Large Scale Metric Learning, A Voyage From Shallow to Deep

Masoud Faraki MASOUD.FARAKI@DATA61.CSIRO.AU

Mehrtash T. Harandi MEHRTASH.HARANDI@DATA61.CSIRO.AU

Fatih Porikli FATIH.PORIKLI@DATA61.CSIRO.AU

Research School of Engineering
Australian National University
Canberra, ACT 0200, Australia

Data61, CSIRO
Locked Bag 8001
Canberra, ACT 2601, Australia

Editor:

Abstract

Despite its attractive properties, the performance of the recently introduced Keep It Simple and Straightforward MEtric learning (KISSME) is greatly dependent on PCA as a preprocessing step. This dependency can lead to difficulties, e.g., when the dimensionality is not meticulously set. To address this issue, we devise a unified formulation for joint dimensionality reduction and metric learning based on the KISSME algorithm. Our joint formulation is expressed as an optimization problem on the Grassmann manifold, hence enjoys properties of Riemannian optimization techniques. We also devise end-to-end learning of a generic deep network for metric learning using our derivation.

Keywords: Mahalanobis Metric Learning, Dimensionality Reduction, Deep Metric Learning, Riemannian Geometry

1. Introduction

Metric learning algorithms are of practical interest when learning from large number of categories (with limited training samples per category) is deemed, if the machinery is meant to deal with unseen classes (e.g., retrieval), or if weaker forms of supervision are considered. In such scenarios, conventional classification approaches are either not applicable or may fail miserably.

The Keep It Simple and Straightforward MEtric learning (KISSME) Koestinger et al. (2012) algorithm is agnostic to the class labels and learns a metric purely from a set of equivalence constraints (similar/dissimilar pairs). Furthermore, the algorithm scales beautifully to large scale problems, makes it a suitable -if not perfect- match for the aforementioned problems.

Some recent studies (e.g., Xiong et al. (2014)) show that the KISSME algorithm is successful only when its input is carefully processed and denoised using PCA. This in turn results in high degree of sensitivity to the dimensionality of the PCA step as shown in Fig. 1. In this paper, we propose to learn a low-dimensional subspace along its metric in the spirit of the KISSME algorithm in a unified fashion. Furthermore, based on our derivation, we show end-to-end learning of a generic

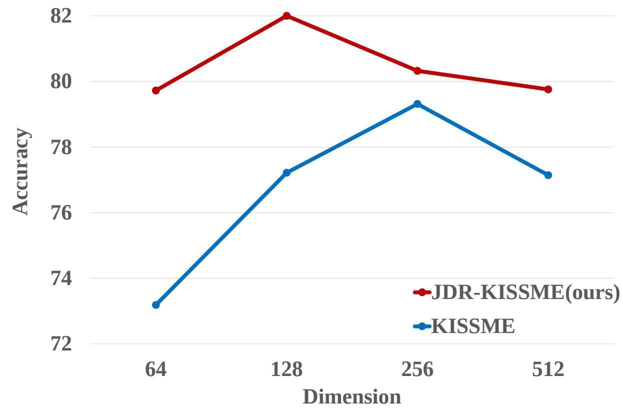

Figure 1: Verification accuracy against dimensionality of the space for an experiment using the surveilance nature images of the CompCars dataset (see §4.3 for more details).

deep Convolutional Neural Network (CNN) for metric learning. In short, our contributions in this paper are

1. We propose a Joint Dimensionality Reduction formulation for KISSME algorithm (JDR-KISSME) that learns a low-dimensional space along its metric in the spirit of the KISSME. In doing so, we benefit from the optimization techniques on Riemannian manifolds Absil et al. (2009) and in particular the geometry of Grassmann manifolds.

2. Upon our development, we propose a few simple, yet effective steps, to train a deep network for metric learning using KISSME verification signal as supervision.

2. Related Work

We review some notable examples of conventional and deep metric learning techniques here. We start by two studies in the restricted mode and follow it up by a classical method devised for the unrestricted case[1].

Mahalanobis Metric for Clustering (MMC) Xing et al. (2003) aims to minimize sum of distances over similar pairs while ensuring dissimilar pairs are far apart. The problem is formulated as an iterative gradient descent algorithm where at each iteration the obtained solution is projected back to the set of Positive Semi-Definite (PSD) matrices to ensure the metric is proper. Projection onto the PSD cone requires eigenvalue decomposition, making MMC computationally expensive when dealing with high-dimensional data. Pairwise Constrained Component Analysis (PCCA) Mignon and Jurie (2012) learns a transformation to project similar pairs inside a ball while dissimilar pairs are pushed away. Optimization is performed by making use of the gradient descent method.

Large Margin Nearest Neighbor (LMNN) Weinberger and Saul (2009) learns a linear transformation of labeled input data to improve kNN classification accuracy. Ideally, the learned transfor-

1. Algorithms in restricted metric learning scenario do not have access to the class labels of the samples. These algorithms can also work in the unrestricted scenario while the opposite is not often the case. This makes the restricted algorithms more appealing as they can address a broader range of problems.

mation is expected to unite the k-nearest neighbors of each point sharing the same label (called target neighbors) while minimizing the number of impostors. This is implemented as a semi-definite programming problem and solved by iterating between a gradient descent step followed by projecting the solution onto the PSD cone.

