Coordinated Local Metric Learning

Shreyas Saxena Jakob Verbeek
Inria*

Abstract

Mahalanobis metric learning amounts to learning a linear data projection, after which the ℓ_2 metric is used to compute distances. To allow more flexible metrics, not restricted to linear projections, local metric learning techniques have been developed. Most of these methods partition the data space using clustering, and for each cluster a separate metric is learned. Using local metrics, however, it is not clear how to measure distances between data points assigned to different clusters. In this paper we propose to embed the local metrics in a global low-dimensional representation, in which the ℓ_2 metric can be used. With each cluster we associate a linear mapping that projects the data to the global representation. This global representation directly allows computing distances between points regardless to which local cluster they belong. Moreover, it also enables data visualization in a single view, and the use of ℓ_2-based efficient retrieval methods. Experiments on the Labeled Faces in the Wild dataset show that our approach improves over previous global and local metric learning approaches.

1. Introduction

Metric learning is a machine learning technique with a wide range of applications in computer vision, *e.g.* local descriptor matching [10], fine-grained object comparison [27], and face verification [22]. Most work considers supervised learning of Mahalanobis metrics, see *e.g.* [9, 14, 15, 22, 25, 41]. The supervision comes as positive and negative pairs that should be close and far apart respectively. The Mahalanobis distance between two points is given by $(x_i - x_j)^\top M(x_i - x_j)$, where M is a positive definite matrix. Since M can always be factored as $M = L^\top L$, Mahalanobis metrics are equivalent to the ℓ_2 metric after linear projection of the data. For complex class distributions, however, linear projection of the data might not be sufficient to obtain a suitable data representation.

To overcome this restriction, several routes have been explored. First, the linear projection in the Mahalanobis metric can be written in terms of kernel evaluations, see *e.g.* [14, 16, 25]. Alternatively, (convolutional) neural networks with a siamese architecture can be learned to give (dis)similar outputs for positive and negative pairs, see *e.g.* [5, 8]. Approaches based on decision trees have also been explored, see *e.g.* [27]. Finally, local metric learning uses a collection of Mahalanobis metrics, each operating in a different part of the input space, see *e.g.* [3, 4, 12, 18, 20, 26, 34, 39, 41, 43]. The partitioning of the space is typically obtained using k-means or Gaussian mixture clustering.

In most existing local metric learning approaches, however, it is unclear how to compute distances between points assigned to different clusters, or distances are defined in an asymmetric manner. Unlike for global metric learning, they can not be interpreted as computing the ℓ_2 distance after a transformation of the data, which hinders data visualization and efficient ℓ_2-based retrieval techniques, such as product quantization and multiple-assignment retrieval [21].

In this paper we propose a solution by embedding the local metrics in a global representation. We use a Gaussian mixture model (GMM) to obtain a soft-partitioning of the data. With this partitioning we define a non-linear embedding of the input data vectors in a higher dimensional feature space. By learning a Mahalanobis metric over this embedding, we simultaneously learn local metrics for each cluster, and also obtain an alignment of the local metrics. This allows us (i) to compute distances between points regardless to which local cluster they belong, (ii) visualize data in a single view, and (iii) use efficient ℓ_2-based retrieval methods. We refer to our approach as "coordinated local metric learning" (CLML).

We validate our approach in face verification and retrieval settings using the Labeled Faces in the Wild (LFW) [19] dataset, and using image representations based on local binary patterns (LBP) [28], convolutional neural networks (CNN) [42], and Fisher vectors (FV) [35]. For all tested representations our approach improves over global metric learning and other local metric learning approaches. For retrieval, the improvements over previous local metric learning approaches [3, 34] are particularly large.

*LEAR team, Inria Grenoble Rhône-Alpes, Laboratoire Jean Kuntzmann, CNRS, Univ. Grenoble Alpes, France

2. Related work

In this section we give an overview of related work that is most relevant to the material we present in this paper.

Mahalanobis metric learning. Many supervised Mahalanobis metric learning methods exist. Most are based on loss functions defined over pairs or triplets of data points, see *e.g.* [9, 14, 15, 22, 25, 40, 41]. We refer the reader to recent survey papers [2, 23] for a detailed review of these.

Methods based on pairwise loss terms learn a metric so that positive pairs (*e.g.* points having the same class label) have a distance that is smaller than negative pairs (*e.g.* points with different class labels). An example of such methods is the logistic discriminant metric learning (LDML) method of Guillaumin *et al*. [15]. Their observation is that since Mahalanobis distances are linear in the entries of M, they can therefore be learned via standard logistic regression. In [16] they regularize by instead learning a low-rank factorization that projects the original data to a low dimensional space in which the ℓ_2 metric is used.

An example of a triplet-based approach is the large-margin nearest neighbor (LMNN) method [41]. Instead of forcing all points of the same class to be close, LMNN requires that the nearest neighbors of each point are of the same class. The metric is learned by minimizing a sum of loss terms over triplets of points, where each loss term encourages the distance between x_i and neighbors in the same class to be at least one distance unit smaller than the distance of x_i to neighbors in different classes.

Local metric learning. To alleviate the limitations of Mahalanobis metric learning, many local metric learning methods have been proposed, see *e.g.* [3, 4, 12, 18, 20, 26, 34, 39, 41, 43]. Here we limit our discussion to five recent state-of-the-art methods.

