

N°d'ordre NNT : 2016LYSEI045

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'INSA LYON

Ecole Doctorale N° 512
INFORMATIQUE ET MATHEMATIQUES

Spécialité de doctorat : Informatique

Soutenue publiquement le 10/05/2016, par :
Lilei ZHENG

Triangular Similarity Metric Learning: a Siamese Architecture Approach

---

Devant le jury composé de :

DORIZZI, Bernadette	Prof.	Télécom SudParis	Présidente
MARCHAND-MAILLET, Stéphane	Prof.	University of Geneva	Rapporteur
THOME, Nicolas	Maître de conférences, HDR	Université Pierre et Marie Curie	Rapporteur
PUECH, William	Prof.	Université de Montpellier	Examinateur
BASKURT, Atilla	Prof.	INSA-LYON	Co-Directeur de thèse
IDRISSI, Khalid	Maître de conférences, HDR	INSA-LYON	Directeur de thèse
GARCIA, Christophe	Prof.	INSA-LYON	Examinateur

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Département FEDORA – INSA Lyon - Ecoles Doctorales – Quinquennal 2016-2020

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **http://www.edchimie-lyon.fr** Sec : Renée EL MELHEM Bat Blaise Pascal 3e etage secretariat@edchimie-lyon.fr Insa : R. GOURDON	M. Stéphane DANIELE Institut de Recherches sur la Catalyse et l'Environnement de Lyon IRCELYON-UMR 5256 Équipe CDFA 2 avenue Albert Einstein 69626 Villeurbanne cedex directeur@edchimie-lyon.fr
E.E.A.	**ELECTRONIQUE, ELECTROTECHNIQUE, AUTOMATIQUE** **http://edeea.ec-lyon.fr** Sec : M.C. HAVGOUDOUKIAN Ecole-Doctorale.eea@ec-lyon.fr	M. Gérard SCORLETTI Ecole Centrale de Lyon 36 avenue Guy de Collongue 69134 ECULLY Tél : 04.72.18 60.97 Fax : 04 78 43 37 17 Gerard.scorletti@ec-lyon.fr
E2M2	**EVOLUTION, ECOSYSTEME, MICROBIOLOGIE, MODELISATION** **http://e2m2.universite-lyon.fr** Sec : Safia AIT CHALAL Bat Darwin - UCB Lyon 1 04.72.43.28.91 Insa : H. CHARLES Safia.ait-chalal@univ-lyon1.fr	Mme Gudrun BORNETTE CNRS UMR 5023 LEHNA Université Claude Bernard Lyon 1 Bât Forel 43 bd du 11 novembre 1918 69622 VILLEURBANNE Cédex Tél : 06.07.53.89.13 e2m2@ univ-lyon1.fr
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTE** **http://www.ediss-lyon.fr** Sec : Safia AIT CHALAL Hôpital Louis Pradel - Bron 04 72 68 49 09 Insa : M. LAGARDE Safia.ait-chalal@univ-lyon1.fr	Mme Emmanuelle CANET-SOULAS INSERM U1060, CarMeN lab, Univ. Lyon 1 Bâtiment IMBL 11 avenue Jean Capelle INSA de Lyon 696621 Villeurbanne Tél : 04.72.68.49.09 Fax :04 72 68 49 16 Emmanuelle.canet@univ-lyon1.fr
INFOMATHS	**INFORMATIQUE ET MATHEMATIQUES** **http://infomaths.univ-lyon1.fr** Sec :Renée EL MELHEM Bat Blaise Pascal 3e etage infomaths@univ-lyon1.fr	Mme Sylvie CALABRETTO LIRIS – INSA de Lyon Bat Blaise Pascal 7 avenue Jean Capelle 69622 VILLEURBANNE Cedex Tél : 04.72. 43. 80. 46 Fax 04 72 43 16 87 Sylvie.calabretto@insa-lyon.fr
Matériaux	**MATERIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry Ed.materiaux@insa-lyon.fr	M. Jean-Yves BUFFIERE INSA de Lyon MATEIS Bâtiment Saint Exupéry 7 avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72.43 71.70 Fax 04 72 43 85 28 Ed.materiaux@insa-lyon.fr
MEGA	**MECANIQUE, ENERGETIQUE, GENIE CIVIL, ACOUSTIQUE** **http://mega.universite-lyon.fr** Sec : M. LABOUNE PM : 71.70 –Fax : 87.12 Bat. Saint Exupéry mega@insa-lyon.fr	M. Philippe BOISSE INSA de Lyon Laboratoire LAMCOS Bâtiment Jacquard 25 bis avenue Jean Capelle 69621 VILLEURBANNE Cedex Tél : 04.72 .43.71.70 Fax : 04 72 43 72 37 Philippe.boisse@insa-lyon.fr
ScSo	**ScSo*** **http://recherche.univ-lyon2.fr/scso/** Sec : Viviane POLSINELLI Brigitte DUBOIS Insa : J.Y. TOUSSAINT viviane.polsinelli@univ-lyon2.fr	Mme Isabelle VON BUELTZINGLOEWEN Université Lyon 2 86 rue Pasteur 69365 LYON Cedex 07 Tél : 04.78.77.23.86 Fax : 04.37.28.04.48

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Triangular Similarity Metric Learning: a Siamese Architecture Approach



Lilei Zheng

Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)
INSA-Lyon

This dissertation is submitted for the degree of
Doctor of Philosophy

Université de Lyon March 2016

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

I would like to dedicate this thesis to my loving parents and my dear wife Liangfen.

路漫漫其修远兮，吾将上下而求索。

— 屈原《离骚》

"The road ahead will be long, and our climb will be steep."

— Qu Yuan (340-278 BC), *Li Sao*

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Acknowledgements

I would like to acknowledge the China Scholarship Council (CSC) and the Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS) to support my thesis.

All my respect and thanks to my supervisors, Khalid Idrissi and Atilla Baskurt, as well as Stefan Duffner and Christophe Garcia. Khalid and Atilla, thank you for admitting me into this great lab LIRIS and for guiding my study in the last three years. I have been always enjoying my life here. Stefan, thank you for sharing me your research ideas and experience. It is you that have taught me a lot of research skills and have brought me the good luck of "paper acceptances". Christophe, thank you for your patience and encouragement on me, you have opened the door of metric learning to me and have been always cheering me up to discover the world behind the door. I will always remember your words, "the key of success is focusing on the work."

Thanks to Yiqiang Chen, Bonan Cuan, Ying Zhang for their valuable comments to my work. Thanks to all the friends and colleagues in LIRIS! *J'aime LIRIS, J'aime Lyon!*

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Abstract

In many machine learning and pattern recognition tasks, there is always a need for appropriate metric functions to measure pairwise distance or similarity between data, where a metric function is a function that defines a distance or similarity between each pair of elements of a set. In this thesis, we propose *Triangular Similarity Metric Learning* (TSML) for automatically specifying a metric from data.

A TSML system is loaded in a siamese architecture which consists of two identical sub-systems sharing the same set of parameters. Each sub-system processes a single data sample and thus the whole system receives a pair of data as the input. The TSML 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, we first propose the *Triangular Similarity*, a novel similarity metric which is equivalent to the well-known Cosine Similarity in measuring a data pair. Based on a simplified version of the Triangular Similarity, we further develop the *triangular loss* function in order to perform metric learning, i.e. to increase the similarity between two vectors in the same class and to decrease the similarity between two vectors of different classes. Compared with other distance or similarity metrics, the triangular loss and its gradient naturally offer us an intuitive and interesting geometrical interpretation of the metric learning objective.

In terms of the mapping function, we introduce three different options: a linear mapping realized by a simple transformation matrix, a nonlinear mapping realized by Multi-layer Perceptrons (MLP) and a deep nonlinear mapping realized by Convolutional Neural Networks (CNN). With these mapping functions, we present three different TSML systems for various applications, namely, pairwise verification, object identification, dimensionality reduction and data visualization. For each application, we carry out extensive experiments on popular benchmarks and datasets to demonstrate the effectiveness of the proposed systems.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Résumé

Dans de nombreux problèmes d'apprentissage automatique et de reconnaissance des formes, il y a toujours un besoin de fonctions métriques appropriées pour mesurer la distance ou la similarité entre des données. La fonction métrique est une fonction qui définit une distance ou une similarité entre chaque paire d'éléments d'un ensemble de données. Dans cette thèse, nous proposons une nouvelle methode, *Triangular Similarity Metric Learning* (TSML), pour spécifier une fonction métrique de données automatiquement.

Le système TSML proposée repose une architecture *Siamese* qui se compose de deux sous-systèmes identiques partageant le même ensemble de paramètres. Chaque sous-système traite un seul échantillon de données et donc le système entier reçoit une paire de données en entrée. Le système TSML comprend une fonction de coût qui définit la relation entre chaque paire de données et une fonction de projection permettant l'apprentissage des formes de haut niveau.

