Pairwise Identity Verification via Linear Concentrative Metric Learning
Lilei Zheng, Stefan Duffner, Khalid Idrissi, Christophe Garcia, Atilla Baskurt

To cite this version:
Lilei Zheng, Stefan Duffner, Khalid Idrissi, Christophe Garcia, Atilla Baskurt. Pairwise Identity Verification via Linear Concentrative Metric Learning. IEEE Transactions on Cybernetics, IEEE, 2018, pp.1 - 12. 10.1109/TCYB.2016.2634011. hal-01435368

HAL Id: hal-01435368
https://hal.archives-ouvertes.fr/hal-01435368
Submitted on 13 Jan 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
1 Introduction

The task of pairwise identity verification is to verify whether a pair of biometric identity samples corresponds to the same person or not, where the identity samples can be face images, speech utterances or any other biometric information from individuals. Formally, in such pairwise verification problems, two identity samples of the same person are called a similar pair, and two samples of two different persons are called a dissimilar pair or a different pair.

Compared with the traditional identity classification task in which a decision of acceptance or rejection is made by comparing an identity sample to models (or templates) of each individual [1], [2], [3], pairwise identity verification is more challenging because of the impossibility of building robust identity models with enough training data [4] for all the individuals. Actually, there may be only one identity sample available for some individuals in pairwise identity verification. Besides, individuals in training and testing should be mutually exclusive, i.e. the testing set comprises only samples from unknown persons that are not part of the training set.

Face images or speech utterances may be the most accessible and widely used identity information. As a result, face verification [1] and speaker verification [2] has been well studied over the last two decades. Especially, pairwise face verification has drawn much attention in recent years thanks to the popularity of the dataset ‘Labeled Faces in the Wild’ (LFW) [4]. Originally, the LFW dataset proposed a restricted training protocol where only a few specified data pairs are allowed for training, a challenging setting for effective learning algorithms to discover principles from a small number of training examples, just like the human beings [5]. On the other hand, in the NIST Speaker Recognition Evaluations (SREs) since 1996, various speaker verification protocols have been investigated [6], [7]. In order to follow the pair generation scheme in the LFW standard protocol, we establish the pairwise speaker verification protocol based on the data from the NIST 2014 i-Vector Machine Learning Challenge [7].

The definition of pairwise identity verification reveals the need of measuring the difference or similarity between a pair of samples, which naturally leads us to the study of metric learning [8], i.e. methods that automatically learn a metric from a set of data pairs. A metric learning framework is implemented with a siamese architecture [9] which consists of two identical sub-systems sharing the same set of parameters. For a given input data pair, the two samples are processed by the two sub-systems respectively. The overall system includes a cost function parameterizing the pairwise relationship between data and a mapping function allowing the system to learn high-level features from the training data.

In terms of the cost function, one can divide metric learning methods into distance metric learning and similarity metric learning, where the cost function is defined based on a distance metric and a similarity measurement, respectively. The objective of such a cost function is to increase the similarity value or to decrease the distance between a similar pair, and to reduce the similarity value or to increase the distance between two dissimilar data samples. In this paper, we investigate two kinds of metric learning methods, namely, Triangular Similarity Metric Learning (TSML) [10] and Discriminative Distance Metric Learning (DDML) [11].

In terms of the mapping function, one can divide metric learning methods into two main families: linear metric learning and nonlinear metric learning. Up to now, work in metric learning has focused on linear methods because
they are more convenient to optimize and less prone to over-fitting. For instance, the best approaches such as the Within Class Covariance Normalization (WCCN) and Cosine Similarity Metric Learning (CSML), have shown their effectiveness on the problem of pairwise face verification [12], [13]. Also, a few approaches have investigated nonlinear metric learning and have shown competitive performance on some classification problems [11], [14], [15]. Moreover, comparing linear systems with their nonlinear variants on a common ground helps to study the effect of nonlinearity on pairwise verification. For example, the nonlinear transformation – Diffusion Maps (DM) – has been introduced to face verification [13] and speaker verification [16], respectively. However, no clear evidence in the comparisons validated the universal effectiveness of DM over the linear systems [13]. Analogously, we present the TSML and DDML methods in both linear and nonlinear formulations for the sake of a thorough evaluation. Note that the nonlinear formulations are developed on the linear ones by adding nonlinear activation functions or stacking one more layer of transformation, thus the implemented nonlinearity is shallow.

Overall, on the problem of pairwise identity verification via metric learning, this paper presents a comprehensive study including two kinds of verification applications (i.e. face verification and speaker verification), two kinds of training settings (i.e. data-restricted and data-unrestricted), two kinds of metric learning cost functions (i.e. TSML and DDML), and three kinds of mapping functions (i.e. linear function, single-layer nonlinear function and multi-layer nonlinear function).

We will show that under the setting of limited training data, a linear metric learning system trained on similar pairs only generally yields competitive verification results. Either linear TSML or linear DDML achieves the state-of-the-art performance on both the LFW image dataset and the NIST speaker dataset.