2.1 Metric Learning and Deep Nets

Similarity/distance metric learning using deep nets originates from the advent of Siamese networks Bromley et al. (1993); Chopra et al. (2005). In recent years, deep metric learning has received growing attention, following the trend of deep CNNs in solving large-scale classification problems. For example, in the spirit of PCCA, a two layer discriminative network for face verification is proposed in Hu et al. (2014). Sun et al. (2014) use an ensemble of networks, each operating on a different face patch for face verification. The networks are trained by making use of a combination of classification and verification cost functions.

Very recent works in deep metric learning include the work of Hoffer and Ailon (2015) and Schroff et al. (2015) in which an LMNN based triplet loss layer is used to direct CNN parameter learning. Song et al. (2016) show careful construction of batches such as including hard triplets during training, leads to better clustering and retrieval qualities.

3. Proposed Method

In this part, we present our idea to jointly learn a low-dimensional space and its metric. Throughout the paper, we use bold lower-case letters (e.g., $\boldsymbol{x}$) to denote vectors and bold upper-case letters (e.g., $\boldsymbol{X}$) to show matrices. $\mathbf{I}_d$ is the $d \times d$ identity matrix. $\det(\cdot)$ shows the matrix determinant. $[\cdot]_+$ indicates the hinge loss function, i.e., $max(0, \cdot)$.

3.1 Problem Statement

Let $\{\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i\}_{i=1}^N$, $\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i \in \mathbb{R}^D$ be a set of N training pairs. Furthermore, let $l_i \in \{-1, 1\}$ denote the label of the i-th pair with $l_i = 1$ indicating that the corresponding pair is similar and $l_i = -1$ otherwise. Koestinger et al. (2012) propose to measure the probability of a pair being (dis)similar by multivariate Gaussian distributions. In particular and by making use of the symmetry in decisions[2], the function

$$\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i) = \frac{\mathcal{N}\Big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i \big| \Sigma_D, \mathbf{0}\Big)}{\mathcal{N}\Big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i \big| \Sigma_S, \mathbf{0}\Big)} \tag{1}$$

models the dissimilarity value of the pair $\big(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i\big)$. Here, $\mathcal{N}\Big(\boldsymbol{x} \big| \Sigma, \mu\Big)$ denotes the multivariate Gaussian with mean μ and covariance Σ evaluated at $\boldsymbol{x}$ and

$$\Sigma_D = \frac{1}{\#(l_i = -1)} \sum_{i, l_i = -1} \big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)\big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)^T . \tag{2}$$

$$\Sigma_S = \frac{1}{\#(l_i = +1)} \sum_{i, l_i = +1} \big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)\big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)^T . \tag{3}$$

2. The discrimination value between the pair $\big(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i\big)$ should match that of $\big(\tilde{\boldsymbol{x}}_i, \boldsymbol{x}_i\big)$.

Intuitively, if the pair $\big(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i\big)$ is dissimilar, one expects $\mathcal{N}\Big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i \big| \Sigma_D, \mathbf{0}\Big)$ attains a high value while $\mathcal{N}\Big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i \big| \Sigma_S, \mathbf{0}\Big)$ being small. This makes $\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)$ large while the opposite happens if the pair $\big(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i\big)$ is similar. The lemma below provides the basis of metric construction using $\delta(\cdot,\cdot)$ as suggested in Koestinger et al. (2012) and is central to our developments presented in the following section.

Lemma 1 *Let $\delta : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}^+$ be the function defined in Eq. (1). The $\log(\delta)$ approximates a form of Mahalanobis metric, up to a constant term in $\mathbb{R}^D$. The approximated Mahalanobis metric is recognized as $\boldsymbol{M} = \mathrm{Proj}(\Sigma_S^{-1} - \Sigma_D^{-1})$ with $\mathrm{Proj}(\cdot) : \mathrm{Sym}(D) \to \mathcal{S}_{++}^D$ being the projection from the set of $D \times D$ symmetric matrices to the cone of Symmetric Positive Definite (SPD) matrices.*

Proof We note that

$$\log\big(\delta(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i)\big) = \frac{1}{2}\Big(\log\det(\Sigma_S) - \log\det(\Sigma_D) + (\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^T(\Sigma_S^{-1} - \Sigma_D^{-1})(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)\Big)\ . \quad (4)$$

The first two terms are constant and hence can be removed without loss of generality. From $(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^T\boldsymbol{M}(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i), \boldsymbol{M} \in \mathcal{S}_{++}^D$, (i.e., general form of the squared Mahalanobis distance) and by noting that $\Sigma_S^{-1} - \Sigma_D^{-1}$ is not necessarily an SPD matrix, we conclude that an approximated Mahalanobis metric associated to $\log(\delta)$ has the form $\boldsymbol{M} = \mathrm{Proj}(\Sigma_S^{-1} - \Sigma_D^{-1})$. ■

3.2 Joint Dimensionality Reduction And Metric Learning

As discussed in § 1, the metric $\boldsymbol{M}$, while scaling well to the number of (dis)similar pairs, is sensitive to the dimensionality of the space (see for example Xiong et al. (2014)). In practice, finding the optimal dimension (or equivalently the most discriminative subspace) is done through PCA. Obviously, finding the subspace along its metric is more appealing and promises better discriminatory power.