Pour la fonction de coût, nous proposons d'abord la similarité triangulaire (Triangular Similarity), une nouvelle similarité métrique qui équivaut à la similarité cosinus. Sur la base d'une version simplifiée de la similarité triangulaire, nous proposons *la fonction triangulaire* (the triangular loss) afin d'effectuer l'apprentissage de métrique, en augmentant la similarité entre deux vecteurs dans la même classe et en diminuant la similarité entre deux vecteurs de classes différentes. Par rapport aux autres distances ou similarités, la fonction triangulaire et sa fonction gradient nous offrent naturellement une interprétation géométrique intuitive et intéressante qui explicite l'objectif d'apprentissage de métrique.

En ce qui concerne la fonction de projection, nous présentons trois fonctions différentes: une projection linéaire qui est réalisée par une matrice simple, une projection non-linéaire qui est réalisée par Multi-layer Perceptrons (MLP) et une projection non-linéaire profonde qui est réalisée par Convolutional Neural Networks (CNN). Avec ces fonctions de projection, nous proposons trois systèmes de TSML pour plusieurs applications: la vérification par paires, l'identification d'objet, la réduction de la dimensionnalité et la visualisation de données. Pour chaque application, nous présentons des expérimentations détaillées sur des ensembles de données de référence afin de démontrer l'efficacité de notre systèmes de TSML.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table of contents

List of figures **xv**

List of tables **xvii**

1 Introduction **1**
1.1 Context 1
1.2 Definitions and Prerequisites 2
1.3 Applications 3
1.3.1 Pairwise Verification 4
1.3.2 Dimensionality Reduction and Data Visualization 4
1.4 Contribution 5
1.5 Outline 5

2 Literature Review: Siamese Neural Networks and Metric Learning **9**
2.1 Introduction 9
2.2 Siamese Neural Networks 10
2.2.1 Perceptron 10
2.2.2 Multi-Layer Perceptrons 12
2.2.3 Siamese Multi-Layer Perceptrons 16
2.2.4 Convolutional Neural Networks 18
2.2.5 Siamese Convolutional Neural Networks 27
2.3 Metric Learning 28
2.3.1 Distance Metric Learning 29
2.3.2 Similarity Metric Learning 37
2.3.3 Other Advances in Metric Learning 42
2.4 Conclusion and Open Problems 45

3 Triangular Similarity Metric Learning **47**
3.1 Introduction 47

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

3.2 Triangular Similarity . . . 48
3.3 Triangular Loss Function . . . 49
3.4 Relation to Traditional Neural Networks . . . 54
3.4.1 Relation to the Mean Squared Error Function . . . 54
3.4.2 Non-Convexity and Backpropagation . . . 55
3.4.3 Batch Gradient Descent or Stochastic Gradient Descent . . . 56
3.4.4 Various Mapping Functions . . . 57
3.5 Visualization of the Objective . . . 57
3.5.1 Example 1: Two Classes . . . 58
3.5.2 Example 2: Four Classes . . . 59
3.6 Conclusion . . . 63

4 Applications on Pairwise Verification 65
4.1 Introduction . . . 65
4.2 Pairwise Face Verification . . . 66
4.2.1 The LFW Protocols and Related Work . . . 67
4.2.2 Linear Triangular Similarity Metric Learning . . . 68
4.2.3 The LFW Dataset and Face Descriptors . . . 71
4.2.4 Experimental Settings . . . 74
4.2.5 Results and Analysis . . . 76
4.3 Pairwise Kinship Verification . . . 80
4.3.1 The KinFaceW Protocols and Related Work . . . 81
4.3.2 The KinFaceW Dataset and Face Descriptors . . . 82
4.3.3 Experiments and Analysis . . . 84
4.4 Linearity in Pairwise Verification . . . 88
4.4.1 Linear and Nonlinear Triangular Similarity Metric Learning . . . 89
4.4.2 Stochastic Gradient Descent . . . 91
4.4.3 Datasets and Feature Vectors . . . 93
4.4.4 Experiments and Analysis . . . 94
4.5 Conclusion . . . 103

5 Applications on Classification and Dimensionality Reduction 105
5.1 Introduction and Related Work . . . 105
5.2 Classification and Visualization on Small-scale Data . . . 107
5.2.1 Multi-layer Perceptrons . . . 108
5.2.2 The Extended Yale B Dataset and Face Descriptors . . . 113
5.2.3 Dimensionality Reduction in Face Classification . . . 115

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

5.2.4 Dimensionality Reduction in Data Visualization 118
5.3 End-to-end Data Visualization on Large-scale Data 119
5.3.1 The MNIST Dataset and Convolutional Neural Networks 122
5.3.2 Dimensionality Reduction in Data Visualization 125
5.4 Conclusion 134

6 Conclusion and Perspectives **135**
6.1 Conclusion 135
6.2 Perspectives 137

References **139**

Appendix A Derivatives **153**
A.1 Derivative of the vector norm 153
A.2 Derivative of the bilinear similarity 154
A.3 Derivative of the parameterized vector norm 155
A.4 Derivative of the Cosine Similarity 156
A.5 Derivative of the linear triangular loss 157

Appendix B Learning on Similar Pairs Only **159**
B.1 Introduction 159
B.2 Cosine Similarity Metric Learning 159
B.3 Logistic Similarity Metric Learning 160
B.4 Experiment and Analysis 163
B.4.1 Experimental Setting 163
B.4.2 Results and Analysis 163
B.4.3 Learning on Similar Pairs Only 164

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

List of figures

1.1 Diagram of Metric Learning 3

2.1 A perceptron 11
2.2 Activation functions 12
2.3 Diagram of an MLP 13
2.4 Diagram of a Siamese MLP 17
2.5 Fully-connected input layer and convolutional input layer 20
2.6 Convolutional layer with three feature maps 21
2.7 Diagram of a pooling layer 22
2.8 Diagram of a complete CNN 23
2.9 Diagram of LeNet-5 24
2.10 Diagram of a Siamese CNN 27
2.11 Diagram of LMNN 32

3.1 Illustration of Triangular Similarity 49
3.2 Illustration of simplified Triangular Similarity 51
3.3 Diagram of Metric Learning 53
3.4 Geometrical interpretation of the triangular gradient 53
3.5 An MLP for a toy problem 58
3.6 Example 1 of TSML: two classes 60
3.7 Example 2 of TSML: four classes (2-d projections) 61
3.8 Example 2 of TSML: four classes (3-d projections) 62
3.9 Ideal final states: polygons 63
3.10 Ideal final states: polyhedrons 63

4.1 ROC curve of TSML-fusion on LFW-a 81
4.2 ROC curves of different methods on KinFaceW-I 87
4.3 ROC curves of different methods on KinFaceW-II 87
4.4 Dotplots illustrating pairwise similarity matrices 96

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

4.5 Learning curves of TSML models on the LFW data 98
4.6 Performance comparison between TSML and TSML-Sim 101
4.7 ROC curves of TSML-Linear-Sim on LFW-funneled 103

5.1 The MLP used in TSML-MLP 109
5.2 Index matrix for mini-batch gradient descent of TSML-MLP 111
5.3 Example images in the Extended Yale B dataset 113
5.4 Classification performance of TSML-MLP on Extended Yale B 116
5.5 Face images that TSML-MLP failed to recognize 116
5.6 Visualization of data from Extended Yale B 118
5.7 Illustration of dimensionality reduction via TSML-MLP 120
5.8 Example images in the MNIST handwritten digits dataset 122
5.9 Diagram of the proposed CNN architecture 123
5.10 Mapping results after tiny-scale training (2-dimensional) 126
5.11 Results after tiny-scale training with different data size and initialization. . . 126
5.12 Mapping results after large-scale training (2-dimensional) 127
5.13 An unfolded view of the MNIST test data (2-dimensional) 128
5.14 Mapping results before and after unfolding (3-dimensional). 129
5.15 An unfolded view of the MNIST test data (4-dimensional) 130
5.16 Mapping results of the MNIST test data by DrLIM 132

B.1 Accuracy-versus-K curve for LSML on LFW-a 165

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

List of tables

1.1 Summary of notations 7

2.1 Connection scheme between the layers S2 and C3 of LeNet-5 25

4.1 Distribution of individuals and images in LFW 72
4.2 Face verification performance of TSML on LFW-a 77
4.3 Face verification performance of TSML-Sim on LFW-a 77
4.4 Time cost of CSML-Sim-I and TSML-Sim-I 77
4.5 Face verification performance of TSML-fusion on LFW-a 80
4.6 Kinship verification performance on KinFaceW-I and KinFaceW-II 85
4.7 Participants in the FG 2015 Kinship Verification Evaluation 85
4.8 FG 2015 Kinship Verification Evaluation on KinFaceW-I 86
4.9 FG 2015 Kinship Verification Evaluation on KinFaceW-II 86
4.10 Distribution of individuals and speech utterances in NIST i-vector 94
4.11 Proportion of triplets $\{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}$ satisfying that $cos(\mathbf{x}_i, \mathbf{y}_i) > cos(\mathbf{x}_i, \mathbf{z}_i)$ . . . 95
4.12 Face verification performance of TSML and TSML-Sim on LFW-funneled . 100
4.13 Spearker verification performance of TSML and TSML-Sim on NIST i-vector 100
4.14 Comparison of state-of-the-art methods on LFW-funneled 102
4.15 Comparison of methods using single face descriptor on LFW-funneled . . . 102

5.1 Face identification performance on Extended Yale B 117
5.2 Significance testing between MLP and TSML-MLP 117
5.3 Comparison on training time of different methods 131
5.4 Comparison on classification accuracy of different methods 133
5.5 Comparison summary of different methods 134

B.1 Face verification performance of LSML on LFW-a 162
B.2 Face verification performance of LSML-sim on LFW-a 162

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Chapter 1

Introduction

1.1 Context

"No two leaves are exactly alike." — Gottfried Wilhelm Leibniz

This dictum of Leibniz reveals that differences are ubiquitous in this world. Besides, we can also realize another fact that he must have compared a lot of leaves, as well as other objects. In other words, the ubiquity of comparison holds for everyone. Usually, for the same comparison, different people may have different judgement of similarity, since they have their own evaluation standards. However, to facilitate exchanges and communications in many collaboration affairs, we have to set up commonly accepted evaluation standards for people in a certain group.