The R^{2}LML method of Huang *et al*. [20] jointly learns a set of local metrics and weights, g_i^s, that assign data points x_i to the local metrics indexed by s. The distance between x_i and x_j is computed using the weighted sum of metrics, where metric s is weighted by the product $g_i^s g_j^s$. They iteratively learn the weights and the metrics, updating one while keeping the other fixed. To determine the weights over the metrics for test points that were not included during training, the weights of the nearest training point are used, which implies a costly lookup when large training sets are used.

Shi *et al*. [34] proposed SCML, a metric learning approach based on sparse combinations of a large base set of rank-1 base metrics. The base metrics are found by clustering the dataset, and then applying Fisher linear discriminant analysis (FLDA) in each cluster. For local metric learning, they take a similar approach as [26, 39], and measure the distance between a test point x and a training point x_i by using a weighted combination of base metrics, where the weights are determined by x. During training, they learn a function that maps each data point to a set of weights over the base metrics. The advantage of their approach is that weights are easily evaluated for new test points. A limitation, however, is that a fixed set of base metrics given by FLDA restricts the class of metrics that can be learned. This is particularly detrimental for high-dimensional data.

Bohné *et al*. [4] proposed LMLML, an approach based on GMM clustering, which learns a metric associated with each cluster. To compare two points x_i and x_j they use a weighted sum of the local metrics, where the weight of each metric is given by $p(s|x_i) + p(s|x_j)$: the sum of the soft-assignments for x_i and x_j to the GMM components. If two points are far away, however, it is not clear that the local metric associated with either data point will be appropriate for a pair-wise comparison. Therefore, they also add a learned global metric to the weighted sum of metrics.

Bhattarai *et al*. [3] proposed a hierarchical method for efficient retrieval that learns a hierarchical clustering of the data by interleaving metric learning and k-means clustering. Each element in the training set is assigned to a leaf of the hierarchy based on the local metrics and clustering. A query is assigned to a leaf node, and retrieval is performed among the data in that leaf-node, using the associated metric. Their hierarchical decomposition speeds up the retrieval since only a fraction of the dataset is accessed for a given query. They report improved retrieval accuracy due to the use of local metrics, as compare to using global ones.

Unlike our proposed approach, none of these methods allow the local metrics to be expressed as the ℓ_2 distance after a non-linear data transformation. This means that the local metrics cannot be used for global data visualization, and do not support efficient retrieval techniques based on ℓ_2-quantization, such as product quantization and multiple-assignment retrieval [21].

The work of Hauberg *et al*. [17] is an exception in this respect: they show that if local metrics vary smoothly in the input space, then they form a Riemannian metric on the data manifold. They define a smoothly varying local metric as a linear combination of a fixed set of local metrics, which are learned separately using any local metric learning algorithm. They perform PCA in the Riemannian metric to obtain a global Euclidean data representation. They show their framework improves w.r.t. Euclidean PCA. Our approach differs in that (i) we learn the local metrics and their alignment in a joint manner, and (ii) to project a point to the global representation [17] requires solving a system second-order ODE's with size quadratic with the data dimension, whereas our approach requires only averaging local linear projections.

Our work is also related to the local linear manifold learning technique of Teh and Roweis [38]. They use a mixture of factor analyzers (MFA) [13] to map data points to local low dimensional coordinate systems associated with the mixture components. To align the local coordinates,

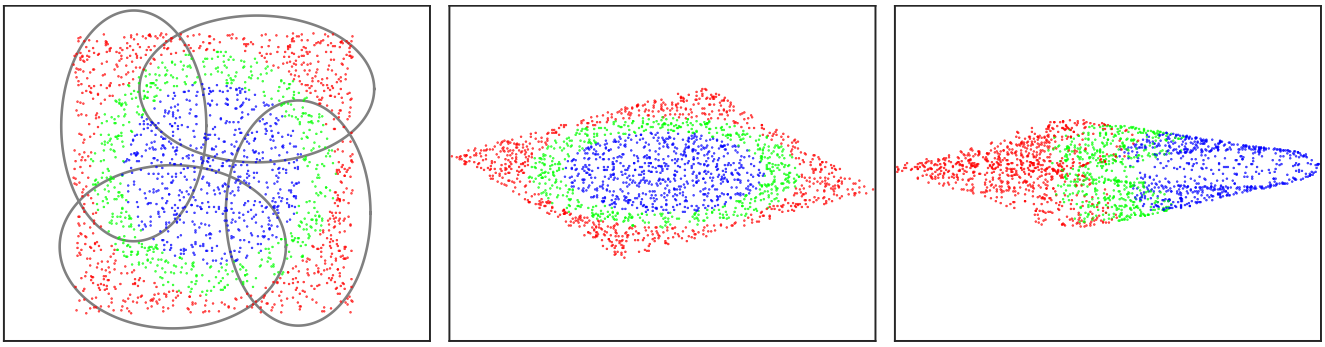

Figure 1. Synthetic dataset with color coded class labels, and the GMM used by our CLML local metric (left). Data projection given by a global Mahalanobis metric (middle) and our local CLML metric (right). The pairwise training constraints are better respected by CLML.

they minimize the Locally Linear Embedding (LLE) [30] objective function. Our work differs in that we learn coordinated local linear projections in a supervised manner. Also, in our work we use a diagonal covariance GMMs — which are faster to train than MFA— and learn local linear maps directly from the original feature space to the global representation instead of mapping from the local MFA subspaces. Since the MFA is learned by optimizing a different cost function, the obtained subspaces might be suboptimal.