As such, our goal is to find a lower dimensional space and its Mahalanobis metric by making use of lemma 1. Formally, we seek a linear mapping $h : \mathbb{R}^D \to \mathbb{R}^d$ and an SPD matrix $\boldsymbol{M}$ such that

$$d_M^2\big(h(\boldsymbol{x}_i), h(\tilde{\boldsymbol{x}}_i)\big) = \big(h(\boldsymbol{x}_i) - h(\tilde{\boldsymbol{x}}_i)\big)^T\boldsymbol{M}\big(h(\boldsymbol{x}_i) - h(\tilde{\boldsymbol{x}}_i)\big) \quad (5)$$

reflects the dissimilarity function in Eq. (1) better. In doing so, we keep an eye on Koestinger et al. (2012) and rewrite Eq.(4) using $h(\boldsymbol{x}) = \boldsymbol{W}^T\boldsymbol{x},\ \boldsymbol{W} \in \mathbb{R}^{D\times d}$ as

$$\begin{aligned}\log\big(\delta(h(\boldsymbol{x}_i), h(\tilde{\boldsymbol{x}}_i))\big) = \frac{1}{2}\Big(&\log\det(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}) - \log\det(\boldsymbol{W}^T\Sigma_D\boldsymbol{W}) \\ &+ (\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^T\boldsymbol{W}\boldsymbol{M}\boldsymbol{W}^T(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)\Big)\ .\end{aligned} \quad (6)$$

This enables us to define the loss over all training pairs as

$$\mathcal{E}\big(\boldsymbol{W}, \boldsymbol{M}\big) \triangleq \sum_{i=1}^{N} l_i\ \log\big(\delta(h(\boldsymbol{x}_i), h(\tilde{\boldsymbol{x}}_i))\big)\ . \quad (7)$$

Minimizing Eq. (7) indeed makes the distances between similar pairs smaller while simultaneously increases the distances between dissimilar pairs. For reasons become clear soon, we opt for an alternating optimization scheme to obtain $\boldsymbol{W}$ and $\boldsymbol{M}$. That is, we keep $\boldsymbol{W}$ fixed to update $\boldsymbol{M}$, followed by updating $\boldsymbol{W}$ by keeping $\boldsymbol{M}$ fixed.

By fixing $\boldsymbol{W}$, we make use of lemma 1 to obtain a closed form update for $\boldsymbol{M}$. We note that the covariance matrix Σ in the space defined by $\boldsymbol{W}$ has the form of $\boldsymbol{W}^T\Sigma\boldsymbol{W}$. Using lemma 1, this in turn leads to the following metric in the latent space

$$\boldsymbol{M}^* = \mathrm{Proj}\Big(\big(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\big)^{-1} - \big(\boldsymbol{W}^T\Sigma_D\boldsymbol{W}\big)^{-1}\Big). \tag{8}$$

To update $\boldsymbol{W}$ while $\boldsymbol{M}$ is fixed, we add an orthogonality constraint to $\boldsymbol{W}$. The orthogonality constraint helps avoiding degeneracy in the solution and is inline with the general practice in dimensionality reduction Weinberger and Saul (2009). As such, we can write the following constrained optimization problem to update $\boldsymbol{W}$

$$\begin{array}{ll} \min_{\boldsymbol{W}} & \mathcal{E}\big(\boldsymbol{W}, \boldsymbol{M}^*\big) \\ \text{s.t.} & \boldsymbol{W}^T\boldsymbol{W} = \mathbf{I}_d \end{array} \tag{9}$$

To minimize (9), we make use of the recent advances in optimization over the matrix manifolds Absil et al. (2009). In particular, the constrained optimization problem in (9) can be understood as a minimization problem on space of tall matrices with orthogonal columns which we solve using Riemannian Conjugate Gradient Descent (RCGD) on Grassmannian.

The geometrically correct setting to minimize a problem with the orthogonality constraint is by making use of the geometry of the Stiefel manifold $\mathrm{St}(d, D) = \{\boldsymbol{W} \in \mathbb{R}^{D\times d}, \boldsymbol{W}^T\boldsymbol{W} = \mathbf{I}_d\}$. The Grassmannian manifold $\mathcal{G}(d, D)$ consists of the set of all linear d-dimensional subspaces of $\mathbb{R}^D$ and is the quotient of $\mathrm{St}(d, D)$ with the equivalence class being (see Absil et al. (2009); Edelman et al. (1998) for details)

$$[\boldsymbol{W}] \triangleq \{\boldsymbol{W}\boldsymbol{R}, \boldsymbol{W} \in \mathrm{St}(d, D), \boldsymbol{R} \in \mathcal{O}(d)\}\ ,$$

with $\mathcal{O}(d)$ denoting the orthogonal group, i.e., $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \mathbf{I}_d$.

A constrained optimization problem with the orthogonality constraint is a problem on Grassmannian if its objective is invariant to the right action of $\mathcal{O}(d)$. This is indeed the case as a result of the following theorem.