In mathematics, the numerical measurement of a common evaluation standard is called a metric. In many machine learning and pattern recognition tasks, there is always a need for appropriate metric functions to measure pairwise distance or similarity between data, where a metric function is a function that defines a distance between each pair of elements of a set. For example, in a Cartesian coordinate system, we can use the Euclidean metric to measure the straight-line distance between two points which are usually represented by two position vectors. Formally, for two vectors $\mathbf{x}$ and $\mathbf{y}$, the Euclidean distance between them is $\|\mathbf{x}-\mathbf{y}\| = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})}$. However, in some machine learning applications, a common metric such as the Euclidean distance may be not proper to measure the semantic distance between two objects. In other words, from the point of view of feature representation, we may think that the vector representations of the two objects do not suit the Euclidean space.

Consequently, a technique called Metric Learning has been developed to improve the collocation of feature representations and distance metrics. Continuing with the example of the Euclidean distance, a mapping function $f(\cdot)$ is introduced and the distance between two vectors $\mathbf{x}$ and $\mathbf{y}$ is measured by $\|f(\mathbf{x})-f(\mathbf{y})\| = \sqrt{(f(\mathbf{x})-f(\mathbf{y}))^T(f(\mathbf{x})-f(\mathbf{y}))}$. With

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

this mapping function $f(\cdot)$, the pairwise distance between vectors can be better measured by the new metric function $\|f(\mathbf{x}) - f(\mathbf{y})\|$ than by $\|\mathbf{x} - \mathbf{y}\|$. The procedure of specifying the mapping function $f(\cdot)$ is thus defined as Metric Learning. Similarly, from the point of view of feature representation, the objective of Metric Learning is to learn a new vector representation $f(\mathbf{x})$ (as a substitute of $\mathbf{x}$) that better suits the Euclidean space.

The principal aim of this thesis is to study Metric Learning techniques. We will outline the most important existing approaches to Metric Learning and present a novel method called TSML, short for Triangular Similarity Metric Learning. We will develop a geometrical interpretation of TSML based on the well known *triangle inequality theorem*. We will focus on investigating the effectiveness of TSML on different applications such as face verification, speaker verification, object classification and data visualization.

1.2 Definitions and Prerequisites

In the previous section, we have used the example of Euclidean metric to illustrate what is Metric Learning. Formally, an acknowledged definition is that Metric Learning is the task of learning a metric function over objects[1] [87, 13]. A metric has to obey four axioms: non-negativity, identity of indiscernibles, symmetry and triangle inequality. In practice, Metric Learning algorithms may ignore one or two axioms and learn a *pseudo-metric*.

This acknowledged definition does not indicate the way of learning. In this thesis, we restrict the definition of Metric Learning as *learning a metric function from data pairs*, which is the commonest manner in current Metric Learning algorithms.

Figure 1.1 presents the general diagram of a Metric Learning algorithm. A pair of data samples $(\mathbf{x}_i, \mathbf{y}_i)$ is given as inputs, indicating the i_{th} training pair from a data set. We are going to learn a mapping function $f(\cdot)$ in the "black box", which realizes a projection and results in two outputs $(\mathbf{a}_i, \mathbf{b}_i)$. We call the space of all the inputs as the original space and call the space of all the outputs as the target space. After the mapping, the distance or similarity between two outputs can be measured by a certain metric, and a cost function or loss function is defined based on this metric. Generally, if the pair of inputs is labeled as being similar, the objective of minimizing this cost function is to make the outputs closer to each other; otherwise, for a dissimilar pair of inputs, the objective is to make the outputs more dissimilar/different, i.e. to separate the dissimilar pair in the target space.

In Fig. 1.1, one may notice that the same function $f(\cdot)$ with the same parameter $\mathbf{W}$ is used to process both inputs. This symmetric architecture is called the *siamese architecture*. Particularly, when the mapping function $f(\cdot)$ is realized by neural networks, this technique

[1] https://en.wikipedia.org/wiki/Similarity_learning#Metric_learning

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

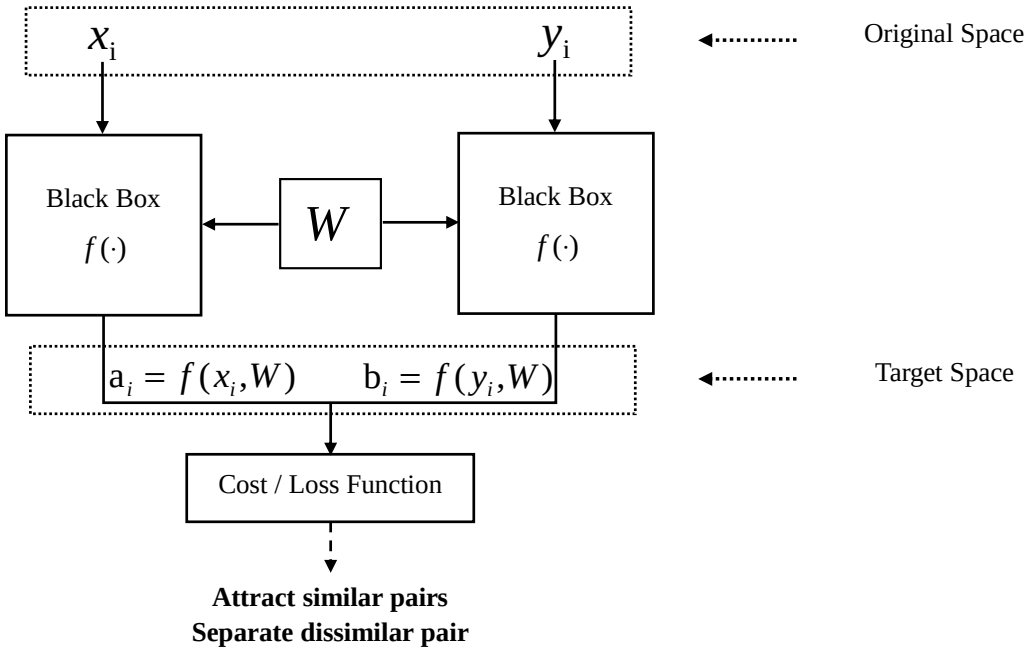


Fig. 1.1 Diagram of Metric Learning.

is also called Siamese Neural Networks. Through the whole text of this thesis, we regard Siamese Neural Networks and Metric Learning as the same technique. Their only difference is that Siamese Neural Networks refer to the symmetric structure of parallel neural networks but Metric Learning emphasizes the pairwise relationship (i.e. the metric) in the data space. In other words, Siamese Neural Networks concern the mapping function $f(\cdot)$ that represents the power (i.e. the complexity) of a system, and Metric Learning mainly concerns the cost function that desires the data relationship in the target space. Nevertheless, an efficient and effective Metric Learning system should be a collaboration of both since the cost function also controls the learning procedure of the mapping.

1.3 Applications

Whenever the notion of pairwise metric between data samples plays an important role, a task can benefit from Metric Learning [13]. For example, in classification tasks, a *k*-Nearest Neighbor (*k*NN) classifier [34] needs a metric to identify the nearest neighbors; for many clustering algorithms such as *k*-means Clustering [110] and Spectral Clustering [124, 164], their performance depends on the quality of distance or similarity measurement between data points. Therefore Metric Learning has been applied to diverse problems such as image classification [121], visual tracking [104], image annotation [59] in the domain of computer

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

vision, as well as document ranking [119], visual localization [92], image retrieval [25] in the domain of content-based information retrieval. In the following, we introduce two fields of application that we focus on in this thesis.

1.3.1 Pairwise Verification

The task of pairwise verification is to verify whether two data samples are semantically similar with each other, i.e. having the same content of interest. Generally, a pair of data samples containing the same semantic content of interest is called a similar pair; otherwise, two samples containing different concerned semantic contents are called a dissimilar pair or a different pair. According to different definitions of the concerned semantic similarity, we study three different problems respectively, namely, pairwise face verification, pairwise kinship verification and pairwise speaker verification.

- **Pairwise face verification**: the task of pairwise face verification is to determine whether two given face images are of the same person or not.
- **Pairwise kinship verification**: a kinship is defined as a relationship between two persons who are biologically related with overlapping genes, thus the task of pairwise kinship verification is to determine whether there is a kin relation between a pair of given face images.
- **Pairwise speaker verification**: the task of pairwise speaker verification is to determine whether two spoken utterances are of the same person or not.