The contributions of this paper with respect to previous works are the following:

- we establish a pairwise speaker verification protocol based on the data from the NIST 2014 i-Vector machine learning challenge, which has mutually exclusive training and test sets of speakers. Both the pairwise face verification protocol of the LFW dataset and this speaker verification task aim at verifying identity information by individuals’ biometric features. Another objective of using the two datasets is to show the effectiveness of the proposed metric learning systems on different kinds of data, i.e. images and speech.
- we present the TSML and DDML methods in both linear and nonlinear formulations for pairwise identity verification problems. A thorough evaluation comparing the different formulations has shown that with limited training data, the linear models are preferable due to its superior performance and its simplicity.
- we study the influence of limited training data. Generally, compared with unlimited training, the limited case suffers from over-fitting. However, we find that training the linear models on similar pairs only considerably reduces the effect of over-fitting to limited training data.
- we also integrate the proposed linear and shallow nonlinear metric learning models with a pre-trained deep Convolutional Neural Network (CNN) model to improve the performance of pairwise face verification. We find that the linear model serves as an effective verification layer stacked to the deep CNN.

The remainder of this paper is organized as follows: Section 2 briefly summarizes the related work on metric learning and feature representations for images and speech. Section 3 presents the objective of metric learning by illustrating the cost functions of TSML and DDML. Section 4 introduces the linear and nonlinear formulations and explains the details of our stochastic gradient descent algorithm for optimization. Section 5 describes the datasets and experiments for pairwise face verification and pairwise speaker verification. Finally, we draw our conclusions in Section 6.

2 RELATED WORK

2.1 Metric Learning and Siamese Neural Networks

Most of linear metric learning methods employ two types of metrics: the Mahalanobis distance or a more general similarity metric. In both of the two cases, a linear transformation matrix \( W \) is learnt to project input features into a target space. Typically, distance metric learning concerns the Mahalanobis distance [17], [18]:

\[
d_W(x, y) = \sqrt{(x - y)^T W (x - y)},
\]

where \( x \) and \( y \) are two sample vectors, and \( W \) is the matrix that needs to be learnt. Note that when \( W \) is the identity matrix, \( d_W(x, y) \) is the Euclidean distance. In contrast, similarity metric learning methods learn a function of the following form:

\[
s_W(x, y) = x^T W y / N(x, y),
\]

where \( N(x, y) \) is a normalization term [19]. Specifically, when \( N(x, y) = 1 \), \( s_W(x, y) \) is the bilinear similarity function [20]; when \( N(x, y) = \sqrt{x^T W x} \sqrt{y^T W y}, s_W(x, y) \) is the generalized cosine similarity function [12].

Nonlinear metric learning methods are constructed by simply substituting the above linear projection with a nonlinear transformation [11], [14], [15], [21]. For example, [11] and [14] employed neural networks to accomplish the nonlinear transformation. These nonlinear methods are subject to local optima and more inclined to over-fit to the training data but have the potential to outperform linear methods on some problems [8], [15]. Compared with linear models, nonlinear models are usually preferred on a redundant training set to well capture the underlying distribution of the data [22].

Since neural networks are the most commonly used nonlinear models, nonlinear metric learning has a natural connection with siamese neural networks [9], [14]. Actually, siamese neural networks can also be linear if the neurons have a linear activation function. From this point of view, siamese neural networks and metric learning denote the same technique of optimizing a metric-based cost function via a linear or nonlinear mapping. The difference exists in their names: “siamese neural networks” concern the symmetric structure of neural networks used for data mapping.
but the term “metric learning” emphasizes the pairwise relationship (i.e. the metric) in the data space.

For readers interested in a broader scope on metric learning in the literature, we recommend a recent survey which has provided an up-to-date and critical review of existing metric learning methods [8]. For those who prefer experimental analysis, an overview and empirical comparison is given in [23].

2.2 Feature Representation for Face and Speaker

For face recognition, tremendous efforts have been put on developing robust face descriptors [13], [24], [25], [26], [27], [28], [29], [30], [31], [32]. Popular face descriptors include eigenfaces [24], Gabor wavelets [27], SIFT [26], Local Binary Patterns (LBP) [25], etc. Especially, LBP and its variants, such as center-symmetric LBP (CSLBP) [33], multi-block LBP (ML-LBP) [34], three patch LBP (TPLBP) [28] and over-complete LBP (OCLBP) [13], have been proven to be effective at describing facial texture. Especially, the high-dimensional variants usually perform better, for example, OCLBP [13]. Recently, another high-dimensional candidate, Fisher Vector (FV) face, which combines dense feature sampling with improved Fisher Vector encoding, has achieved striking results on pairwise face verification [30]. Besides, compared with the above handcrafted descriptors, automatical feature learning using Convolutional Neural Networks (CNN) has attracted a lot of interest in Computer Vision during the past decade [35], [36], [37]. In contrast to the handcrafted features, these CNN-based approaches usually rely on large training data to learn a lot of parameters, but they have substantially raised the state-of-the-art records on almost all the challenges in Computer Vision [38].

For speaker recognition, the most popular features are developed on generative models such as Gaussian Mixture Model-Universal Background Model (GMM-UBM) [39]. Building on the success of GMM-UBM, Joint Factor Analysis (JFA) proposes powerful tools to model the inter-speaker variability and to compensate for channel/session variability in the context of GMMs [40]. Moreover, inspired by JFA, a new feature called i-vector is developed [41], [42], [43]. JFA models the speaker variability in the high-dimensional space of GMM supervectors, whereas i-vectors are extracted in a low-dimensional space named total variability space. Taking advantage of the low dimensionality of the total variability space, many machine learning techniques can be applied to speaker verification [44]. Probabilistic Linear Discriminant Analysis (PLDA) [45] is one of the most popular techniques used for speaker verification: different variants such as the Gaussian PLDA (G-PLDA) [16], [46], Heavy-Tailed PLDA (HT-PLDA) [47], [48], [49] and Nonlinear PLDA [50] have been studied. In addition, Pairwise Support Vector Machines (PSVM) [51], [52] have been proposed to verify utterance pairs of different speakers; and fusing PSVM with PLDA can further improve the verification performance [46]. Recently, the metric learning framework DDML [11] was also shown to be helpful for PLDA-based speaker verification [50].