Theorem 2 *The objective defined in Eq. (7) is invariant to the right action of $\mathcal{O}(d)$, i.e., $\mathcal{E}\big(\boldsymbol{W}, \boldsymbol{M}^*\big) = \mathcal{E}\big(\boldsymbol{W}\boldsymbol{R}, \boldsymbol{M}^*\big), \boldsymbol{R} \in \mathcal{O}(d)$.*

Proof First, we show that the $\log\det(\cdot)$ terms in Eq. (6) are invariant to the action of $\mathcal{O}(d)$. Consider the first term for example. Direct insertion results in

$$\begin{aligned} \det(\boldsymbol{R}^T\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\boldsymbol{R}) &= \det(\boldsymbol{R}^T)\det(\boldsymbol{W}^T\Sigma_S\boldsymbol{W})\det(\boldsymbol{R}) \\ &= \det(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}), \end{aligned}$$

where we used the fact that $\det(\boldsymbol{R}^T) = \det(\boldsymbol{R}^{-1}) = 1/\det(\boldsymbol{R})$.

Now we show that the term with the metric is invariant to the action of $\mathcal{O}(d)$ as well. Let $\boldsymbol{A}^+ = \mathrm{Proj}(\boldsymbol{A}), \forall \boldsymbol{A} \in \mathrm{Sym}(d)$. Using SVD, it is easy to see that $\boldsymbol{R}^T\boldsymbol{A}^+\boldsymbol{R} = \mathrm{Proj}(\boldsymbol{R}^T\boldsymbol{A}\boldsymbol{R}), \forall \boldsymbol{R} \in$

$\mathcal{O}(d)$. This in turn leads to recognizing Eq. (8) by replacing $\boldsymbol{W}$ with $\boldsymbol{W}\boldsymbol{R}$ as

$$
\begin{aligned}
&\operatorname{Proj}\Big(\big(\boldsymbol{R}^T\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\boldsymbol{R}\big)^{-1} - \big(\boldsymbol{R}^T\boldsymbol{W}^T\Sigma_D\boldsymbol{W}\boldsymbol{R}\big)^{-1}\Big)\\
&= \operatorname{Proj}\Big(\boldsymbol{R}^{-1}\big(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\big)^{-1}\boldsymbol{R}^{-T} - \boldsymbol{R}^{-1}\big(\boldsymbol{W}^T\Sigma_D\boldsymbol{W}\big)^{-1}\boldsymbol{R}^{-T}\Big)\\
&= \operatorname{Proj}\Big(\boldsymbol{R}^T\Big(\big(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\big)^{-1} - \big(\boldsymbol{W}^T\Sigma_D\boldsymbol{W}\big)^{-1}\Big)\boldsymbol{R}\Big)\\
&= \boldsymbol{R}^T\operatorname{Proj}\Big(\big(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\big)^{-1} - \big(\boldsymbol{W}^T\Sigma_D\boldsymbol{W}\big)^{-1}\Big)\boldsymbol{R}\,.
\end{aligned}
$$

where we used the fact that $\boldsymbol{R}^T = \boldsymbol{R}^{-1}$.

As such and again by replacing $\boldsymbol{W}$ with $\boldsymbol{W}\boldsymbol{R}$ for the term with $\boldsymbol{M}^*$ involved (the third term in Eq. (6)), we arrive at

$$
\begin{aligned}
&(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^T\boldsymbol{W}\boldsymbol{R}\boldsymbol{M}^*\boldsymbol{R}^T\boldsymbol{W}^T(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)\\
&\qquad= (\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^T\boldsymbol{W}\boldsymbol{R}\boldsymbol{R}^T\operatorname{Proj}\Big(\big(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\big)^{-1} - \big(\boldsymbol{W}^T\Sigma_D\boldsymbol{W}\big)^{-1}\Big)\boldsymbol{R}\boldsymbol{R}^T\boldsymbol{W}^T(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)\\
&\qquad= (\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i)^T\boldsymbol{W}\operatorname{Proj}\Big(\big(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\big)^{-1} - \big(\boldsymbol{W}^T\Sigma_D\boldsymbol{W}\big)^{-1}\Big)\boldsymbol{W}^T(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i).
\end{aligned}
$$

This concludes the proof as it shows all the terms are invariant to the right action of $\mathcal{O}(d)$. ■

To perform RCGD on $\mathcal{G}(d, D)$, we need to compute the Riemannian gradient of the loss $\mathcal{E}(\boldsymbol{W}, \boldsymbol{M})$ with respect to $\boldsymbol{W}$. For a smooth function $f : \mathcal{G}(d, D) \to \mathbb{R}$ defined over the Grassmannian, the Riemannian gradient at $\boldsymbol{W}$ denoted by $\operatorname{grad}_{\boldsymbol{W}} f$ is an element of the tangent space $T_{\boldsymbol{W}}\mathcal{G}$ and is given by

$$
\operatorname{grad}_{\boldsymbol{W}} f = \big(\mathbf{I}_d - \boldsymbol{W}\boldsymbol{W}^T\big)\,\nabla_{\boldsymbol{W}} f, \tag{10}
$$

where $\nabla_{\boldsymbol{W}} f$ is a $D \times d$ matrix of partial derivatives of f with respect to the elements of $\boldsymbol{W}$, i.e., $[\nabla_{\boldsymbol{W}} f]_{i,j} = \frac{\partial f}{\partial W_{i,j}}$. Below, we derive $\nabla_{\boldsymbol{W}}\mathcal{E}(\boldsymbol{W}, \boldsymbol{M})$. For a symmetric matrix Σ

$$
\nabla_{\boldsymbol{W}} \log\det(\boldsymbol{W}^T\Sigma\boldsymbol{W}) = 2\Sigma\boldsymbol{W}\big(\boldsymbol{W}^T\Sigma\boldsymbol{W}\big)^{-1}\,.
$$