In all the pairwise verification tasks, we usually use the verification accuracy or its contrary, the verification error, to evaluate a Metric Learning method.

1.3.2 Dimensionality Reduction and Data Visualization

When we set the dimension of the target space lower than the dimension of the original space (see Fig. 1.1), Metric Learning techniques perform dimensionality reduction on the inputs. Furthermore, if the target space is a visualizable space, i.e. the dimension of the target space is lower than 3 so that one can see the objects in it, this particular kind of dimensionality reduction is also called data visualization.

In general, when a dimensionality reduction technique projects the original data to a lower dimensional target space, some information of the raw data is discarded. Therefore, besides reducing the dimensionality, maintaining the most useful information for a specific task always plays an important role in practice. For example, in a classification task, the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

classification accuracy usually decreases as the dimension of the target space reduces, so the capability of accurate classification is an important criterion of a good dimensionality reduction technique.

1.4 Contribution

The major contribution of this thesis is that we discovered the *Triangular Similarity* and invented the *triangular loss* as a metric learning cost function. By incorporating various mapping function with the triangular loss, we constructed different TSML systems and applied them to the above two applications, i.e. pairwise verification and dimensionality reduction.

- Recent benchmarks such as the dataset 'Labeled Faces in the Wild' (LFW) [75] and the dataset 'Kinship Face in the Wild' (KinFaceW) [112] established a challenging study of seeking effective learning algorithms which have the ability to discover principles from small numbers of training examples. In these tasks of pairwise verification, we found that under the setting of limited training data, a linear system generally performed better than nonlinear systems because the nonlinear machines were more prone to over-fitting the small training set. On the two popular datasets, our linear TSML system achieved competitive verification performance with the state-of-the-art.

- Without the constraint of limited training, we presented the nonlinear systems to realize flexible dimensionality reduction on data of images, i.e. the Extended Yale B dataset [51] and the MNIST handwritten digits dataset [96]. We succeeded in projecting the original high dimensional image features or the raw images into visualizable spaces while maintaining accurate classification in the target space. Moreover, taking advantage of classical manifold learning theories, the nonlinear TSML systems offered a new perspective of data visualization that significantly advanced the state-of-the-art.

1.5 Outline

In the following chapter (Chapter 2), we will outline some of the most important Metric Learning techniques as well as the advances in Siamese Neural Networks. Furthermore, we will discuss a few open problems in designing a good Metric Learning approach in a siamese architecture.

In Chapter 3, we will then focus on presenting our own approach, the Triangular Similarity Metric Learning (TSML). We will first introduce the definition of Triangular Similarity and

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

then explain the triangular loss function, followed by a geometrical interpretation of the cost function and its gradient function.

After having described the methodology, we will move to the applications. In Chapter 4 we will show the effectiveness of our method on the applications of pairwise verification by experimental comparison with other state-of-the-art methods. We will investigate the effects of several anti-over-fitting strategies by experimental justification.

In Chapter 5, we will apply the proposed method for classification and data visualization on small-scale data and large-scale data, respectively. We will integrate the triangular loss function with neural networks such as Multi-layer perceptrons (MLP) and deep Convolutional Neural Networks (CNN) to realize nonlinear mapping. A particular application of classical manifold learning theories will also be presented.

Finally, Chapter 6 will conclude this thesis with a short summary and draw some perspectives for future research.

Throughout this thesis, we use standard matrix notations to present mathematical objects. A summary of common used notations is given in Table 1.1.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 1.1 Summary of notations

Notation	Description
$\mathbf{A}$	Matrix; or a set of matrices and vectors
A_{ij}	The $(i,j)_{th}$ element of the matrix $\mathbf{A}$
$\mathbf{A}_i$	Indexed matrix for some purpose, e.g. the i_{th} matrix
$\mathbf{A}^{(i)}$	Indexed matrix for some purpose, e.g. the i_{th} matrix
$\mathbf{a}$	Vector (column vector), $\mathbf{a} = [a_1, a_2, \ldots, a_n]^T$
a_i	The i_{th} element of the vector $\mathbf{a}$
$\mathbf{a}_i$	Indexed vector for some purpose, e.g. the i_{th} vector
$\mathbf{a}^{(i)}$	Indexed vector for some purpose, e.g. the i_{th} vector
a	Scalar
N	Scalar
e^N	Natural exponent to the power N
$log(N)$ or $ln(N)$	Natural logarithm of the scalar N
$f(\cdot)$	Function
$f'(\cdot)$	Derivative of the function $f(\cdot)$
$\frac{\partial J}{\partial \mathbf{A}}$	Partial derivative of the cost J over the parameter $\mathbf{A}$
$\Delta\mathbf{a}$	Differential of the parameter vector $\mathbf{a}$
$\Delta\mathbf{A}$	Differential of the parameter matrix $\mathbf{A}$
$\mathbf{A}^T$	Transpose of the matrix $\mathbf{A}$
$tr(\mathbf{A})$	Trace of the matrix $\mathbf{A}$
$\mathbf{I}$	The identity matrix
$\mathbf{a} \odot \mathbf{b}$	Element-wise multiplication between two vectors $\mathbf{a}$ and $\mathbf{b}$
$\|\mathbf{a}\|$	L2-norm (Euclidean norm) of the vector $\mathbf{a}$
$\mathbf{a}^2 = \mathbf{a}^T\mathbf{a}$	Square of the vector $\mathbf{a}$
$\|\mathbf{A}\|$	Frobenius norm of the matrix $\mathbf{A}$
$\mathbf{A} * \mathbf{B}$	Convolution operation (2-d) between two matrices $\mathbf{A}$ and $\mathbf{B}$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Chapter 2

Literature Review: Siamese Neural Networks and Metric Learning

2.1 Introduction

We have mentioned in the previous chapter that we regard Siamese Neural Networks and Metric Learning as two names of the same technique. As their names suggest, the phrase "Siamese Neural Networks" concerns the symmetric structure of parallel neural networks used for mapping but the term "Metric Learning" emphasizes the pairwise relationship (i.e. the metric) in the data space. Actually, while most current Metric Learning methods specify a linear metric, Siamese Neural Networks can be considered as the pioneer of learning a nonlinear metric. In this chapter we will review related literature on Siamese Neural Networks and Metric Learning, respectively.

For Siamese Neural Networks, we will start from introducing a classical type of neural networks, Multi-Layer Perceptrons (MLP). After that, we will present an advanced type of neural networks, Convolutional Neural Networks (CNN) that are of more complex and powerful architectures. Besides, we will show their siamese variants, the Siamese MLP and the Siamese CNN, respectively.

For Metric Learning, we will focus on learning a linear metric since most of current Metric Learning algorithms are linear. We divide current Metric Learning methods into two main families: learning a distance metric or a similarity metric. We will review typical exemplars in each family and also some other advances in Metric Learning.

At last, we will summarize the natural connections between Siamese Neural Networks and Metric Learning as well as the difference between them, followed by a discussion on a few open problems of designing a good architecture and choosing a good metric.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

2.2 Siamese Neural Networks

The word "siam" was the ancient name of Thailand, and the adjective "siamese" means someone or something from Thailand[1]. Its usage of indicating a symmetric structure derives from the phrase "Siamese twins" which refers to the most famous pair of conjoined twins, *Chang and Eng Bunker*[2], from Thailand.

Neural Networks (NN) denote a machine learning technique inspired by the human brain and its capability of accomplishing simple and complex tasks by communications and cooperations between a great amount of neurons, each performing a very simple operation. Like the human brain, an NN is a trainable structure consisting of a set of inter-connected units, each implementing a very simple function, and together eventually realizing a complex classification or regression function. The set of parameters used to configure a certain function is usually called the set of weights in an NN, which can be efficiently learned by the *Backpropagation* algorithm [142].

Combining the two together, a Siamese NN is a special type of NN that consists of two identical sub-networks sharing the same set of weights. We begin this section by introducing the basic components of an NN.

2.2.1 Perceptron

The most well known type of neural unit is called a perceptron which was introduced by Rosenblatt [141]. Its basic structure is illustrated in Fig. 2.1. With n numerical inputs $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and one output y, the value of y is defined as a function of the sum of weighted inputs $\mathbf{w}^T\mathbf{x}$ and an additional bias term b:

$$y = \varphi(\mathbf{w}^T\mathbf{x} + b), \tag{2.1}$$

where $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$ denotes weights for all the inputs, and $\mathbf{w}^T\mathbf{x} = \sum_{i=1}^{n} w_i x_i$ is the weighted sum. The function $\varphi(\cdot)$ is usually called an activation function.

In order to use *Backpropagation* as the learning algorithm for an NN, the activation function has to be differentiable. Commonly used activation functions include the linear function, the sigmoid function, the *tanh* function (i.e. the hyperbolic tangent function) and the ReLU (Rectified Linear Unit) function. The four types of activation functions and their derivatives are listed as below.