3. Globally aligning local Mahalanobis metrics

Since any positive definite $D \times D$ matrix M can be decomposed as $M = L^\top L$, the Mahalanobis distance between two points x_i and x_j can be written as the ℓ_2 distance between these points after projection with L, *i.e.*

$$(x_i - x_j)^\top M(x_i - x_j) = \| Lx_i - Lx_j \|_2^2 \,. \qquad (1)$$

Note that L can be a $D \times D$ matrix, or a $d \times D$ matrix with $d < D$. In the latter case $(Lx_i) \in \mathbb{R}^d$ is a low dimensional projection of $x_i \in \mathbb{R}^D$, and $\text{rank}(M) \leq d$.

To obtain a more general class of metrics, we define several local Mahalanobis metrics. We cluster the data using a k-component Gaussian mixture model (GMM), which defines a soft-assignment of the data over the k clusters.

We can compute distances between points assigned to the same cluster s using a local metric learned for that cluster, defined by a local projection matrix L_s. It is, however, not clear how to compare vectors that are assigned to different clusters. In order to combine the local metrics, we define a global representation in which we integrate the local projections given by the different L_s. Similar to global Mahalanobis metric learning, and unlike previous work, our formulation amounts to projecting the data (in a locally linear way) to a new representation, and computing the ℓ_2 metric in this new representation. This allows us to compute distances between any pair of samples, regardless of their cluster assignments.

Figure 1 gives an illustration on a synthetic dataset. The dataset (left panel) is constructed in such a way that a global linear projection cannot bring all points of each class close together, while keeping points of different classes apart. This is observed in the data projected using a global Mahalanobis metric (middle panel). Using our "coordinated local metric learning" (CLML) approach, which we describe below, we obtain a data representation that respects the pairwise training constraints much better (right panel).

3.1. Coordinated local metric learning

As pointed out above, we can interpret a Mahalanobis metric as computing the ℓ_2 distance after linear projection of the data. Local Mahalanobis metrics can therefore be interpreted as locally mapping the data points x to several different, local, coordinate systems via projections $L_s x$.

Since the ℓ_2 metric is invariant to translation, rotation, and reflection of the coordinates, we can arbitrarily modify the projection of x_i to a local coordinate system to

$$z_{is} := R_s L_s x_i + b_s, \qquad (2)$$

where R_s denotes an orthonormal matrix, *i.e.* for which $R_s^\top R_s = I$, which can implement rotations and reflections, and b_s denotes a translation. Using these transformations we can coordinate the local projections so that they align across different local models. In particular, given that x_i and x_j are assigned to different clusters r and t respectively, we can set the $\{R_s, b_s\}$ to ensure that z_{ir} and z_{jt} are close if x_i and x_j form a positive pair, and far away if they form a negative pair.

Instead of learning the $\{R_s, b_s\}$ for fixed L_s that were learned in advance, we will learn both the local metrics and their alignment in a joint manner. To that end, we can absorb R_s into L_s without loss of generality, and define a mapping of the data points x_i to a global coordinate system as

$$z_i := \sum_{s=1}^{k} q_{is} z_{is} = \sum_{s=1}^{k} q_{is}\big(L_s x_i + b_s\big), \qquad (3)$$

where $q_{is} := p(s|x_i)$ is the soft-assignment of x_i to cluster s. In the case of hard-assignments z_i is given by the local

projection of the cluster to which it is assigned. In the case of soft-assignments, z_i is the weighted average of the local projections to the global representation. Our goal is now to learn $\{L_s, b_s\}$ so that z_i and z_j are close for positive pairs, and far away for negative pairs. We assume the GMM clustering is fixed.

Note that we can re-write this weighted average of locally linear projections as a single linear projection

$$z_i = \tilde{L}\tilde{x}_i, \tag{4}$$

where $\tilde{L} = (L_1, b_1, \ldots, L_k, b_k)$ collects the local linear projections, and $\tilde{x}_i = \left(q_{i1}(x_i^\top, 1), \ldots, q_{ik}(x_i^\top, 1)\right)^\top$ contains k copies of x_i appended with a one, each weighted by the corresponding soft-assignment. The projection matrix $\tilde{L}$ defines the local metrics used to compare points that are assigned to the same cluster, but also the rotations, reflections, and translations to globally align the local representations.

For a given partitioning of the data, the z_i are obtained as a linear projection of the transformed input vectors $\tilde{x}_i$, the ℓ_2 distance between z_i and z_j is therefore equivalent to a Mahalanobis distance between $\tilde{x}_i$ and $\tilde{x}_j$. The problem of learning a globally aligned ensemble of local metrics therefore takes the same form of learning a global Mahalanobis metric; be it using the expanded high-dimensional data representation given by the $\tilde{x}_i$. As a result, existing Mahalanobis metric learning methods can be used to learn the projection matrix $\tilde{L}$ for CLML.

It is easy to see that CLML generalizes Mahalanobis metrics. For $k \geq 1$, if each cluster s uses the same projection given by $L_s = L$ and $b_s = b$, then for arbitrary soft-assignments $z_i = Lx_i + b$ is a linear projection of x_i, and the ℓ_2 distance between z_i and z_j is a Mahalanobis distance between x_i and x_j given by $\| L(x_i - x_j) \|_2$. With proper regularization, we therefore expect performance that is at least on par with global metric learning.