Also,

$$
\nabla_{\boldsymbol{W}}\big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)^T\boldsymbol{W}\boldsymbol{M}\boldsymbol{W}^T(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i) = 2\big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)\big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)^T\boldsymbol{W}\boldsymbol{M}.
$$

Therefore,

$$
\begin{aligned}
\nabla_{\boldsymbol{W}}\mathcal{E}\big(\boldsymbol{W}, \boldsymbol{M}\big) = \sum_{i=1}^{N} l_i\Big(&\Sigma_S\boldsymbol{W}\big(\boldsymbol{W}^T\Sigma_S\boldsymbol{W}\big)^{-1} - \Sigma_D\boldsymbol{W}\big(\boldsymbol{W}^T\Sigma_D\boldsymbol{W}\big)^{-1}\\
&+ \big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)\big(\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\big)^T\boldsymbol{W}\boldsymbol{M}\Big)\,.
\end{aligned} \tag{11}
$$

Having an alternating optimization solution not only lets us obtain $\boldsymbol{M}$ in closed form according to Eq. (8), but also justifies the use of Grassmannian, making the search space more confined. Putting everything together, the algorithm to learn $\boldsymbol{W}$ and $\boldsymbol{M}$ is depicted in Alg. 1. In our experiments, we observed that the algorithm typically converges in less than 30 iterations.

Algorithm 1 The proposed JDR-KISSME algorithm

Input: $\{\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i, l_i\}_{i=1}^N$ a set of training pairs in $\mathbb{R}^D$ with their similarity labels, the target dimensionality d
Output: Projection $\boldsymbol{W}$ and metric $\boldsymbol{M}$
1: Compute Σ_D and Σ_S using Eq. (2) and Eq. (3)
2: Initialize $\boldsymbol{W}$ to an orthonormal matrix (e.g., truncated identity)
3: Compute $\boldsymbol{M}^*$ using Eq. (8)
4: **repeat**
5: $\quad \boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \mathcal{E}\big(\boldsymbol{W}, \boldsymbol{M}^*\big)$ using RCGD on Grassmannian
6: $\quad \boldsymbol{W} \leftarrow \boldsymbol{W}^*$
7: $\quad$ Update $\boldsymbol{M}^*$ using Eq. (8)
8: $\quad \boldsymbol{M} \leftarrow \boldsymbol{M}^*$
9: **until** convergence

3.3 Incorporating the Solution into Deep Nets

In this part, we elaborate on how the previous developments can be incorporated into deep networks. The goal is to learn a mapping from images to a compact Euclidean space such that distances correspond to a notion of semantics between the images. Let us assume that a generic network provides us with an embedding from the image space to $\mathbb{R}^d$. We denote the functionality of this network on an input image x by $f(x)$.

Since a metric $\boldsymbol{M}$ is an SPD matrix, it can be decomposed as $\boldsymbol{M} = \boldsymbol{L}^T\boldsymbol{L}$. As a result, the distance between two images x_i and $\tilde{x}_i$ passing through the network can be written as

$$d_M^2\big(x_i, \tilde{x}_i\big) = \big(f(x_i) - f(\tilde{x}_i)\big)^T \boldsymbol{M}\big(f(x_i) - f(\tilde{x}_i)\big) = \|\boldsymbol{L}\big(f(x_i) - f(\tilde{x}_i)\big)\|_2^2 \,.$$

This lets us incorporate the metric $\boldsymbol{M}$ into a deep net in the form of a Fully Connected (FC) layer immediately before a loss layer. The whole setup can be trained via BackPropagation (BP). Generally speaking, training a deep CNN for metric learning is cast as one of the following forms (see Song et al. (2016) and Schroff et al. (2015) for more details).

- **Pairwise**: training data consists of pairs of similar and dissimilar images. Given a predefined margin τ, training is guided by a loss function which seeks to learn an embedding such that distances between similar samples are smaller than τ while those between dissimilar ones are greater than τ. In this manner, the cost function for a batch with N pairs $\{x_i, \tilde{x}_i\}$ and their corresponding similarity label $l_i \in \{1, -1\}$ can be written as

$$\sum_{i=1}^{N} \Big[\big(\|f(x_i) - f(\tilde{x}_i)\|_2^2 - \tau\big) l_i\Big]_+ \tag{12}$$

- **Triplewise**: training data consists of triplets of images: one anchor x, a sample in the same class x^+, and one differently labelled sample x^-, and a predefined margin τ. Then, a loss function supervises training such that for each triplet, the distance between x and x^- becomes greater than the distance between x and x^+ plus τ. Thus, the cost function for a batch with N triplets is

$$\sum_{i=1}^{N} \Big[\|f(x_i) - f(x_i^+)\|_2^2 - \|f(x_i) - f(x_i^-)\|_2^2 + \tau\Big]_+ \tag{13}$$

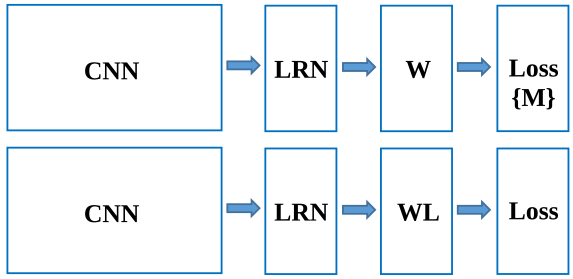

Figure 2: Incorporating JDR-KISSME into deep nets. The dimensionality reduction can be seen as an FC layer immediately before a loss layer. One can either have the metric $\boldsymbol{M}$ in computing the loss (top panel) or since $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^T$, combine it with the dimensionality reduction layer (bottom panel).