[1] https://en.wikipedia.org/wiki/Siamese
[2] https://en.wikipedia.org/wiki/Chang_and_Eng_Bunker

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

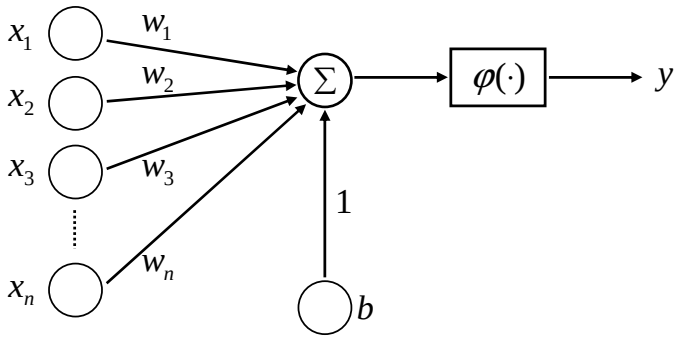


Fig. 2.1 A perceptron

- **Linear function**:

$$\varphi(t) = t, \tag{2.2}$$

$$\varphi'(t) = 1. \tag{2.3}$$

- **Sigmoid function**:

$$\varphi(t) = \frac{1}{1+e^{-t}}, \tag{2.4}$$

$$\varphi'(t) = \varphi(t)[1-\varphi(t)]. \tag{2.5}$$

- **Tanh function**:

$$\varphi(t) = \frac{e^{t}-e^{-t}}{e^{t}+e^{-t}}, \tag{2.6}$$

$$\varphi'(t) = 1-\varphi^2(t). \tag{2.7}$$

- **ReLU function**:

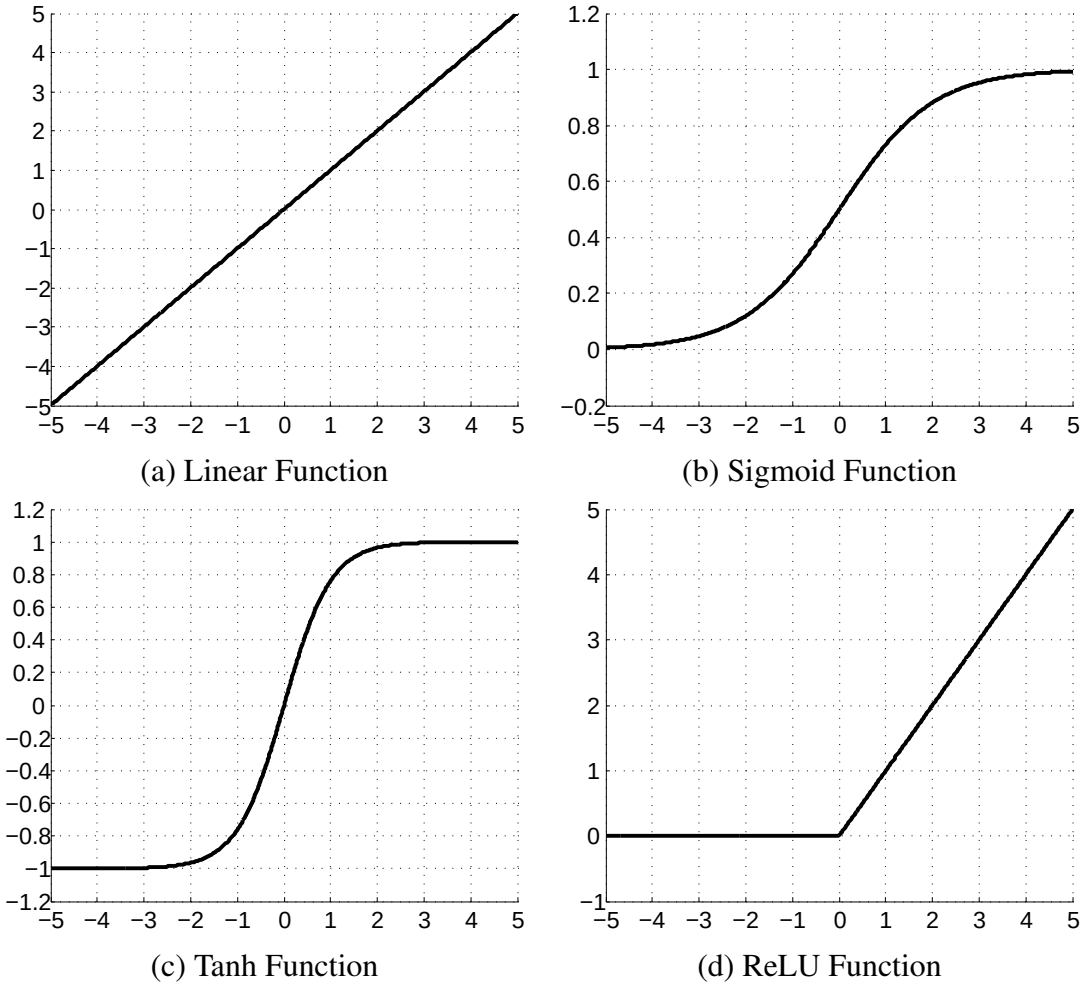
$$\varphi(t) = max(0,t), \tag{2.8}$$

$$\varphi'(t) = \begin{cases} 1, & t > 0; \\ 0, & t \leq 0. \end{cases} \tag{2.9}$$

Note that the ReLU function is actually not differentiable at the point 0*. Hence in practical implementations,* $\varphi'(0)$ *is usually set to* 0*.*

Figure 2.2 shows curves of the four activation functions. The linear function (Fig. 2.2 (a)) is often used in an NN for linearly separable problems. In contrast, it is the *nonlinear activation function* that allows an NN to compute nontrivial problems using only a small number of nodes. The sigmoid non-linearity is shown in Fig. 2.2 (b). It takes a real-valued number and "squashes" it into range between 0 and 1. In particular, large negative numbers become 0 and large positive numbers become 1. The sigmoid function has been frequently

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) Linear Function (b) Sigmoid Function

(c) Tanh Function (d) ReLU Function

Fig. 2.2 Four typical activation functions in a perceptron.

used because of its nice interpretation as the firing rate of a neuron: from not firing at all (i.e. 0) to fully-saturated firing at an assumed maximum frequency (i.e. 1). Unlike the sigmoid function, the tanh function squashes a real-valued input to the range $[-1,1]$ where its output is zero-centered (Fig. 2.2 (c)). Therefore in practice the tanh function is usually preferred to the sigmoid function [100]. However, the ReLU activation function was argued to be more biologically plausible [54] than the widely used sigmoid and tanh functions. As of the year 2015, the ReLU activation function has been the most popular activation function for deep neural networks [99].

2.2.2 Multi-Layer Perceptrons

By combining several interconnected perceptrons together, Multi-Layer Perceptrons (MLP) are able to approximate arbitrary nonlinear mappings and thus have been the most popular

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

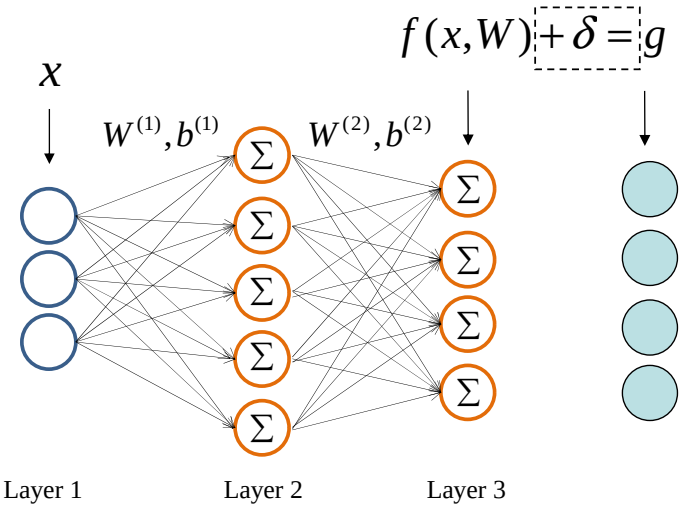


Fig. 2.3 Diagram of a 3-layer MLP. A node with the sign $\sum$ represents a preceptron.

kind of NN since the 1980's [142]. It finds applications in diverse fields such as image recognition [181] and speech recognition [106, 21].

A classical MLP consists of an input layer, one or more hidden layer(s) and an output layer of neurons. An MLP is a *feed-forward* neural network, i.e. the activation of the neurons is propagated layer-wise from the input to the output layer [43]. Figure 2.3 illustrates the structure of a 3-layer MLP consisting of an input layer, an output layer and only one hidden layer, where a node with the sign $\sum$ represents a preceptron described in the previous section. We use an input vector $\mathbf{x}$ to represent the inputs and let $\mathbf{W}$ denote the weights of the MLP, i.e. all the parameters between any two adjacent layers, namely, $\mathbf{W}:\{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}\}$. With an mapping function $f(\cdot)$, this MLP produces an output vector $f(\mathbf{x}, \mathbf{W})$.

Generally, in a multi-class classification problem, the size of the output layer (i.e. the output dimension), is fixed to the number of classes in this problem. The objective of such an MLP is to make the network outputs approximating predefined target values (or ground truth) for different classes. In practice, the error $\boldsymbol{\delta}$ between the output $f(\mathbf{x}, \mathbf{W})$ and a predefined target vector $\mathbf{g}$ is used to update the network parameters via the Backpropagation algorithm [142]. Moreover, these predefined target values are typically binary for classification problems. For example, for a 4-class classification problem, we set unit vectors $[1,0,0,0]^T$, $[0,1,0,0]^T$, $[0,0,1,0]^T$, $[0,0,0,1]^T$ as target vectors for the 4 classes, respectively.