In our experiments, instead of studying the CNN for face verification or the PLDA for speaker verification, we focus on investigating the same metric learning models on the two verification tasks. In terms of feature representations, we choose Fisher Vector faces as the face descriptors and i-vectors as the speech utterance descriptors.

3 Metric Learning Objectives

Metric learning algorithms usually employ the siamese architecture [9] to compare a pair of data inputs. Figure 1 shows the principal approach. A pair of data is given at the input, and two outputs are produced respectively with the current mapping function \( f(\cdot) \). These outputs are constrained by a metric-based cost function \( J(\cdot) \). By minimizing this cost function, we can achieve the objective of attracting similar pairs and separating dissimilar pairs. Concretely, if the pair of inputs are similar (i.e. from the same individual), the objective is to make the outputs more similar than the inputs; otherwise, the objective is to make the outputs more dissimilar/different. Popular choices of the measurement on the output vectors include the Euclidean distance [11], [18] and the Cosine Similarity [12], [20]. Therefore, we apply a distance metric learning method DDML [11] and a similarity metric learning method TSML [10] for the problem of pairwise identity verification.

By representing the face images or speech utterances as numerical vectors, we use a triplet \((x_i, y_i, s_i)\) to represent a pair of training input instances, where \(x_i\) and \(y_i\) are two vectors, and \(s_i = 1\) (respectively \(s_i = -1\)) means that the two vectors are similar (respectively dissimilar). Taking a projection \(f(z, W)\) on the inputs, we obtain a new pair \((a_i, b_i)\) in the target space, where \(a_i = f(x_i, W)\) and \(b_i = f(y_i, W)\). Then, the TSML or DDML cost function is constructed to define the pairwise relationship between \(a_i\) and \(b_i\). Finally, the procedure of learning the metric is carried out by minimizing the cost on a set of training pairs.

3.1 Triangular Similarity Metric Learning

TSML concerns the Triangular Similarity which is equivalent to the Cosine Similarity [53]. On the two outputs \(a_i\) and
In contrast, DDML focuses on the pairwise distance between similar pairs and separating dissimilar pairs.

To opposite unit vectors along the other diagonal (the blue vectors opposite). (b) The gradient function suggests unit vectors on the diagonals as targets for \( a_i \) and \( b_i \); the same target vector for a similar pair \( (s_i = 1) \); or the opposite target vectors for a dissimilar pair \( (s_i = -1) \).

\[ c_i = a_i + s_i b_i = a_i + b_i \]
\[ a_i, \theta \]
\[ s_i = 1 \]
\[ s_i = -1 \]

\[ J_i = \frac{1}{2} ||a_i||^2 + \frac{1}{2} ||b_i||^2 - ||c_i|| + 1, \]  
(1)

where \( c_i = a_i + s_i b_i \); \( c_i \) can be regarded as one of the two diagonals of the parallelogram formed by \( a_i \) and \( b_i \) (Fig. 2(a)). Moreover, this cost function can be rewritten as:

\[ J_i = \frac{1}{2} (||a_i|| - 1)^2 + \frac{1}{2} (||b_i|| - 1)^2 + ||a_i|| + ||b_i|| - ||c_i||. \]  
(2)

We can see that minimizing the first part aims to make the vectors \( a_i \) and \( b_i \) having unit length 1; the second part concerns the well-known triangle inequality theorem: the sum of the lengths of two sides of a triangle must always be greater than the length of the third side, i.e. \( ||a_i|| + ||b_i|| - ||c_i|| > 0 \). More interestingly, with the length constraints by the first part, minimizing the second part is equivalent to minimizing the angle \( \theta \) inside a similar pair \( (s_i = 1) \) or maximizing the angle \( \theta \) inside a dissimilar pair \( (s_i = -1) \), in other words, minimizing the Cosine Similarity between \( a_i \) and \( s_i b_i \):

\[ \cos(a_i, s_i b_i) = s_i \frac{a_i^T b_i}{||a_i|| ||b_i||}. \]  
(3)

The gradient of the cost function (Equation (1)) with respect to the parameters \( W \) is:

\[ \frac{\partial J_i}{\partial W} = \left( a_i - \frac{c_i}{||c_i||} \right)^T \frac{\partial a_i}{\partial W} + \left( b_i - \frac{s_i c_i}{||c_i||} \right)^T \frac{\partial b_i}{\partial W}. \]  
(4)

We can obtain the optimal cost at the zero gradient: \( a_i - \frac{c_i}{||c_i||} = 0 \) and \( b_i - \frac{s_i c_i}{||c_i||} = 0 \). In other words, the gradient function has \( \frac{c_i}{||c_i||} \) and \( \frac{s_i c_i}{||c_i||} \) as targets for \( a_i \) and \( b_i \), respectively. Fig. 2(b) illustrates that: for a similar pair, \( a_i \) and \( b_i \) are mapped to the same target vector along the diagonal (the red solid line); for a dissimilar pair, \( a_i \) and \( b_i \) are mapped to opposite unit vectors along the other diagonal (the blue solid line). This perfectly reveals the objective of attracting similar pairs and separating dissimilar pairs.