An important difference between the two categories is that only methods in the first group can work in the restricted metric learning scenario. We present our extension to deep networks utilizing the pairwise protocol, making our method applicable to wider set of problems. To this end, we start with an initial orthonormal $\boldsymbol{W}$ and compute $\boldsymbol{M}$ using Eq. (8), relying on the network to provide features in the low-dimensional space [3]. To tune $\boldsymbol{W}$ and $\boldsymbol{M}$ via BP, two possibilities are

- Initialize the weights of the last FC layer to be $\boldsymbol{W}$ and engage the metric $\boldsymbol{M}$ directly in computing the distances in the loss layer. To this end, we perform Stochastic Gradient Descent (SGD) while $\boldsymbol{M}$ is kept fixed. We update $\boldsymbol{M}$ after a number of SGD iterations or when the network reaches to a reasonably good representation. In this case, BP updates the network according to the KISSME loss (i.e., Eq. (7)) while $\boldsymbol{M}$ is learned in a closed form manner using the output of the network and Eq. (8) (see the top panel in Fig. 2 for a conceptual diagram). We refer to this solution as "pairwise+KISSME".

- Since $\boldsymbol{M}$ is an SPD matrix (i.e., $\boldsymbol{M} = \boldsymbol{L}^T\boldsymbol{L}$), it can be absorbed in the last FC layer. Here, we initialize the weights of the last FC layer to be $\boldsymbol{W}\boldsymbol{L}$. Then, we train the network using BP. If the explicit form of the metric is required, the weights of the FC layer can be factorized into an orthogonal matrix $\boldsymbol{W}$ and a full-rank matrix $\boldsymbol{L}$ using any spectral decomposition such as QR decomposition (see the bottom panel in Fig. 2 for a conceptual diagram). We refer to this solution as "pairwise+KISSME-Compact".

Empirically, we observed that pairwise+KISSME solution is more stable and works better. We conjecture that the separation of learning $\boldsymbol{W}$ and $\boldsymbol{M}$ is the reason here. In § 4.2.3, we compare the two scenarios in more details. Before concluding this part, we would like to mention that placing a Local Response Normalization (LRN) block (see Fig. 2) before the dimensionality reduction block helps the convergence in our solution. This is inline with other deep metric learning models Liu et al. (2016); Schroff et al. (2015).

3. Recent works in vision extensively use successful architectures such as AlexNet, GoogleNet or variants of VGGNet Song et al. (2016); Schroff et al. (2015); Liu et al. (2016). This strategy provides better control over the training process.

4. Experiments

In this section, we assess and contrast the performance of our proposal against the KISSME baseline and several state-of-the-art methods. We begin by evaluating JDR-KISSME using the Comprehensive Cars (CompCars) dataset Yang et al. (2015). We then demonstrate the strength of the solution when incorporated into deep networks using the CUB200-2011 Wah et al. (2011) and Cifar100 Krizhevsky (2009) datasets.

4.1 Car Verification (Shallow Experiment)

As our first experiment, we tackle the task of car verification using the recently released CompCars dataset. The dataset is one of the largest benchmarks for image verification containing 214,345 images of 1,687 car models from two significantly different scenarios: web nature and surveillance nature. Web nature data is split into three subsets without overlap. Related to verification is part II and part III of the dataset. Part II contains 4,454 images in 111 models (classes) while there are 22,236 images spanning 1,145 models in part III.

We followed Yang et al. (2015), the standard verification protocol on this dataset, which splits part III to three sets with different levels of difficulty, namely easy, medium, and hard. Each set contains 20,000 pairs including equal number of similar and dissimilar pairs. Each image in the easy pairs is chosen from the same viewpoint, while each pair in the medium pairs is selected from a random viewpoint. Each dissimilar pair in the hard subset is selected from the same car make. As for feature extraction, again following Yang et al. (2015), we utilized their available GoogLeNet Szegedy et al. (2015) fine tuned on part II of the CompCars.

On the web nature part of the CompCars, Yang et al. (2015) utilize Joint Bayesian algorithm Chen et al. (2012) which works similar in spirit to the KISSME. More specifically, the inference is based on a symmetry hypothesis comparable to Eq. (1) but with the Gaussian distributions built based on the between-class and within-class scatter matrices. Hence, it takes the class labels of the samples into its inference. Sochor et al. (2016) obtain recognition improvements by augmenting CNN inputs with complementary 3D vehicle information, such as information about 3D orientation.

In Table 1, we compare our JDR-KISSME method against the conventional KISSME algorithm and the state-of-the-art methods. First, we note that the JDR-KISSME shows consistent improvements over the KISSME. For example, the accuracy gap between the JDR-KISSME and the KISSME over the hard subset reaches 4.9%. The JDR-KISSME achieves the state-of-the-art verification accuracies on the medium and hard protocols while working on par with Yang et al. (2015) on the easy test. However, we note that the work of Yang et al. (2015) utilizes class labels for training (and hence not applicable to the restricted metric learning scenarios) while our method does not rely on the availability of such information.