3.2. Implementation

Optimization. We use the LDML [15] objective function to learn our local metrics parameterized by $\tilde{L}$. Let the label $y_{ij} \in \{-1, +1\}$ denote whether (x_i, x_j) is a positive or a negative pair. LDML then minimizes the log-loss

$$\mathcal{L}(\tilde{L}, b) = \sum_{i,j} \ln\left\{1 + \exp\left(-y_{ij}(b - ||z_i - z_j||^2)\right)\right\}, \tag{5}$$

where b is a scalar (estimated along with $\tilde{L}$) that determines at which distance pairs are considered positive or negative. We add a Frobenius norm regularizer over $\tilde{L}$ to avoid overfitting, and cross-validate the regularization weight. We use a global LDML metric to initialize the local metrics.

In our implementation we use the sum formulation of Eq. (3), which avoids explicitly storing the $\tilde{x}_i$. Compared to global metric learning, we only need to additionally store the soft-assignments. In practice this is a negligible overhead, in addition the assignments can be thresholded to be sparse. The cost to compute the z_i increases sub-linearly with k because of this sparsity.

Clustering. To partition the input space in CLML, we learn diagonal covariance GMMs. In our experiments we consider two alternatives for the data on which the GMMs are learned. We either learn the mixture in the original feature space, or learn the mixture in the projection space obtained by global LDML metric learning. The rationale for the latter option is that the GMM clustering will be more meaningful in the global metric learning space.

Efficient retrieval. For efficient face retrieval we use the multiple assignment approach of [21]. The z_i in the retrieval set are clustered using k-means, and each z_i is assigned to the nearest center. A query z is assigned to the m closest k-means centers. Only points assigned to these m centers are returned, ranked by their distances to the query z.

4. Experimental Evaluation

4.1. Dataset, protocols, and features

For our experiments we use the Labeled Faces in the Wild (LFW) [19] dataset. It contains a total of 13,233 faces of 5,749 people collected from the web. The dataset was designed for verification experiments, where we have to determine for a pair of face images if they depict the same person or not. In our experiments, we use the standard "unrestricted" training protocol, that allows the use of all pairs in the training set. The verification accuracy is measured using ten-fold cross-validation. The train set is used to learn a metric, and to estimate a threshold on the metric. Using these, the pairs in the test set is classified as positive or negative, and the accuracy of this classification is reported.

Since the LFW verification accuracy is saturating in recent years [24], we focus on the more challenging retrieval-based evaluation of Bhatterai *et al*. [3]. The set of 423 queries consists one image of each person in LFW with five or more images. All images not in the query set form the retrieval set, and are used to learn the metric. We augment the retrieval set with up to one million distractor faces provided by Bhatterai *et al*., which belong to people not present in the LFW dataset. The 1-call@n performance measure is the fraction of queries for which at least one of the top n ranked result faces is of the same person. We also use the mean average precision (mAP) measure, which gives us a single number per setting instead of a full 1-call@n curve.

We align face images with a similarity transform based on detection of points on the eyes, nose, and mouth, see [11]. We consider three representations. The first is the LBP features of Bhattarai *et al*. [3], which allows direct comparison to their work. We compute a 9,860 dimensional descriptor by concatenating 58 dimensional LBPs [28] on

	GMM feature space	
Nr. of local metrics k	Original	Global metric
2	73.04	73.09
4	74.04	75.18
6	73.58	75.24
8	73.92	75.59

Table 1. Evaluation of CLML over FV features using different GMM clustering methods. Performance is measured in mAP.

each cell of a 10×17 grid over the face. The second is similar to the Fisher vector (FV) features of Simonyan *et al*. [35]. We densely compute at each pixel a root-SIFT descriptor [1], using 24×24 pixel patches. The descriptors are projected to 64 dimensions using PCA. Spatial layout information is incorporated by appending the 2D image coordinates of the patch center to the descriptors [31], providing a 66 dimensional local feature. We represent the face by computing a 16,896 dimensional FV [32] using 128 Gaussian components. The third feature is derived from the penultimate layer of a convolutional neural network trained on the CASIA WebFace dataset [42], which contains 494,414 faces of 10,575 subjects. The network architecture is similar to the one proposed in [42] and the dimensionality of the extracted feature is 320.

4.2. Experimental evaluation results

For comparability with Bhattarai *et al*. [3], we use projections to $d=32$ dimensions unless stated otherwise.

Comparison of CLML with global metric learning. In Table 1 we compare CLML using GMMs trained either in the original FV features, or on data projected to $d=32$ dimensions by a global LDML metric, *c.f*. Section 3.2. In all cases, the clustering obtained LDML projections leads to better results. Where the difference between the two clustering approaches is only 0.05 for $k=2$ local metrics, it increases for larger numbers of clusters, up to 1.67 for $k=8$ local metrics. In all subsequent experiments we therefore use clustering using LDML projections. Clustering in the LDML projected space is also much faster: in this case it takes about 0.5 secs. to learn the GMM on 10,000 points.