Training an MLP: Backpropagation

The Backpropagation algorithm [142] is the most common and maybe the most universal training algorithm for feed-forward NNs. The word "backpropagation" is the abbreviation

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

for "backward propagation of errors". Hence the Backpropagation algorithm can be divided into two phases: (1) *a forward phase* from the input layer to the output layer to compute errors; (2) *a backward phase* from the output layer to the input layer to update the weights. In the following paragraphes, we take the 3-layer MLP in Fig. 2.3 as an example to introduce the two phases.

A forward phase is first taken to compute errors for some training data. Formally, for any given input sample $\mathbf{x}_i$, assuming its output through the MLP is $\mathbf{a}_i = f(\mathbf{x}_i, \mathbf{W})$. At the first step, from the input layer to the hidden layer, with the parameter matrix $\mathbf{W}^{(1)}$ and the bias vector $\mathbf{b}^{(1)}$, values in the hidden layer are computed as

$$\mathbf{h}_i = \varphi(\mathbf{z}_i^{(1)}) = \varphi(\mathbf{W}^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}). \tag{2.10}$$

At the second step, from the hidden layer to the output layer, with the parameter matrix $\mathbf{W}^{(2)}$ and the bias vector $\mathbf{b}^{(2)}$, the output values are calculated as

$$\mathbf{a}_i = \varphi(\mathbf{z}_i^{(2)}) = \varphi(\mathbf{W}^{(2)}\mathbf{h}_i + \mathbf{b}^{(2)}). \tag{2.11}$$

The function $\varphi(\cdot)$ here is an activation function in a perceptron (see Section 2.2.1). Finally, cost function of this MLP is simply the Mean Squared Error (MSE) between the computed outputs and their desired targets for all training samples:

$$J = \frac{1}{N}\sum_{i=1}^{N} J_i = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{a}_i - \mathbf{g}_i)^2, \tag{2.12}$$

where N is the number of all possible training samples, $\mathbf{g}_i$ is the target vector for the output $\mathbf{a}_i$. Remind that $\mathbf{g}_i$ is usually hand-crafted unit vectors. For example, for a 3-class classification problem, we set unit vectors $[1,0,0]^T$, $[0,1,0]^T$, $[0,0,1]^T$ as target vectors for the 3 classes, respectively. Minimizing the cost function leads to an optimal solution of correctly classifying the training data, which is realized by the backward phase.

A backward phase is then taken to update the set of parameters $\mathbf{W} : \{\mathbf{W}^{(2)}, \mathbf{b}^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\}$. Taking derivative of Equation (2.12), the gradient for the i_{th} training sample is:

$$\frac{\partial J_i}{\partial \mathbf{W}} = (\mathbf{a}_i - \mathbf{g}_i)\frac{\partial \mathbf{a}_i}{\partial \mathbf{W}}, \tag{2.13}$$

and the derivative on the output layer, with respect to $\mathbf{z}_i^{(2)} = \mathbf{W}^{(2)}\mathbf{h}_i + \mathbf{b}^{(2)}$, is:

$$\boldsymbol{\delta}_i^{(2)} = \varphi'(\mathbf{z}_i^{(2)}) \odot (\mathbf{a}_i - \mathbf{g}_i), \tag{2.14}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

where the notation $\odot$ means element-wise multiplication and the function $\varphi'(\cdot)$ here is the derivative of an activation function $\varphi(\cdot)$. Subsequently, the derivative on the hidden layer, with respect to $\mathbf{z}_i^{(1)} = \mathbf{W}^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}$, is:

$$\boldsymbol{\delta}_i^{(1)} = \varphi'(\mathbf{z}_i^{(1)}) \odot [(\mathbf{W}^{(2)})^T \boldsymbol{\delta}_i^{(2)}]. \tag{2.15}$$

The differentials of the network parameters are computed as:

$$\Delta_i \mathbf{W}^{(2)} = \boldsymbol{\delta}_i^{(2)} \mathbf{h}_i^T, \tag{2.16}$$

$$\Delta_i \mathbf{b}^{(2)} = \boldsymbol{\delta}_i^{(2)}, \tag{2.17}$$

$$\Delta_i \mathbf{W}^{(1)} = \boldsymbol{\delta}_i^{(1)} \mathbf{x}_i^T, \tag{2.18}$$

$$\Delta_i \mathbf{b}^{(1)} = \boldsymbol{\delta}_i^{(1)}. \tag{2.19}$$

After that, the parameters $\mathbf{W} : \{\mathbf{W}^{(2)}, \mathbf{b}^{(2)}, \mathbf{W}^{(1)}, \mathbf{b}^{(1)}\}$ can be iteratively updated by using the following function:

$$\mathbf{W} \leftarrow \mathbf{W} - \mu \sum_{i=1}^{N} \Delta_i \mathbf{W}, \tag{2.20}$$

where μ is the learning rate in an online gradient descent learning algorithm, $\Delta_i \mathbf{W}$ indicates the differentials in Equations (2.16–2.19). For the iterative updating algorithm, the starting values of the weights $\mathbf{W}$ have a significant effect on the training process. Empirically, weights should be chosen randomly but in such a way that the activation function is primarily activated in its linear region [100, 53]. After adequate training iterations, an optimal $\mathbf{W}$ will be reached as a proper solution to the predefined cost function (Equation (2.12)).

Training an MLP: gradient descent

Gradient descent is the way of realizing Backpropagation and minimizing functions [9]. Given a function defined by a set of parameters, gradient descent starts with an initial set of parameter values and iteratively moves toward a set of parameter values in order to minimize the function. This iterative minimization is achieved using calculus, taking proportional steps in the negative direction of the function gradient (e.g. Equation (2.20)).

At each iteration, if it requires a complete pass through the entire training data set to compute an average gradient, this type of learning is referred as *batch gradient descent*, where "batch" indicates the entire training set. Alternatively, if a single training sample is chosen from the training set at each iteration, it is called *stochastic gradient descent*. Stochastic gradient descent is generally preferred due to the following three reasons [100]:

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Advantages of Stochastic Gradient Descent

1. Stochastic gradient descent is usually much faster for a single iteration.
2. Stochastic gradient descent can be used for tracking changes.
3. Stochastic gradient descent may result in better solutions.

Firstly, since stochastic gradient descent trains on a single sample in each iteration but batch gradient descent requires the entire training set, stochastic gradient descent is most often much faster in iterative updating. Besides, stochastic gradient descent is particularly useful to model a function changing when the underlying local data distribution changes gradually over time. Since batch gradient descent always considers the whole training data, it captures the global distribution and produces a mean solution. In this case, stochastic gradient descent usually yields better approximation results and thus is preferred for online learning.

Besides the need of online learning, offline learning is still useful in some applications, e.g. for small and medium scale problems, the entire training data always obey a certain distribution. In this case, some advanced optimization algorithms, such as the Conjugate Gradient Descent (CGD) algorithm [115] and the Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm [108], can help batch gradient descent to automatically accelerate the learning speed and produce good results very quickly.

Nowadays, it is more likely to get a large scale problem with a redundant training set, thus one may prefer stochastic gradient descent or its variant, mini-batch gradient descent. Like stochastic gradient descent, mini-batch gradient descent is also designed for online learning. In particular, it takes several training samples in each iteration. Usually, as the trade-off between stochastic gradient descent and batch gradient descent, mini-batch gradient descent is the best choice among the three for online optimization problems [100].

2.2.3 Siamese Multi-Layer Perceptrons

Despite MLP has been the most popular kind of NN since the 1980's [142] and the siamese architecture has been first presented in 1993 [24], most Siamese NNs utilized Convolutional Neural Networks (CNN) for image analysis [24, 32, 43]. Until recently, a few works studied Siamese MLPs: Chen and Salman [31] applied a Siamese MLP to extract speaker-specific information (2011); Yih *et al.* [177] employed it to measure the similarity between texts (2011); Berlemont *et al.* [18] used it for gesture recognition (2015); Zheng *et al.* [182] adopted it for face identification (2015).

