6	-	30.3	31.7
FV	MLBoost	32	44.7	69.0	78.0	28.6	48.0	52.5	20.1	34.0	38.3
FV	MLBoost-Tree-d3-Global	32	42.6	65.7	75.7	27.7	45.9	48.9	19.1	33.3	37.1
FV	MLBoost-Tree-d3	32	48.0	69.0	75.2	30.3	42.8	45.4	22.0	33.1	36.4
FV	MLBoost-Tree-d4-Global	32	43.0	66.0	75.2	27.6	45.4	48.9	19.1	33.3	36.9
FV	MLBoost-Tree-d4	32	51.1	70.2	73.8	27.0	39.2	42.3	22.5	30.3	31.7

Table 3: Performance of the hierarchical tree structure and comparison with PCCA [2].

Hierarchical tree structure for efficient large-scale identity-based face retrieval

This last part of this experimental section focuses on the use of our MLBoost algorithm for face retrieval, in the context of semi-supervised hierarchical clustering, with/without distractors. Here again, we use LBP and FV signatures and set the number of training pairs to 2×16384 and the dimension of the output space to 32. We restrict ourselves to a binary clustering tree of 3 and 4 levels (denoted with suffix "-d3" and "-d4"). Table 3 reports the 1-recall@n performance for the two descriptors, for different depths of the tree and for different amounts of distractors.

Searching the faces in only one leaf of the tree affects the overall performance: misclassifications that can happen at any level of the tree cannot be recovered later. This loss increases with the depth of the tree. We can observe this by comparing the performance of the global metric used without and with the tree structure ("MLBoost" in Table 3) (methods whose names include "MLBoost-Tree-d*-Global"). The loss is of about 1-2% at $n = 1$ with FV.

We can also observe that the local metric (the lines denoted as "MLBoost-Tree-d*") improves the performance of the tree-based structure for the small values of n, not only compensating entirely the loss induced by the tree but also giving a significant improvement of more than 6% ($n = 1$, without distractors, with FV). Comparing our method with the state-of-the-art method of [2] (lines denoted as "PCCA-Tree-d*") shows that our methods can perform up to 30% better for some configurations. We can also see that the FV signature provides significant gain compared to the LBP signature.

5 Conclusions

This paper has introduced a boosted ML method called MLBoost that, in contrast with most of ML methods, does not require any hyper-parameters to be tuned and is not subject to overfitting. The paper also shows how to use the proposed MLBoost algorithm for computing hierarchical organizations of face databases allowing to reduce the computational cost of identity-based face retrieval. The proposed experimental validation has demonstrated the robustness of the method w.r.t. overfitting. In addition, its performance is significantly better than PCCA, which is one of the state-of-the-art ML method. Finally, local ML has been shown to provide better performance than global ML in this context.

Acknowledgments

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-12-SECU-0005 (project PHYSIONOMIE).

References

[1] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[2] Binod Bhattarai, Gaurav Sharma, Frederic Jurie, and Patrick Pérez. Some faces are more equal than others: Hierarchical organization for accurate and efficient large-scale identity-based face retrieval. In *European Conference on Computer Vision (ECCV) Workshops*, pages 1–13, 2014.

[3] Jinbo Bi, Dijia Wu, Le Lu, Meizhu Liu, Yimo Tao, and Matthias Wolf. AdaBoost on low-rank psd matrices for metric learning. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2617–2624. IEEE, 2011.

[4] Qiong Cao, Yiming Ying, and Peng Li. Similarity metric learning for face recognition. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2408–2415. IEEE, 2013.

[5] Dong Chen, Xudong Cao, Fang Wen, and Jian Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3025–3032. IEEE, 2013.

[6] Harr Chen and David R. Karger. Less is more: Probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 429–436, New York, NY, USA, 2006. ACM.

[7] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 209–216, New York, NY, USA, 2007. ACM.

[8] Luc Devroye. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 1996.

[9] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4: 933–969, 2003.

[10] Andrea Frome, Yoram Singer, Fei Sha, and Jitendra Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[11] Douglas Gray, Shane Brennan, and Hai Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *IEEE International workshop on performance evaluation of tracking and surveillance*. Citeseer, 2007.

[12] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Is that you? metric learning approaches for face identification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 498–505. IEEE, 2009.

[13] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. Face recognition from caption-based supervision. *International Journal of Computer Vision*, 96(1):64–82, 2012.

[14] Junlin Hu, Jiwen Lu, and Yap P. P. Tan. Discriminative deep metric learning for face verification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1882, 2014.

[15] Gary Huang, Marwan Mattar, Honglak Lee, and Erik G. Learned-Miller. Learning to align from scratch. In *Advances in Neural Information Processing Systems*, pages 764–772, 2012.

[16] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[17] Martin Köstinger, Martin Hirzer, Paul Wohlhart, Peter M. Roth, and Horst Bischof. Large scale metric learning from equivalence constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2288–2295. IEEE, 2012.

[18] Venice E. Lion, Jiwen Lu, and Yongxin Ge. Regularized bayesian metric learning for person re-identification. In *ECCV Workshop on Visual Surveillance and Re-Identification*, 10 2014.

[19] Alexis Mignon and Frédéric Jurie. PCCA: A new approach for distance learning from sparse pairwise constraints. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2666–2672. IEEE, 2012.

[20] Timo Ojala, Matti Pietikainen, and Topi Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987, 2002.

[21] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer, 2010.

[22] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.

[23] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and algorithms*. MIT press, 2012.

[24] Chunhua Shen, Junae Kim, Lei Wang, and Anton Van Den Hengel. Positive semidefinite metric learning using boosting-like algorithms. *The Journal of Machine Learning Research*, 13(1):1007–1036, 2012.

[25] Karen Simonyan, Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Fisher Vector Faces in the Wild. In *British Machine Vision Conference*, 2013.

[26] Dapeng Tao, Lianwen Jin, Yongfei Wang, Yuan Yuan, and Xuelong Li. Person re-identification by regularized smoothing kiss metric learning. *Circuits and Systems for Video Technology, IEEE Transactions on*, 23(10):1675–1685, 2013.

[27] Vladimir N. Vapnik and Samuel Kotz. *Estimation of dependences based on empirical data*, volume 41. Springer-Verlag New York, 1982.

[28] Andrea Vedaldi and Brian Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[29] Nakul Verma, Dhruv Mahajan, Sundararajan Sellamanickam, and Vinod Nair. Learning hierarchical similarity metrics. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2280–2287. IEEE, 2012.

[30] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

[31] Fei Xiong, Mengran Gou, Octavia Camps, and Mario Sznaier. Person re-identification using kernel-based metric learning methods. In *ECCV 2014*, pages 1–16. Springer, 10 2014.

[32] Gang Yuan, Zhaoxiang Zhang, and Yunhong Wang. Enhancing person re-identification by robust structural metric learning. In *Image and Graphics (ICIG), 2013 Seventh International Conference on*, pages 453–458. IEEE, 2013.