In Table 2 we evaluate CLML on the three features, while varying the number of local metrics. We also state results obtained when cross-validating the number of local metrics, as well as results of global LDML metrics and the ℓ_2 metric. The results lead to the following observations. (i) CLML generally improves when using more local metrics. (ii) Cross-validation over the number of local metrics successfully selects a (near) optimal number of local metrics. In subsequent experiments we cross-validate the number of local metrics for CLML. (iii) The FV features lead to better results than the LBP and CNN features. (iv) For all tested

	Features		
Nr. local metrics k	LBP	FV	CNN
2	41.51	73.09	59.78
4	44.02	75.18	61.43
8	47.94	75.59	64.64
16	49.00	76.20	66.07
32	49.98	75.61	70.98
64	49.70	75.58	73.83
Cross-validated	49.89 (26)	74.99 (28)	73.83 (64)
Global LDML metric	36.95	68.12	58.46
ℓ_2 metric	13.24	22.88	63.06

Table 2. Performance of CLML in retrieval mAP for the three features, while varying the number of local metrics.

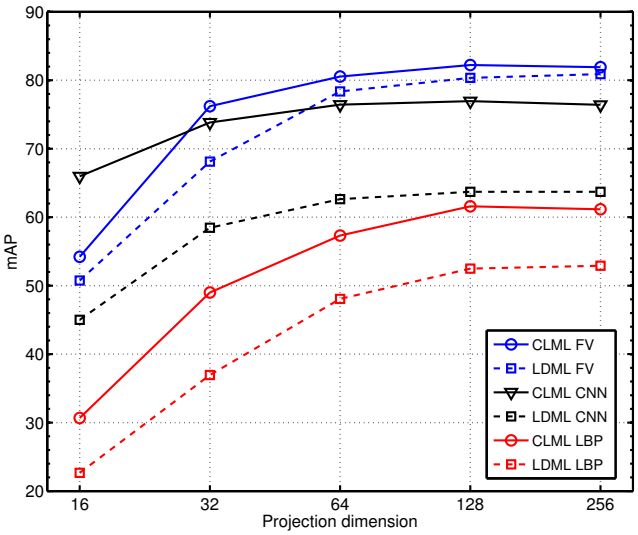

Figure 2. Performance in mAP of local and global metrics for the three features, using projection dimensions from 16 to 256.

settings, CLML consistently improves over LDML.

In Figure 2 we compare CLML and LDML for the three features across a range of projection dimensions. The results show that CLML consistently improves over LDML for all projection dimensions with the three features. The improvements are particularly large for the CNN and LBP features. For the LBP and FV features, the best results are obtained with CLML at $d = 128$, with $k = 16$ set by cross-validation: 61.6% and 82.2% mAP respectively. LDML with the same number of parameters, *i.e*. with $d = 128 \times 16 = 2048$, obtains 53.0% and 80.9% respectively. This shows that the improvement of CLML is not simply because it has more parameters. In the case of CNN descriptors, the best performance is obtained with CLML at $d = 128$ and $k = 64$ set by cross-validation: 76.95% mAP. Since the descriptors are only 320 dimensional, we cannot compare to LDML with the same number of parameters.

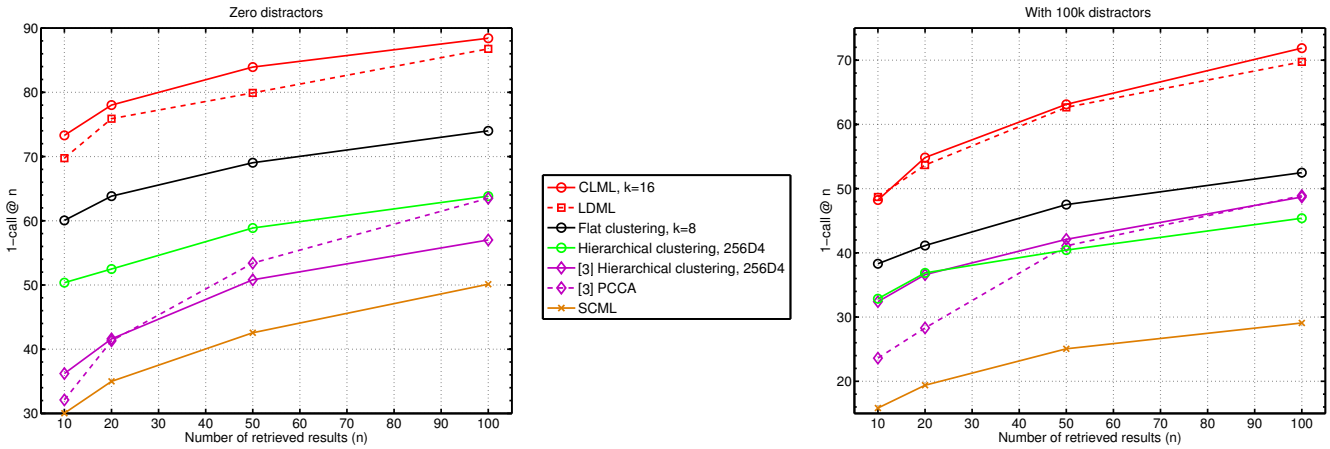

Figure 4. Retrieval using LFW images only (left), and using LFW plus 100,000 distractor faces (right). The results marked with [3] correspond to those reported therein. Results for SCML and LDML have been produced using publicly available code. See text for details.