### 3.2 Discriminative Distance Metric Learning

In contrast, DDML focuses on the pairwise distance between feature vectors. Unlike the Cosine Similarity naturally defines a minimum of -1 and a maximum of 1, the Euclidean distance has only a minimum of 0 and no maximum. Hence a margin is usually defined in distance metric learning to assume that two vectors with a distance larger than the margin are well separated.

Typically, for a pair of outputs \( a_i \) and \( b_i \), DDML defines the cost function as:

\[ J_i = \frac{1}{2} g(1 - s_i(1 - (a_i - b_i)^2)), \]  
(5)

where \( g(z) = \frac{1}{2} log(1 + exp(Tz)) \) is the generalized logistic loss function [54], \( T \) is a sharpness parameter usually set to 10. Minimizing the logistic loss function means to minimize the value of

\[ z_i = 1 - s_i(1 - (a_i - b_i)^2). \]  
(6)

Specifically, for a similar pair \( (s_i = 1) \), \( z_i \) can be simplified as \( (a_i - b_i)^2 \), and minimizing \( z_i \) requires \( a_i \) and \( b_i \) to be identical; for a dissimilar pair \( (s_i = -1) \), the equation suggests maximizing \( -z_i = (a_i - b_i)^2 - 2 \), that is to separate a dissimilar pair with a distance of \( \sqrt{2} \). An illustration of the objective is shown in Fig. 3.

The gradient of the DDML cost function (Equation (5)) with respect to the parameters \( W \) is:

\[ \frac{\partial J_i}{\partial W} = \frac{s_i(a_i - b_i)}{1 + exp(-T(1 - s_i + s_i(a_i - b_i)^2))} \frac{\partial(a_i - b_i)}{\partial W}. \]  
(7)

### 3.3 Cost and Gradient for Batch Training

In practice, we may consider a few data pairs as a small batch in each training iteration, thus the overall cost and
The gradient of a batch is simply the average from all the training pairs in the batch:

\[
J = \frac{1}{n} \sum_{i=1}^{n} J_i, \tag{8a}
\]

\[
\frac{\partial J}{\partial W} = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial J_i}{\partial W}, \tag{8b}
\]

where \(n\) is the number of training pairs in a batch, \(J_i\) is the TSML cost in Equation (1) or the DDML cost in Equation (5), the corresponding gradient \(\frac{\partial J_i}{\partial W}\) is calculated by Equation (4) or Equation (7). Finally, the gradient can be used in the Backpropagation algorithm [55] to perform gradient descent and search an optimal solution.

4 Linear and Nonlinear Mappings

When a cost function defines the pairwise relationship between data in the target space, a mapping function represents the system’s ability of learning to achieve the goal of the cost function. From the point of view of neural networks, different mapping functions can be considered as different combinations of neurons in network layers. We study three kinds of mapping functions here:

Single layer of linear neurons

The simplest neurons are the linear neurons without bias term which only involve a parameter matrix \(W\). For a given input \(z \in \mathbb{R}^d\), the output is simply \(f(z, W) = Wz\). For instance, the TSML gradient of the \(i_{th}\) pair with respect to the parameter matrix \(W\) is:

\[
\frac{\partial J_i}{\partial W} = (a_i - \frac{c_i}{\|c_i\|}x_i^T) + (b_i - s_i \frac{c_i}{\|c_i\|})y_i^T. \tag{9}
\]

Single layer of nonlinear neurons

Besides the parameter matrix \(W\), nonlinear neurons involve a bias term, and a nonlinear activation function, e.g. the tanh function [22]. For a given input \(z \in \mathbb{R}^d\), the output is:

\[
f(z, W) = \tanh(Wz + h), \tag{10}
\]

where \(h\) denotes the bias term of the neurons. This equation can be rewritten as:

\[
f(z', W') = \tanh(W'z'), \tag{11}
\]

where \(z' = [z; 1]\) and \(W' = [W; h]\). Remind that derivative of the tanh function is \(\tanh'(z) = 1 - \tanh^2(z)\). Based on the linear case in Equation (9), the derivative of the TSML cost function with respect to the parameters \(W' : \{W, h\}\) is:

\[
\frac{\partial J_i}{\partial W'} = (a_i - \frac{c_i}{\|c_i\|}) \frac{\partial a_i}{\partial W'} + (b_i - s_i \frac{c_i}{\|c_i\|}) \frac{\partial b_i}{\partial W'}
\]

\[
= \left[ (1 - a_i \odot a_i) \odot (a_i - \frac{c_i}{\|c_i\|}) \right] [x_i; 1]^T \tag{12}
\]

\[
+ (1 - b_i \odot b_i) \odot (b_i - s_i \frac{c_i}{\|c_i\|})[y_i; 1]^T,
\]

where the notation \(\odot\) means element-wise multiplication. The derivation of this equation can be easily obtained with the chain rule used in the Backpropagation algorithm [22].

Multiple layers of nonlinear neurons

By combining several interconnected nonlinear neurons together, Multi-Layer Perceptrons (MLP) are able to approximate arbitrary nonlinear mappings and thus have been the most popular kind of neural networks since the 1980’s [55]. We adopt a 3-layer MLP, containing one input layer and two layers of nonlinear neurons, to realize the nonlinear mapping. Similar with Equation (12) and according to the Backpropagation chain rule, we can calculate derivatives with respect to each parameter of the MLP for a given training pair.