Table 1: Accuracy on different protocols of the CompCars dataset.

Method	Easy	Medium	Hard
BoxCars Sochor et al. (2016)	85.0%	82.7%	76.8%
Joint Bayesian Yang et al. (2015)	**90.7%**	85.2%	78.8%
KISSME Koestinger et al. (2012)	88.9%	83.3%	75.4%
JDR-KISSME(ours)	90.1%	**86.0%**	**79.5%**

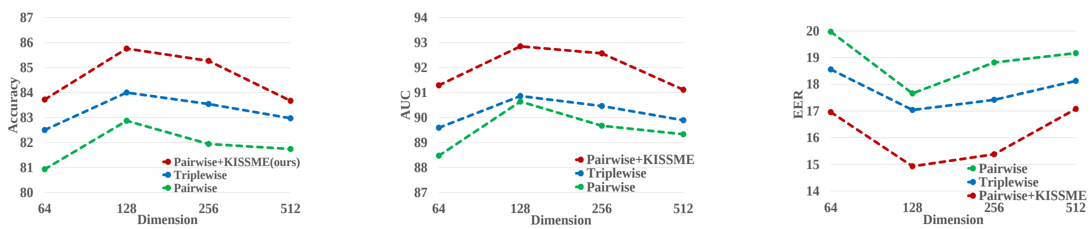

Figure 3: Verification accuracy, AUC, and EER score metrics on the CUB200-2011 dataset.

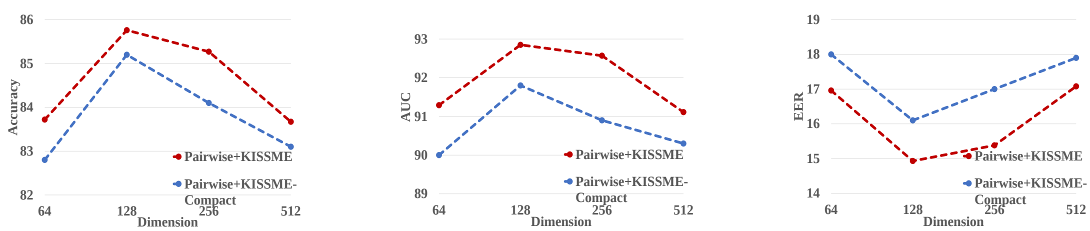

Figure 4: Comparison between the pairwise+KISSME and pairwise+KISSME-Compact deep solutions on the CUB200-2011 dataset.

4.2 Deep Experiments

In this part, we show the effectiveness of pairwise+KISSME, our deep metric learning method. To this end, we perform comparisons with the two most common ways of training a CNN for metric learning, i.e., pairwise metric learning (Eq. (12)) and triplet solution (Eq. (13)). We recall from § 3.3 that the training starts by initializing two matrices, $\boldsymbol{W}$ (or equivalently the last FC layer) and $\boldsymbol{M}$. To initialize $\boldsymbol{W}$, we rely on the initial CNN to provide embeddings of training images up to the last FC layer. Then, the FC layer is initialized with PCA (of a certain dimensionality). This is consistent with the original KISSME algorithm. Next, the metric $\boldsymbol{M}$ is initialized using the output of the FC layer (and Eq. (8)). This is required in the loss layer to perform BP.

To measure the performances, we randomly generate 30,000 similar and 30,000 dissimilar pairs from our test data. We report verification accuracy, Area Under the ROC Curve (AUC), and Equal Error Rate (EER) values for our algorithm and of the baselines for comparisons. We use Matconvnet package Vedaldi and Lenc (2015) for implementation.

Before delving into experiments, we note that our aim here is to present a better, yet general way of metric learning for deep networks. In doing so, we are not chiefly concerned about best mining practices as suggested in Song et al. (2016); Schroff et al. (2015).

Table 2: Accuracy, AUC, and EER on the Cifar100 dataset.

Method	Accuracy	AUC	EER
KISSME Koestinger et al. (2012)	63.1%	69.2%	36.4%
Pairwise	64.3%	70.0%	35.4%
Triplewise	65.8%	72.0%	34.2%
Pairwise+KISSME(ours)	**68.2%**	**75.2%**	**31.7%**

4.2.1 Bird Verification

As our first experiment for deep metric learning, we considered the task of image verification using CUB200-2011 dataset Wah et al. (2011). CUB200-2011 has 200 classes of birds with 11,788 images. We used images of the first 100 classes as training and validation sets and the remaining classes for testing. As the CNN, we utilized the VGG-CNN-M-1024 Chatfield et al. (2014) pretrained on the ImageNet Deng et al. (2009).

Fig. 3 summarizes the three score metrics of our deep metric learning technique and of our two baselines for various embedding sizes (or equivalently subspace dimensionality). Similar to the previous experiment, our solution is consistently superior to the baseline techniques for all embedding sizes. For example, the difference in the verification accuracy between our method and its closest competitor, i.e., triplet solution, is about 2% for the size 128 over the 60,000 test pairs.