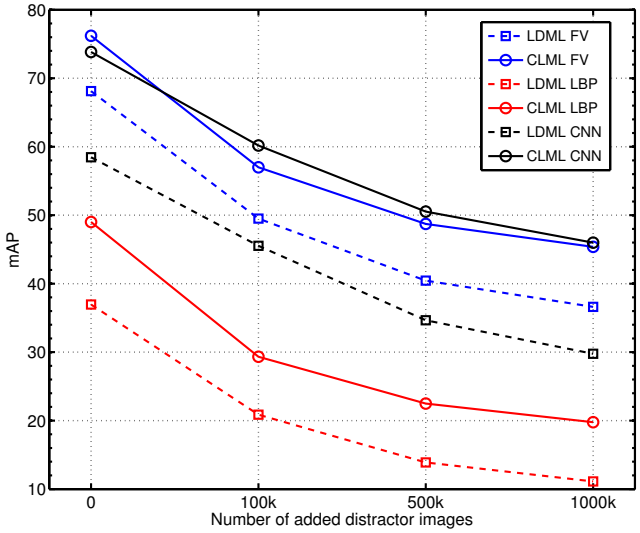

Figure 3. Performance of CLML and LDML with different numbers of distractors added to the LFW images.

Using a full-rank 320×320 Mahalanobis metric, however, yields 63.71% mAP, which is significantly worse.

Large-scale face retrieval experiments. In our second set of retrieval experiments we add up to one million additional distractor images to the LFW images.

In Figure 3 we evaluate the results of CLML and LDML for the three features, while increasing the number of distractors from zero to one million. We observe that performance degrades gracefully, and that the improvement of CLML over LDML is stable as a function of the number of distractors for all three features.

In Figure 4 we make a direct comparison to Bhattarai *et al*. [3]: re-plotting the 1-call@n curves reported there. From the other state-of-the-art methods discussed in Section 2 we compare to SCML [34].[1] For LMLML [4] code is not available, while for R^{2}LML [20] we found the code too inefficient to use with our high dimensional features.

For Bhattarai *et al*. we report the results for their "256D4" setting, which they found to give best results and uses eight local metrics, and also include their global metric learning results obtained with PCCA [25]. Our CLML results substantially improve over the results of Bhattarai *et al*., *e.g*. from under 40% to over 70% 1-call@n for $n = 10$ for the case without distractors (Figure 4, left panel). Interesting we also obtained large improvements over Bhattarai *et al*. using global LDML metrics.

To understand the large performance difference, we re-implemented their approach (Figure 4, left panel, green curve), and obtained improvements of about 10 points w.r.t. their results. We found that most of this improvement is due to the ℓ_2 regularization that we use, but Bhattarai *et al*. did not. We also implemented a non-hierarchical variant of their approach, based on "flat" k-means clustering, but which is otherwise the same (Figure 4, left panel, black curve). This leads to another improvement of about 10 points, which suggests that flat clustering leads to clusters that are better suited for retrieval. Our global metric learning results obtained with LDML (Figure 4, left panel, red dashed curve) are yet another 10 points better. This shows that the benefit of using local metrics is counterbalanced by only retrieving points assigned to the same cluster as the query, as is done by Bhattarai *et al*. and for the green and black curves.

Using SCML [34] we obtained the worst retrieval results. This is because SCML learns metrics using a limited set of base metrics, which is detrimental for high-dimensional data. To improve results we tuned the number of base metrics (600 gave best results), and also excluded faces of people with less than 3 images to compute the base metrics with FLDA, which also improved the results. The number of clusters used to produce the base metrics in SCML is another hyper-parameter that might require further tuning.

[1] Code available at http://mloss.org/software/view/553.

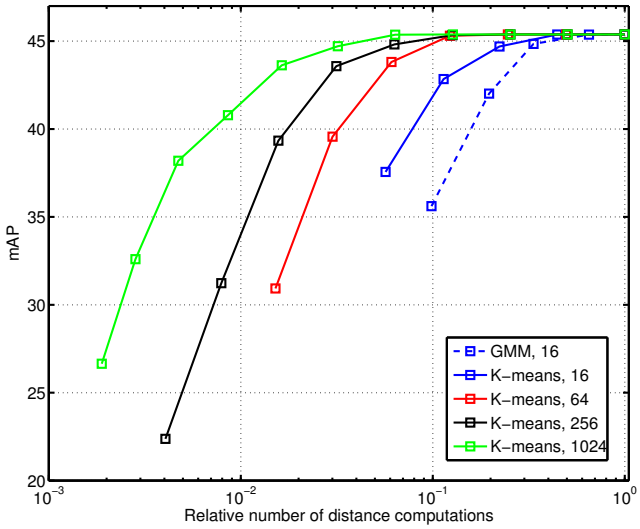

Figure 5. Retrieval mAP and speed on LFW plus one million distractor faces, using CLML with $d = 32$ and FV feaures. Varying the number of quantization cells p, and number of assignments m.

Efficient retrieval with CLML metrics. CLML projects all data in a single representation in which the ℓ_2 distance is used. Therefore we can decouple the clustering used for local metric learning, and the clustering used for the efficient quantization-based multiple-assignment retrieval method discussed in Section 3.2.

In Figure 5 we consider the trade-off between the retrieval mAP and search speed. The speed is measured as number of distance computations relative to the number needed for exhaustive search. Each curve shows, for a quantization into p cells, the performance using multiple assignment to $m = 1, 2, 4, \ldots, p$ clusters, where $m = 1$ corresponds to the lower-left point of each curve.

The blue curves show performance using 16 clusters: either using a k-means clustering computed over the z_i (solid), or using the GMM clustering used for the local metrics (dashed). Using the GMM clustering (dashed blue curve), a speedup factor 10 relative to exhaustive search can be achieved by single assignment ($m = 1$), but at the cost of a drop of around 10 points in mAP. Using k-means clustering over the z_i (solid blue curve) we obtain larger speedups and higher mAP values. The results show that it is more effective to dissociate the clustering used for the local metrics from the one used for retrieval, unlike the approach taken by Bhattarai *et al*. [3].