For the DDML cost function, we can obtain derivatives with respect to the weights of neuron layers in the same way with the TSML method. For all the linear and shallow nonlinear systems, we employ the same stochastic gradient descent optimization to update their weights until reaching an optimal solution.

4.1 Stochastic Gradient Descent

Since all the three types of mapping functions have similar cost and gradient functions, we employ the same algorithm to perform optimization. The proposed method is based on stochastic gradient descent and is summarized in Algorithm 1. More advanced optimization algorithms such as conjugate gradient descent, L-BFGS [10], [56] could be used as well but their analysis would go beyond the scope of this paper. We adopt early-stopping [57] to prevent the over-fitting problem. Thus a small set is separated from the training data for validation, and the model with the best performance on the validation set is retained for evaluation on the test set. In addition, we use a momentum [22] term to speed up training. The momentum \(\lambda\) is empirically set to be 0.99 for all the experiments. Following [30], [58], the input vectors will be passed through L2 normalization before training, i.e. the length of input vectors is normalized to 1.

Initializing the weights

For the linear mapping, like in [10], [12], [58], we initialize the transformation matrix with the identity matrix. For the nonlinear mappings, we use the normalized random initialization [59] that is considered to be helpful for the tanh networks. Concretely, weights of each layer are initialized with an uniform distribution as:

\[
\{W^{(j)}, h^{(j)}\} \sim U[-\frac{\sqrt{6}}{\sqrt{n_j + n_j+1}}, \frac{\sqrt{6}}{\sqrt{n_j + n_j+1}}], \tag{13}
\]

where \(\{W^{(j)}, h^{(j)}\}\) denotes the parameters between the \(j_{th}\) and \((j + 1)_{th}\) layers; \(n_j\) and \(n_j+1\) represent the number of nodes in the two layers, respectively.

5 Experiments and Analysis

5.1 Datasets

In order to validate the generality of the proposed approaches, we carried out pairwise identity verification experiments on two datasets in different domains: the LFW image dataset for pairwise face verification [4] and the NIST i-vector dataset for pairwise speaker verification [7].
Distribution of individuals and images in the 10 subsets, where the individuals are mutually exclusive.

<table>
<thead>
<tr>
<th>Index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of individuals</td>
<td>601</td>
<td>555</td>
<td>552</td>
<td>560</td>
<td>567</td>
<td>527</td>
<td>597</td>
<td>601</td>
<td>580</td>
<td>609</td>
<td>5749</td>
</tr>
<tr>
<td>Number of images</td>
<td>1369</td>
<td>1367</td>
<td>1089</td>
<td>1324</td>
<td>1016</td>
<td>1166</td>
<td>1690</td>
<td>1222</td>
<td>1207</td>
<td>1783</td>
<td>13233</td>
</tr>
</tbody>
</table>

Distribution of individuals and speech utterances in the 10 subsets, where the individuals are mutually exclusive.

<table>
<thead>
<tr>
<th>Index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of individuals</td>
<td>496</td>
<td>496</td>
<td>496</td>
<td>496</td>
<td>496</td>
<td>496</td>
<td>496</td>
<td>494</td>
<td>494</td>
<td>4958</td>
<td></td>
</tr>
<tr>
<td>Number of utterances</td>
<td>3660</td>
<td>3664</td>
<td>3568</td>
<td>3741</td>
<td>3702</td>
<td>3566</td>
<td>3605</td>
<td>3636</td>
<td>3744</td>
<td>3686</td>
<td>36522</td>
</tr>
</tbody>
</table>

Algorithm 1: Stochastic Gradient Descent for TSML

| input : Training set; Validation set;  
output: Parameter set \( W \) |
| parameters: Learning rate \( \alpha = 10^{-4} \); Momentum \( \lambda = 0.99 \); Iterative tolerance \( P_t = 4 \times 10^5 \); Validation frequency \( F_t = 10^3 \); |
| % initialization |
| if linear mapping then |
| \( W_0 \leftarrow I \);  
% I is the identity matrix |
| if nonlinear mapping then |
| randomly initialize \( W_0 \) according to Equation (13);  
\( \Delta W_0 \leftarrow 0 \);  
Perform L2 normalization on the training set;  
Perform L2 normalization on the validation set;  
% optimization by Backpropagation |
| for \( t = 1, 2, \ldots, P_t \) do |
| % select training data for each epoch |
| Randomly select a similar pair and a dissimilar pair from the training set;  
% forward propagation |
| Calculate \( J \) on the selected training pairs;  
% backpropagation |
| Calculate the corresponding gradient \( \frac{\partial J}{\partial W_{t-1}} \);  
% updating using momentum |
| \( \Delta W_t = \lambda \Delta W_{t-1} + \frac{\partial J}{\partial W_{t-1}} \);  
\( W_t \leftarrow W_{t-1} + \alpha \Delta W_t \);  
% checking on the validation set regularly |
| if \( (P_t \mod F_t) == 0 \) then |
| compute the Decision Accuracy according to Equation (14);  
% output the best matrix on the validation set |
| \( W_\star \leftarrow W_t \) gives the best result on the validation set;  
return \( W_\star \). |

5.1.1 LFW dataset

The LFW dataset contains numerous annotated images from the web. For all the images, we used the cropped 150 × 150 ‘funneled’ version of LFW [4]. We only used the View 2 subset of LFW for performance evaluation. In View 2, to do 10-fold cross validation, all the 5749 persons in the dataset are divided into 10 subsets where the individuals are mutually exclusive. The total number of images for all the persons is 13,233, however, the number of images for each individual varies from 1 to 530. Table 1 summarizes the data distribution of individuals and images in the 10 subsets.