A Siamese MLP is a symmetric architecture consisting of two MLPs, where they actually share the same set of parameters $\mathbf{W}$ (Fig. 2.4). Like the standard MLP minimizes the MSE

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

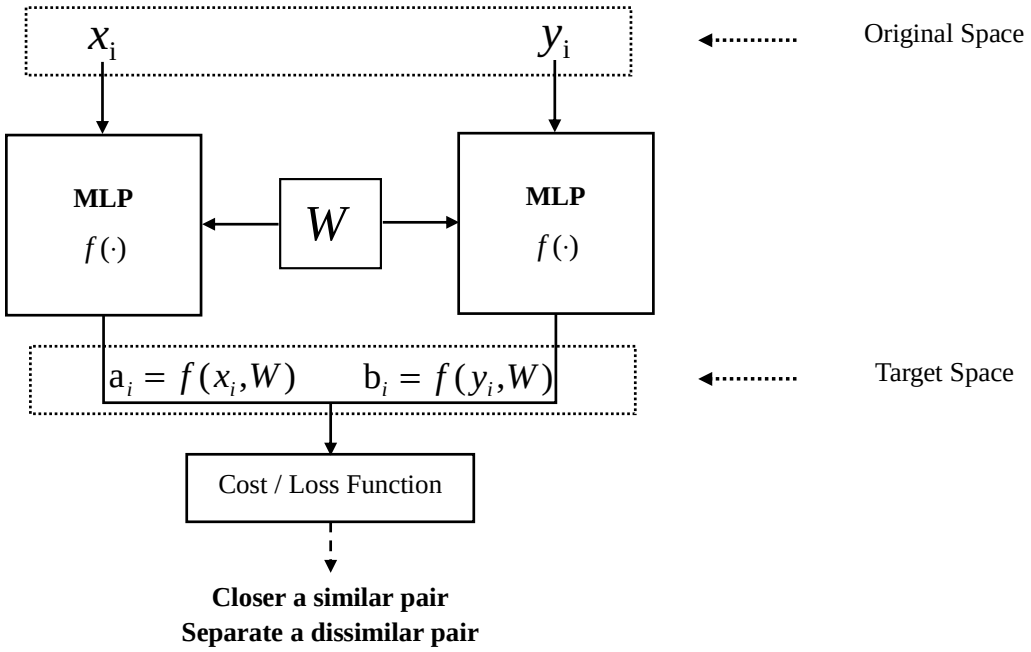


Fig. 2.4 Diagram of a Siamese MLP.

error function to get a proper classification solution, different kinds of cost functions may be designed in the Siamese MLP for different applications. For example, Berlemont *et al.* proposed a cost function based on the Cosine Similarity metric in their Siamese MLP and employed it to recognize and reject inertial gestures [18]; Zheng *et al.*developed a triangular loss function for the Siamese MLP to realize dimensionality reduction and data visualization [182].

Compared with the standard MLP (Fig. 2.3), instead of constraining the outputs approaching some predefined target values (Section 2.2.2), the Siamese MLP defines a specific objective: (1) for an input pair from the same class, making the pairwise similarity between their outputs larger or making the pairwise distance between the outputs smaller; (2) for an input pair from different classes, making the pairwise similarity between their outputs smaller or making the pairwise distance between the outputs larger. By this objective, there is no need to handcraft target vectors for training classes. Consequently, unlike the classical MLP fixes the size of the output layer (i.e. the output dimension) to the number of classes in a certain problem, dimension of the target space can be arbitrarily specified by the Siamese MLP.

Another advantage of the Siamese MLP over the classical MLP is that the Siamese MLP is able to learn on data pairs instead of fully labeled data. In other words, the Siamese MLP is applicable for weakly supervised cases where we have no access to the labels of

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

training instances: only some side information of pairwise relationship is available. This is a meaningful setting in various applications where labeled data are more costly than the side information [13]. Examples include users' implicit feedback on the internet (e.g. clicks on search engine results), citations among articles or links in a social network, kinship relationship between individuals [114].

Training a Siamese MLP is almost the same to the training procedure of a standard MLP. The only difference is that a Siamese MLP takes pairs of training samples into account in each iteration while a standard MLP takes single data samples.

2.2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a specialized kind of neural networks for processing data that have a known *grid-like topology* [16]. The word "convolutional" indicates that the networks employ a mathematical operation called *convolution*. As images are the most common grid-like data (i.e. 2-dimensional data), CNNs are widely used for image and video analysis [96, 127, 91]. Besides images, CNNs are also able to process other 2-dimensional input such as speech time-series [95]. See [98] for a more in-depth history of CNN applications.

Compared with an MLP, a CNN is a more complex type of neural networks since it includes more kinds of neuron layers. And the complexity makes CNNs known as an important technique in the history of *deep learning* [99]. The layers in an MLP are also called fully-connected layers (see Fig. 2.3) since any two adjacent layers are fully connected. A typical CNN consists of a number of convolutional and pooling layers optionally followed by some fully-connected layers.

Convolutional layer

A fully-connected layer receives a single vector as an input and transform it through perceptrons (see Fig. 2.3). Consequently, it is difficult to apply it directly to process an image which is usually represented by a matrix of pixels. A compromised way is stretching the matrix as a vector, i.e. concatenating the rows or columns of the matrix, before delivering it to the fully-connected layer. However, this would cause two problems: (1) realigning the matrix ignores local correlations of neighboring pixels in an image, but the local correlation carries rich texture information that is important in detection or recognition tasks; (2) the input dimension will be too high. Even for a 100×100 small image, the size of the input layer (i.e. the layer receives the image) is $10,000$. As the next layer is fully connected to the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

input layer, the number of parameters (i.e. the number of connections) would be very large and thus the MLP is too complex to train and is more inclined to get over-fitting.

In contrast, a convolutional layer alleviates these problems. For the first problem of ignoring local correlations, the convolutional layer directly takes a matrix as an input without breaking the existing grid-like topology in an image. For the second problem, the convolutional layer uses two basic ideas: *local receptive fields* and *shared weights* to reduce the complexity of the NN (i.e. to reduce the number of parameters).

Figure 2.5 illustrates the input layers in an MLP and a CNN, respectively. Concretely, an MLP takes a vector as the input, and each node in the first hidden layer has connections with all the nodes in the input layer (See Fig. 2.5 (a) and Fig. 2.3). A CNN receives a matrix as the input, but connects a hidden node to a small region of nodes in the input layer. This region is called the *local receptive field* for the hidden node. For example, in Fig. 2.5 (b), we define a convolutional input layer to process 10×12 matrices and use a 5×5 sliding window to transform the local receptive fields to the hidden nodes. In this example, the sliding window moves towards the right or downwards with a step size 1, thus we have the hidden layer with the size 6×8. It is worth knowing that unlike the size of a hidden layer in an MLP can be arbitrarily specified, the size of a hidden layer in a CNN is conditioned by three factors: the size of the former convolutional layer, the size of the sliding window and the step size.

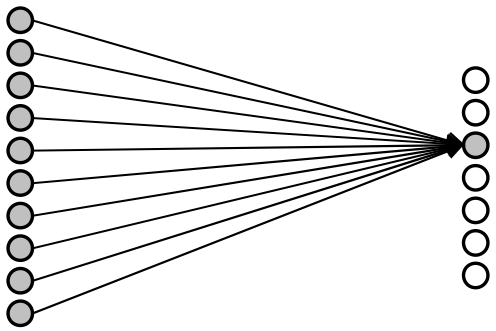
Moreover, the sliding window uses the same mapping function for all the hidden nodes, i.e. all the mappings between a local receptive field and a hidden node share the same weights. This idea is called *shared weights* or *parameter sharing*[3] [126]. A probable principle behind this idea is that natural images have the property of being stationary, meaning that the basic statistical features (e.g. edges or textures) detected from one part of the image are also useful to represent any other parts. For instance, the two local receptive fields in Fig. 2.5 (b) are processed by the same mapping function which performs a convolution operation. Let $\mathbf{X}$ denote the input matrix and $\mathbf{X}^{(ij)}$ denote the local receptive filed with the upper left corner at position (i, j), e.g. the two sub-matrices in Fig. 2.5 (b) are $\mathbf{X}^{(11)}$ and $\mathbf{X}^{(55)}$, respectively. With a weight matrix $\mathbf{W}$ and a bias term b, the output value y delivered to the hidden layer is:

$$y = \varphi(\mathbf{W} * \mathbf{X}^{(ij)} + b), \tag{2.21}$$

where the function $\varphi(\cdot)$ is an activation function that we often use in a perceptron (Section 2.2.1) and the operator $*$ represents a *convolution operation* [16]. Comparing this function with the mapping function in a perceptron (Equation (2.1)), there are two differences: (1) the inputs and weights are matrices instead of vectors; (2) the inner product

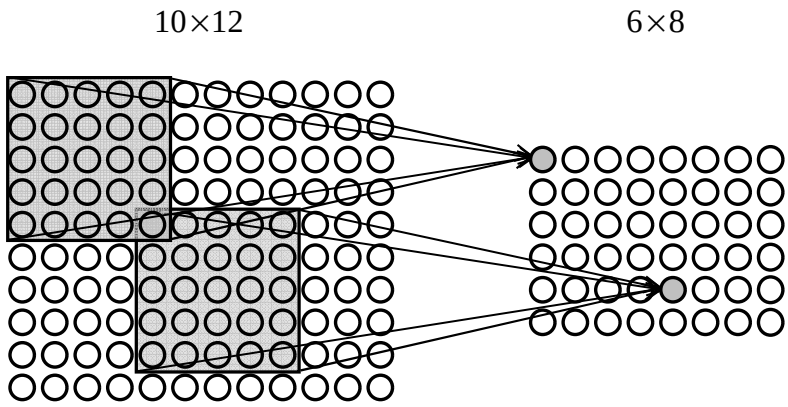
[3] http://deeplearning.net/tutorial/lenet.html