Moreover, dissociating the clusterings, allows more flexibility in choosing the speed-*vs*.-accuracy operating point. By using k-means clustering with more than 16 centers we can substantially improve the search results: as seen by the red, black, and green curves for p equal to 64, 256, and 1024 respectively. For example, with $p = 1024$ clusters (green curve) and assignment to $m = 64$ clusters we can reduce the search time by a factor 14, without compromising the mAP.

Using LFW training data		
Guillaumin *et al*. [15]	12K	87.5 ± 0.4
Chen *et al*. [7]	12K	93.2 ± 1.1
Simonyan *et al*. [35]	12K	93.0 ± 1.1
Ours, FV, without metric learning	12K	78.9 ± 0.9
Ours, FV, LDML, $d = 768$	12K	92.3 ± 0.5
Ours, FV, LDML, $d = 128$	12K	92.1 ± 0.5
Ours, FV, CLML, $d = 128$ $(k = 6)$	12K	92.8 ± 0.4
Ours, FV, LDML, $d = 32$	12K	91.6 ± 0.4
Ours, FV, CLML, $d = 32$ $(k = 4)$	12K	92.4 ± 0.5
Using external training data		
Taigman *et al*. [37]	4.4M	97.4 ± 0.3
w/o metric learning, 2D alignment	4.4M	94.3 ± 0.4
Yi *et al*. [42]	500K	97.7 ± 0.3
w/o metric learning	500K	96.3 ± 0.3
Parkhi *et al*. [29]	2.6M	99.0
w/o metric learning	2.6M	97.3
Schroff *et al*. [33]	200M	99.6
Ours, CNN, w/o metric learning	500K	96.2 ± 0.8
Ours, CNN, LDML, $d = 128$	500K	96.5 ± 1.0
Ours, CNN, CLML, $d = 128$ $(k = 2)$	500K	96.4 ± 0.9

Table 3. Comparing CLML using FV features with other metric learning methods. Performance as LFW verification accuracy.

Our speedup is comparable to the factor of 10 reported by Bhatterai *et al*. [3] for 16 clusters in their hierarchical approach, but our approach leads to better retrieval results. With $p = 1024$ and $m = 8$, a speedup factor larger than 100 can be obtained while loosing less than 5 mAP points.

Face verification experiments. In Table 3 we compare our results obtained using local CLML metrics and global LDML ones to the state-of-the-art using the LFW face verification evaluation.

When using no outside training data, the results of Chen *et al*. [7] (93.2 ± 1.1) and Simonyan *et al*. (93.0 ± 1.1) are sate-of-the-art. Using the ℓ_2 metric as a baseline we obtain 78.9 ± 0.9, which is improved using global LDML metrics to 92.1 ± 0.5 and 91.6 ± 0.4 for $d = 128$ and $d = 32$ dimensional projections respectively. For both projection dimensions, CLML improves over LDML, to 92.8 ± 0.4 and 92.4 ± 0.5 respectively. We also observe a consistent improvement when comparing LDML ($d = 128$ and $d = 768$) with CLML ($d = 32, k = 4$ and $d = 128, k = 6$) using the same number of parameters. This underlines once more that the improvements by CLML are not simply due a larger number of parameters.

Our results differ slightly from those of Simonyan *et al*. [35] due to a more efficient implementation: (i) They used GMMs with 512 components for the FV, while we use only

Figure 6. Visualization of the data projections learned by LDML (left) and CLML (right). Data points of the 40 most frequent people in the dataset have been color coded. Other data points are plotted in blue and pink for males and females respectively. On the sides of the CLML visualization we show outliers faces (marked with black circles) of males in the female cluster (right), and vice-versa (left). Interestingly, male outlier faces are mostly young boys, while female outlier faces mostly display extreme poses or expressions. Best viewed on screen.

128, yielding $4\times$ smaller descriptors. (ii) They average over left-right flipped versions of face image which we do not. (iii) Besides a Mahalanobis metric, they also learn a similarity of the form $x_i^T M x_j$, which they average with the Mahalanobis metric, similar to Cao *et al*. [6].

In the bottom part of Table 3 we report results obtained using CNN features. Using the CNN features with the ℓ_2 metric we obtain 96.2 ± 0.8 verification accuracy, similar to the results of Yi *et al*. [42] (96.3 ± 0.3) which used the same training data for their network. Surprisingly, using our CNN descriptors we found that metric learning, either with LDML or CLML, gives only small improvements over the ℓ_2 baseline. The reason for this might be that the performance of the ℓ_2 distance over the CNN features is already very high for the face verification task, or that the pair-wise loss function of LDML is less suitable for verification than the triplet-based loss used by Parkhi *et al*. [29], or the weighted chi-squared metric used by Taigman *et al*. [37]. Yi *et al*. [42] used a multi-task learning objective to train their CNN jointly for both verification and recognition.

The quoted results from the literature other than [42], are using CNNs trained on datasets that are 5 to 400 times larger, and therefore not directly comparable. Taigman *et al*. [37] use 4.4 million images and combine the output of three different CNNs and use 3D face alignment. Using only 2D aligned images (as we do in our work), they reported slightly worse than ours before metric learning (94.3 ± 0.4). Parkhi *et al*. [29] recently reported results using a deeper convolutional architecture [36] and 2D face alignment over 2.6 million images (99.0). The state of the art results of Schroff *et al*. [33] are based on an extremely large proprietary dataset of 200 million images, for which no alignment was used.