We used Fisher Vector faces as vector representation of face images, where data of the vectors are directly provided by [30]1 (Data for the setting 3), and the dimension of a Fisher Vector face is 67,584. However, directly taking the original facial vectors for learning causes computational problems, i.e. the time required for multiplications of the 67,584-d vectors would be unacceptable. Therefore, following [12], [13], we apply Whitened Principal Component Analysis (WPCA) to reduce the vector dimension to 500.

5.1.2 NIST i-vector dataset

We used the data of the NIST 2014 Speaker i-Vector Challenge [7], which consist of i-vectors derived from conversational telephone speech data in the NIST speaker recognition evaluations from 2004 to 2012. Each i-vector, the identity vector, is a vector of 600 components. Along with each i-vector, the amount of speech (in seconds) used to compute the i-vector is supplied as metadata. Segment durations were sampled from a log normal distribution with a mean of 39.58 seconds. This dataset consists of a development set for building models and a test set for evaluation.

We only used the development data of this Challenge and established an experimental protocol of pairwise speaker verification. There are 36,572 speech utterances in total in this experiment, belonging to 4958 different speakers. The number of utterances for a single speaker varies from 1 to 75. Like in LFW, we also split the data into 10 subsets to do 10-fold cross validation. Table 2 shows the distribution of individuals and speech utterances in the 10 subsets.

5.2 Experimental Setup

On both of the two datasets, we performed cross-validation on the 10 folds: there are overall 10 experiments, in each repetition, sample pairs from 9 folds are used for training, and sample pairs from the remaining fold are used for testing. As we have announced in Section 4.1, some training data are separated as an independent validation set to do early-stopping.

5.2.1 Fixed testing

To perform evaluation on the test set for each experiment, it is better to fix the sample pairs in each fold so that we can fairly compare different approaches on the same test data. Specifically, 600 image pairs are provided in each fold of the

1. http://www.robots.ox.ac.uk/~vgg/software/face_desc/
LFW dataset, where 300 are similar and the other 300 are dissimilar [4]. In the NIST i-vector dataset, there are more samples for each individual than in the LFW dataset, so we generate more sample pairs for each fold, namely, 1200 similar pairs and 1200 dissimilar pairs.

5.2.2 Restricted and unrestricted training
Following [4], we defined two training settings in our experiments: the restricted setting in which only the fixed sample pairs in each fold can be collected for training, e.g. the specified 300 similar and 300 dissimilar pairs in each fold of the LFW dataset; in contrast, the unrestricted setting allows to generate more sample pairs for training by using the identity information of all the samples. As mentioned previously, the test sample pairs are the same for both restricted and unrestricted settings.

5.2.3 Maximal decision accuracy
Like the minimal Decision Cost Function (minDCF) in [7], we define a Decision Accuracy (DA) function to measure the overall verification performance on a set of data pairs:

$$DA(\gamma) = \frac{\text{number of right decisions}(\gamma)}{\text{total number of pairs}}$$

where the threshold $\gamma$ is used to make a decision on the final distance or similarity values: for the TSML system, $cos(a, b) > \gamma$ means $(a, b)$ is a similar pair, otherwise it is dissimilar; for the DDML system, $(a - b)^2 < \gamma$ denotes a similar pair, otherwise it is dissimilar. The maximal DA (maxDA) over all possible threshold values is the final score recorded. We report the mean maxDA scores (±standard error of the mean) of the 10 experiments. For the speaker verification results, we also measure the mean Equal Error Rate (EER) as it is commonly used in the speaker recognition field [47], [52].

5.3 Experimental Results
At the beginning, we directly calculated maxDA scores on the whitened feature vectors, i.e. the 500-dimensional FV vectors for the LFW dataset and 600-dimensional i-vectors for the NIST i-vector dataset. We consider this evaluation as the baseline. According to the different neuron models defined in Section 4, we evaluated three kinds of metric learning approaches in the experiments:

- TSML-Linear and DDML-Linear: using a single layer of linear neurons without bias term;
- TSML-Nonlinear and DDML-Nonlinear: using a single layer of nonlinear neurons with a bias term;
- TSML-MLP and DDML-MLP: using two layers of nonlinear neurons with bias terms;

All these models are trained on both similar and dissimilar pairs. Results on the LFW-funnelled dataset and the NIST i-vector dataset are summarized in Tables 3 – 6. We also re-implement the state-of-the-art WCCN method [13], [58] as a comparison.

Learning on Similar Pairs Only: comparing WCCN with the proposed six metric learning models, we find that WCCN achieves better performance under the restricted training. The major difference between WCCN and the other models is that WCCN concerns only intra-personal variance but ignores the inter-personal information [13], [58]. In other words, WCCN performs learning on similar pairs only but the current TSML and DSML systems take into account both similar and dissimilar pairs. To clarify this issue, we train the proposed models on similar pairs only as six new models: TSML-Linear-Sim, TSML-Nonlinear-Sim and TSML-MLP-Sim; DDML-Linear-Sim, DDML-Nonlinear-Sim and DDML-MLP-Sim. The results are also shown in Tables 3 – 6.