4.2.2 Tiny Image Verification

As another expperiment for deep metric learning, we studied the task of image verification using the Cifar100 dataset Krizhevsky (2009) which has 60,000 images of size 32×32. To this end, we trained the LeNet-5 LeCun et al. (1998) network on the Cifar10 dataset Krizhevsky (2009) for 22,500 iterations of SGD. We then cropped the pretrained network at the fourth layer and fine tuned it on the Cifar100 similar to the previous experiment. We kept the embedding size to 32 for this experiment (i.e., $\boldsymbol{W}$ was 64×32 and $\boldsymbol{M}$ was 32×32). All other experimental details (e.g., train/test split, number of test pairs, etc) were the same as the bird verification experiment.

In Table 2, we compare our method against the so called pairwise and triplet methods as well as the original KISSME on the pretrained network (i.e., without fine tuning). Here again, our solution comfortably outperforms the other methods for all the studied metrics over the 60,000 test pairs.

4.2.3 Further Analysis

In this part, we empirically compare the pairwise+KISSME method to the pairwise+KISSME-Compact solution discussed in § 3.3. To this end, we conducted further experiments on the CUB200-2011 (bird verification) dataset. The accuracy, AUC and EER values for various embedding sizes are depicted for the two solutions in Fig. 4. From Fig. 4, we conclude that the pairwise+KISSME solution leads to superior performances and is more stable, hence our proposal. This is a consistent extension to the JDR-KISSME, our developments in the shallow mode, where we have an alternating algorithm to find the two matrices $\boldsymbol{W}$ and $\boldsymbol{M}$.

We conjecture that the separation of learning $\boldsymbol{W}$ and $\boldsymbol{M}$ is the reason. More specifically, assume the ideal projection is $\boldsymbol{W}^*$. From Lemma 1, we conclude that the ideal metric is obtained as $\boldsymbol{M}^* = \Gamma(\boldsymbol{W})$ where the function $\Gamma(\cdot)$ is a nonlinear function as a result of the projection to the positive definite cone. The pairwise+KISSME is more aligned with Lemma 1 as the metric is explicitly obtained from the representation. On the other hand, the pairwise+KISSME-Compact removes the dependency of $\boldsymbol{M}^*$ on $\boldsymbol{W}^*$ in the hope of learning $\boldsymbol{M}^*$, $\boldsymbol{W}^*$ and the underlying nonlinear projection together.

4.3 Experimental setup

For fine tuning the CNNs, we randomly generated 300,000 similar pairs and 300,000 dissimilar pairs (and equal number of triplets) and fed them to the CNNs. For all experiments, we set maximum number of SGD iterations to 20,000, margin to $\tau = 1.0$, momentum to $\mu = 0.9$, and learning rate to $\eta = [10^{-4}, 10^{-7}]$ in log-space range. We observed that increasing the learning rate of the fully connected layer by a factor of 10 helps faster convergence. A simlar observation is reported in Song et al. (2016). To augment the data, we resorted to only flipping the images at random.

We also provide details of the experiment used to generate Fig. 1. We used the surveillance images of the CompCars dataset which has 44,481 images in 281 classes. We randomly split the dataset into 140 classes for training and used the remaining 141 classes for testing. We generated 200,000 training pairs and 60,000 testing pairs, randomly from the training and testing sets. To extract image descriptors, we computed SIFT features on a dense grid and then computed Bag Of Word representations using a dictionary of size 4096, trained by the k-means algorithm.

5. Conclusions

In this paper, we introduced a joint dimensionality reduction technique for the KISSME algorithm, namely JDR-KISSME. Our motivation stems from the fact that the KISSME fails badly when its input is not meticulously denoised using PCA. Along the way, we formulated the solution as a Riemannian optimization problem. Furthermore, based on our proposal, we showed an end-to-end learning of a generic deep network for metric learning. Our experiments demonstrate consistent improvements of the JDR-KISSME and its deep extension over the original KISSME and state-of-the-art methods.

References

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. British Machine Vision Conference*, 2014.

Dong Chen, Xudong Cao, Liwei Wang, Fang Wen, and Jian Sun. Bayesian face revisited: A joint formulation. In *Proc. European Conference on Computer Vision*, pages 566–579, 2012.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 539–546, 2005.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. 20(2):303–353, 1998.

Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Conference on Learning Representations Workshops*, pages 84–92. Springer, 2015.

Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Discriminative deep metric learning for face verification in the wild. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1875–1882, 2014.

Martin Koestinger, Martin Hirzer, Paul Wohlhart, Peter M Roth, and Horst Bischof. Large scale metric learning from equivalence constraints. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2288–2295, 2012.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Hongye Liu, Yonghong Tian, Yaowei Yang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2175, 2016.

Alexis Mignon and Frédéric Jurie. Pcca: A new approach for distance learning from sparse pairwise constraints. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2666–2672, 2012.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

Jakub Sochor, Adam Herout, and Jiri Havel. Boxcars: 3d boxes as cnn input for improved fine-grained vehicle recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3006–3015, 2016.

Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation by joint identification-verification. In *Proc. Advances in Neural Information Processing Systems*, pages 1988–1996, 2014.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

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.

Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. *Proc. Advances in Neural Information Processing Systems*, 15:505–512, 2003.

Fei Xiong, Mengran Gou, Octavia Camps, and Mario Sznaier. Person re-identification using kernel-based metric learning methods. In *Proc. European Conference on Computer Vision*, pages 1–16. 2014.

Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3973–3981, 2015.