Data visualization. To illustrate the benefit of CLML for data visualization we plot LFW images projected using CLML and LDML in Figure 6. We learned $d = 256$ dimensional projections on the FV features, and map these to 2D by PCA. For CLML the number of local metrics was set to $k = 12$ by cross-validation. CLML leads to a much better separation of the faces of different people, despite the limited improvement of CLML (81.9) over LDML (80.8) in mAP for $d = 256$. Using CLML we can more clearly see the two groups corresponding to male and female faces. We used the LFW gender labels from the BeFIT website.[2]

5. Conclusion

We have presented our coordinated local metric learning (CLML) approach which learns local Mahalanobis metrics, and integrates them in a global representation where the ℓ_2 distance is used. This allows data visualization in a single view, and the use of efficient ℓ_2-based retrieval methods. Our low-dimensional global representation is obtained as a linear projection of an expanded data representation, defined using the input data and a Gaussian mixture clustering. We have presented results of extensive face retrieval and verification experiments on the Labeled Faces in the Wild dataset. In all settings CLML improves over global LDML metrics, or gives comparable results. For face retrieval we obtain substantial improvements over global metrics and previously reported local metric learning results. Our approach also allows efficient multiple-assignment retrieval, which gives a better speed-accuracy trade-off than earlier work for face retrieval in a large-scale dataset with a million distractor faces.

[2] See http://fipa.cs.kit.edu/431.php

References

[1] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] A. Bellet, A. Habrard, and M. Sebban. A Survey on Metric Learning for Feature Vectors and Structured Data. *ArXiv e-prints*, 1306.6709, 2013.

[3] B. Bhattarai, G. Sharma, F. Jurie, and P. Pérez. Some faces are more equal than others: Hierarchical organization for accurate and efficient large-scale identity-based face retrieval. In *ECCV Workshops*, 2014.

[4] J. Bohné, Y. Ying, S. Gentric, and M. Pontil. Large margin local metric learning. In *ECCV*, 2014.

[5] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature verification using a siamese time delay neural network. In *NIPS*, 1993.

[6] Q. Cao, Y. Ying, and P. Li. Similarity metric learning for face recognition. In *ICCV*, 2013.

[7] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: high dimensional feature and its efficient compression for face verification. In *CVPR*, 2013.

[8] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[9] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *ICML*, 2007.

[10] A. Dosovitskiy, J. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, 2014.

[11] M. Everingham, J. Sivic, and A. Zisserman. Taking the bite out of automatic naming of characters in TV video. *Image and Vision Computing*, 27(5):545–559, 2009.

[12] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007.

[13] Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, May 1996.

[14] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *NIPS*, 2006.

[15] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *ICCV*, 2009.

[16] M. Guillaumin, J. Verbeek, and C. Schmid. Multiple instance metric learning from automatically labeled bags of faces. In *ECCV*, 2010.

[17] S. Hauberg, O. Freifeld, and M. Black. A geometric take on metric learning. In *NIPS*, 2012.

[18] Y. Hong, Q. Li, J. Jiang, and Z. Tu. Learning a mixture of sparse distance metrics for classification and dimensionality reduction. In *ICCV*, 2011.

[19] G. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[20] Y. Huang, C. Li, M. Georgiopoulos, and G. Anagnostopoulos. Reduced-rank local distance metric learning. In *ECML*, 2013.

[21] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*, 33(1):117–128, 2011.

[22] M. Köstinger, M. Hirzer, P. Wohlhart, P. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *CVPR*, 2012.

[23] B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

[24] S. Liao, Z. Lei, D. Yi, and S. Li. A benchmark study of large-scale unconstrained face recognition. In *International Joint Conference on Biometrics*, 2014.

[25] A. Mignon and F. Jurie. PCCA: A new approach for distance learning from sparse pairwise constraints. In *CVPR*, 2012.

[26] Y.-K. Noh, B.-T. Zhang, and D. Lee. Generative local metric learning for nearest neighbor classification. In *NIPS*, 2010.

[27] E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *CVPR*, 2007.

[28] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7):971–987, 2002.

[29] O. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015.

[30] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[31] J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33(16):2216–2223, 2012.

[32] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.

[33] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. 2015.

[34] Y. Shi, A. Bellet, and F. Sha. Sparse compositional metric learning. In *AAAI*, 2014.

[35] K. Simonyan, O. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. In *BMVC*, 2013.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[37] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[38] Y. Teh and S. Roweis. Automatic alignment of local representations. In *NIPS*, 2003.

[39] J. Wang, A. Kalousis, and A. Woznica. Parametric local metric learning for nearest neighbor classification. In *NIPS*, 2012.

[40] J. Wang, K. Sun, F. Sha, S. Marchand-Maillet, and A. Kalousis. Two-stage metric learning. In *ICML*, 2014.

[41] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009.

[42] D. Yi, Z. Lei, S. Liao, and S. Li. Learning face representation from scratch. In *Arxiv preprint*, 2014.

[43] D.-C. Zhan, M. Li, Y.-F. Li, and Z.-H. Zhou. Learning instance specific distances using metric propagation. In *ICML*, 2009.