5.3.1 More training data
The first phenomenon we can observe is that unrestricted training produces better results than restricted training. More training data generally bring up an accuracy improvement to each model. We have known since mid-seventies [5], [38], [60] that many methods increase in accuracy with increasing training data until they reach optimal performance. Indeed, more training data better capture the underlying distribution of the whole dataset and thus reduce the overfitting gap between training and test. Especially for the pairwise verification problem that requires learning on data pairs, compared with restricted training only allows to use a few specified training pairs in a dataset, unrestricted training covers enough data pairs and thus protect the models from over-fitting to a small portion of training data.

5.3.2 Linear vs. nonlinear
The second observation is that the linear models generally perform better than the shallow nonlinear models. Specifically, more parameters (i.e. additional bias terms or/and more layers of neurons) and the nonlinearity make the
nonlinear models more powerful to adapt themselves to the training data. However, without any additional techniques to prevent over-fitting, generalization to the test data is not guaranteed. Figure 4 shows the learning curves of TSML-Linear, TSML-Nonlinear and TSML-MLP in restricted training, we can see that all of them easily fit the training data. Especially, with the most parameters, TSML-MLP is the strongest learning machine that reaches the accuracy of 100% on the training data with the fewest iterations, but it performs the worst on the test data. More regularization techniques, such as weight decay [22] and dropout [61], can be introduced to reduce the risk of over-fitting for such slightly deeper nonlinear model, but their analysis would go beyond the scope of this paper. In contrast, with the same experimental setting, linearity naturally indicates the property of generalization and thus makes TSML-Linear better fit to the unseen data, i.e. the validation and test sets.

5.3.3 Concentrative training on limited data pairs

Figure 5 compares the performance of the linear models of both TSML and DDML on the LFW-funneled dataset and the NIST i-vector dataset, respectively. In general, under the restricted training, the models trained on similar pairs only, i.e. TSML-Linear-Sim and DDML-Linear-Sim, yield significantly better results; under the unrestricted training, all the linear models perform comparably well.

In general, a linear concentrative model\(^2\) should be adopted for restricted training because of its superior performance. Moreover, it should be also preferred for unrestricted training due to faster training. Compared with models trained on both similar and dissimilar pairs, the linear concentrative models only take into account half of the training data but yield comparable verification results.

Concretely, the setting of equal quantity of similar and dissimilar pairs is problematic for restricted training. As-

### Table 5
Mean maxDA scores (± standard error of the mean) of pairwise face verification by the DDML systems on the LFW-funneled image dataset. ‘-Sim’ means learning on similar pairs only.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Restricted Training</th>
<th>Unrestricted Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>84.83±0.38</td>
<td>91.17±0.36</td>
</tr>
<tr>
<td>WCCN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDML-Linear</td>
<td>91.10±0.45</td>
<td>92.48±0.35</td>
</tr>
<tr>
<td>DDML-Nonlinear</td>
<td>91.03±0.61</td>
<td>91.80±0.29</td>
</tr>
<tr>
<td>DDML-MLP</td>
<td>90.82±0.45</td>
<td>91.42±0.40</td>
</tr>
<tr>
<td>DDML-Linear-Sim</td>
<td>89.57±0.45</td>
<td>89.53±0.44</td>
</tr>
<tr>
<td>DDML-Nonlinear-Sim</td>
<td>91.05±0.27</td>
<td>90.39±0.25 / 0.0807</td>
</tr>
<tr>
<td>DDML-MLP-Sim</td>
<td>91.11±0.27</td>
<td>91.42±0.25 / 0.0807</td>
</tr>
</tbody>
</table>

### Table 6
Mean maxDA scores (± standard error of the mean) and mean EER of pairwise speaker verification by the DDML systems on the NIST i-vector dataset. ‘-Sim’ means learning on similar pairs only.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Restricted Training</th>
<th>Unrestricted Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>87.78±0.39 / 0.1335</td>
<td>91.97±0.33 / 0.0853</td>
</tr>
<tr>
<td>WCCN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDML-Linear</td>
<td>89.77±0.21 / 0.1127</td>
<td>94.32±0.23 / 0.0612</td>
</tr>
<tr>
<td>DDML-Nonlinear</td>
<td>87.98±0.29 / 0.1262</td>
<td>93.36±0.23 / 0.0703</td>
</tr>
<tr>
<td>DDML-MLP</td>
<td>89.11±0.27 / 0.1143</td>
<td>92.39±0.25 / 0.0807</td>
</tr>
<tr>
<td>DDML-Linear-Sim</td>
<td>92.94±0.20 / 0.0748</td>
<td>94.42±0.24 / 0.0590</td>
</tr>
<tr>
<td>DDML-Nonlinear-Sim</td>
<td>91.98±0.25 / 0.0850</td>
<td>93.74±0.22 / 0.0662</td>
</tr>
<tr>
<td>DDML-MLP-Sim</td>
<td>89.08±0.27 / 0.1133</td>
<td>89.69±0.36 / 0.1075</td>
</tr>
</tbody>
</table>

See the figures for detailed graphs.
summing a \( n \)-class problem with two samples in each class, the number of all possible similar pairs is \( n \). But the number of all possible dissimilar pairs is \( 2n(n-1) \), which is much larger than the number of similar pairs. However, the restricted configuration requires the number of dissimilar pairs is the same as the number of similar pairs. For example, only 300 similar pairs and 300 dissimilar pairs are provided in each subset of the LFW dataset. As a consequence, learning on such limited number of dissimilar pairs causes serious over-fitting problems to the normal models, that is why they perform worse than the linear concentrative models. In contrast, when the training is unrestricted, enough dissimilar pairs can be covered during training and the risk of over-fitting is reduced. Hence the normal models trained on both similar and dissimilar pairs perform well in unrestricted training.

In short, restricted training on equal quantity of similar and dissimilar pairs does not accord with the ratio of similar and dissimilar pairs in practice. The similar pairs indeed deliver more positive contributions for learning a better metric. Apart from our suggestion of learning on similar pairs only, this goal can be achieved by other techniques such as shifting the Cosine Similarity boundary [62], using hinge loss functions to filter invalid gradient descent from dissimilar pairs [11] or weighting the gradient contributions from similar and dissimilar pairs [12], [63]. Overall, our proposed concentrative training is a competitive choice due to its simplicity.

5.3.4 TSML vs. DDML

Comparing the two metric learning methods, TSML and DDML, we find comparable performance records in Tables 3 – 6. This is reasonable because the Euclidean distance is naturally related to the Cosine Similarity. For the square of the Euclidean distance between two vectors, we have \((a - b)^2 = (a - b)^T(a - b) = a^2 + b^2 - 2a^Tb\). When the vectors are normalized to unit length, i.e. \(a^2 = b^2 = 1\), the previous equation can be written as \((a - b)^2 = 2 - 2\cos(a, b)\). That means in our situation, minimizing the distance between data pairs is equivalent to maximizing the pairwise similarity value.

5.4 Comparison with the State-of-the-Art

We compared the proposed TSML-Linear-Sim method with several state-of-the-art methods on the LFW dataset under the image-restricted configuration with no outside data [64]. The comparison is summarized in Table 7, and the corresponding ROC curves are shown in Fig. 6. The curves of MRF-MLBP [65] and MRF-Fusion-CSKDA [66] are missing because the curve data are not provided on the public result page. We can see that MRF-Fusion-CSKDA occupies the first place and the proposed TSML-Linear-Sim takes the second one with a relatively large gap (91.90\% vs. 95.89\%). This is because MRF-Fusion-CSKDA employed multi-scale binarized statistical image features and made a fusion on multiple features [66]. However, the proposed TSML-Linear-Sim method is much simpler as it has only utilized a single feature, the FV vectors.

Thus we collected the results of methods using a single feature in Table 8. Especially, we also applied another state-of-the-art approach WCCN [13] on the FV vectors as a comparison. We can see that the proposed TSML-Linear-Sim method achieves the best performance (91.90\%) among all the methods using a single feature. Especially, TSML-Linear-Sim significantly surpasses the conventional Support Vector Machines (SVM) method [30] on the FV vectors by 4.43\% points (from 87.47\% to 91.90\%).

5.5 Stacked to Pre-trained Deep Nonlinearity

As we have mentioned, the proposed shallow nonlinearity was constrained due to lack of proper generalization strategies and more training data. At present, the success of deep learning in speech recognition and visual object recognition shows that the deep nonlinearity is able to learn discriminative representations of data [38]. To release the power of nonlinearity, deep learning approaches require large datasets and perform training in a supervised way [35], [36], [37]. However, it is difficult to directly train a deep metric learning system on a large dataset having hundreds of or even millions of data samples [35], [71] because the number of sample pairs will be dramatically raised. Actually training semi-supervised siamese neural networks is much slower than training supervised neural networks [53]. Recent empirical work showed that training siamese neural networks on carefully chosen triplets instead of data pairs is helpful for fast convergence [72], [73].

Besides, it was also found that even a simple classifier can make good decision on the features produced by the learned deep models [35], [36], [71]. Therefore we stack
the linear models, and it might be possible to improve the nonlinear models by tuning the size of layers, trying different initialization methods or adding regularization techniques. However, the simple linear metric learning model is indeed a good and quick option that demands less effort on hyperparameter tuning than the shallow nonlinear ones. Thus we suggest the deep nonlinearity for robust feature learning on large datasets and the shallow linearity for classification [37].

6 Conclusion

In this paper, we have evaluated two metric learning methods – TSML and DDML – for pairwise face verification on the LFW dataset and pairwise speaker verification on the NIST i-vector dataset. Under the setting of limited training pairs, we found that learning a linear model on similar pairs only is a simple but effective solution for identity verification. When labeled outside data are available, a pre-trained deep CNN model helps the linear TSML and DDML systems to reach competitive performance on face verification.

We presented several strategies and confirmed their effectiveness on reducing the risk of over-fitting. These strategies include using more training pairs; using a linear model to keep generalization; learning on similar pairs only for restricted training; separating a validation set to perform early stopping; introducing a deep CNN model pre-trained on a large dataset. With these strategies, the nature of learning a good metric of the TSML and DDML methods makes themselves effective on the two different pairwise verification tasks.

The defined pairwise verification task is not limited to only human identities, the objects can be documents, audio, images or individuals in any other categories. For any pairwise verification problems with objects that can be represented as numerical vectors, we believe that the proposed methods are applicable, and the observed phenomena are repeatable.

Acknowledgments

Thanks to the China Scholarship Council (CSC) and the Laboratoire d’InfoRmatique en Image et Systèmes d’information (LIRIS) for supporting this work.

References