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



Fully-connected layer (Input layer) First hidden layer

(a) Input layer in an MLP



Convolutional layer (Input layer) First hidden layer

(b) Input layer in a CNN

Fig. 2.5 Comparison of input layers: fully-connected layer (MLP) vs. convolutional layer (CNN). Size of the local receptive field (i.e. the square window) is 5×5.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

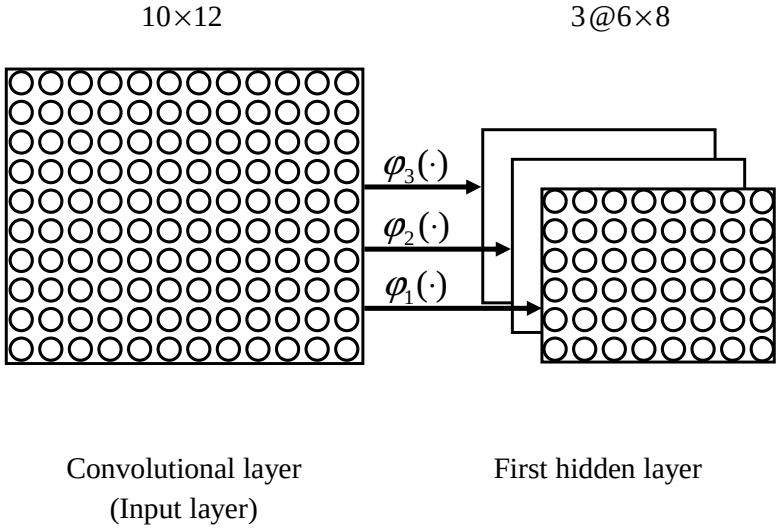


Fig. 2.6 Convolutional layer with three feature maps.

between two vectors is substituted by the 2-dimensional convolution operator between two matrices. Especially, since this convolutional mapping function has the capacity of detecting a certain feature, it is usually called a *feature map*. In practical image analysis, only one feature is not enough to describe the content of an image, hence we need multiple features maps in a CNN. Figure 2.6 shows an example of 3 feature maps from the convolutional input layer to the hidden layer. These 3 feature maps are defined by $\varphi_i(\cdot)$ $(i = 1,2,3)$, respectively. We can say that each function learns a feature map on the input image, resulting in 3 pages of hidden units in the following layer.

In summary, taking advantage of the ideas of *local receptive fields* and *shared weights*, a convolutional layer realizes local feature detection and avoids the problem of overlarge input that a fully-connected layer may be hard to handle. Furthermore, the convolution operation has a property of *invariance to local translation*, which plays an important role in detecting local features since we care more about whether a feature is present rather than where its exact position is [16].

Pooling layer

One may notice that after the convolution operation, the size of the hidden layer is still very large. To further reduce the computational complexity, a pooling layer or a sub-sampling layer is usually used immediately after a convolutional layer. Concretely, a pooling operation in a hidden layer summarizes the information in a local region and delivers the summarized statistic to the next hidden layer. The local region is usually called a *pool*. For example, in

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

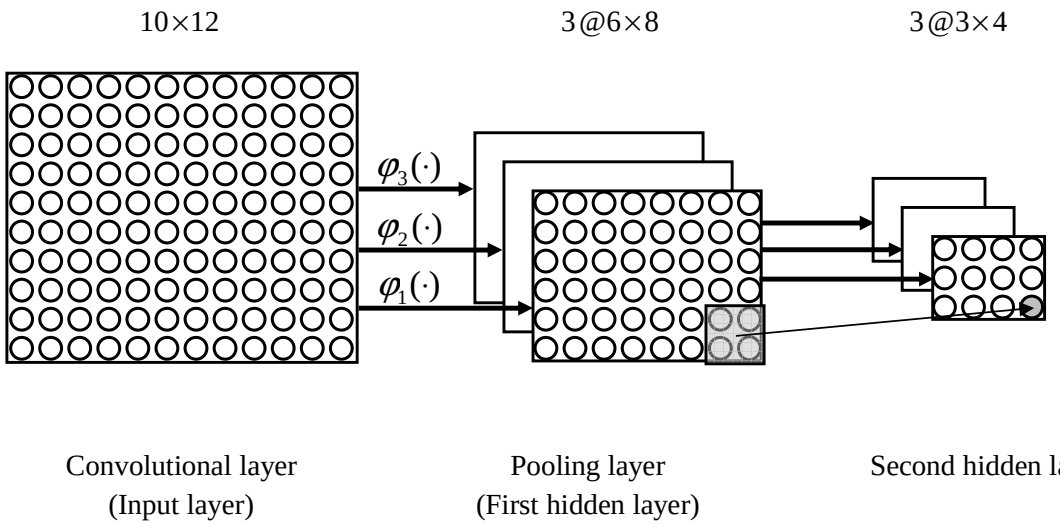


Fig. 2.7 Diagram of a pooling layer.

Fig. 2.7, a page of hidden units are divided into non-overlapped 2×2 pools and an output is generated from each pool. Overall, the three 6×8 pages in the pooling layer are downsampled to three 3×4 pages in the second hidden layer. This example employs non-overlapping pooling, but it is worth knowing that like in the convolutional layer, overlapping regions are also allowed by using a sliding window.

Let $\mathbf{X} : \{X_{ij}\}$ denote the matrix in a pool, common used pooling functions include:

- *Max pooling*: taking the maximal element in $\mathbf{X}$ as the output,

$$y = \max(\mathbf{X}) = \max_{i,j} X_{ij}. \tag{2.22}$$

- *Average pooling*: taking the average value of all the elements $\{X_{ij}\}$ as the output,

$$y = \frac{1}{N} \sum_{i,j} X_{ij}, \tag{2.23}$$

  where N is the number of elements in $\mathbf{X}$.

- *L2-norm pooling*: taking the l_2 norm of all the elements $\{X_{ij}\}$ as the output [45],

$$y = (\sum_{i,j} X_{ij}^2)^{\frac{1}{2}}, \tag{2.24}$$

  please note that it is not the l_2 norm of a matrix, but the Frobenius norm [132]. These functions carry no parameters to perform pooling, however, there are also pooling

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

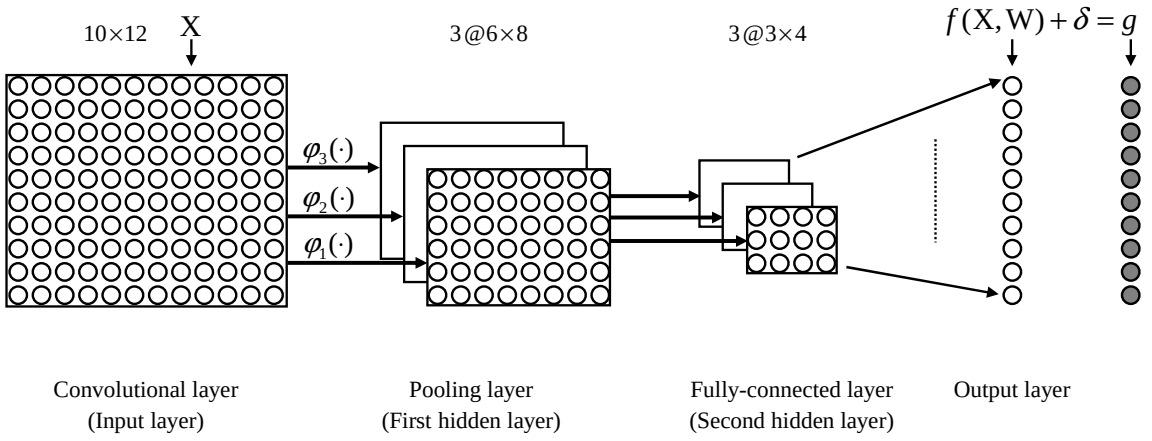


Fig. 2.8 A complete CNN architecture.

functions with trainable parameters [96, 49]. A pooling operation with parameters is also considered as a kind of *feature map*.

Overall architecture

After collecting all necessary components, we can now present a complete CNN architecture. Following the previous convolutional and pooling layers, an output layer is simply linked to the second hidden layer with fully connections (see Fig. 2.8). Just like in an MLP (Fig. 2.3), a ground truth (i.e. a target vector) $\mathbf{g}$ is also set up for the values in the output layer.

Despite that a CNN realizes a more complex transformation than an MLP, the objective of the CNN is still to make the outputs $f(\mathbf{X},\mathbf{W})$ approximating predefined target values. The error $\boldsymbol{\delta}$ between the output $f(\mathbf{X},\mathbf{W})$ and the target vector $\mathbf{g}$ is used to update the network parameters via the Backpropagation algorithm [142].

Training a CNN

Like training an MLP, we also use Backpropagation and gradient descent (Section 2.2.2) to train a CNN. The only required supplements are the derivatives of the pooling and convolutional functions.

For a pooling layer, since it usually does not carry any parameters to perform learning, the backward phase for a pooling operation is very simple. For example, in the forward phase, a max pooling function simply delivers the maximal value in a pool to the next layer, thus it is the neuron unit having this maximal value to receive the error during the backward phase. Accordingly, if we have average pooling in the forward phase, the error would be averagely distributed to all the units in the pool. For a convolutional layer, a depth description of the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

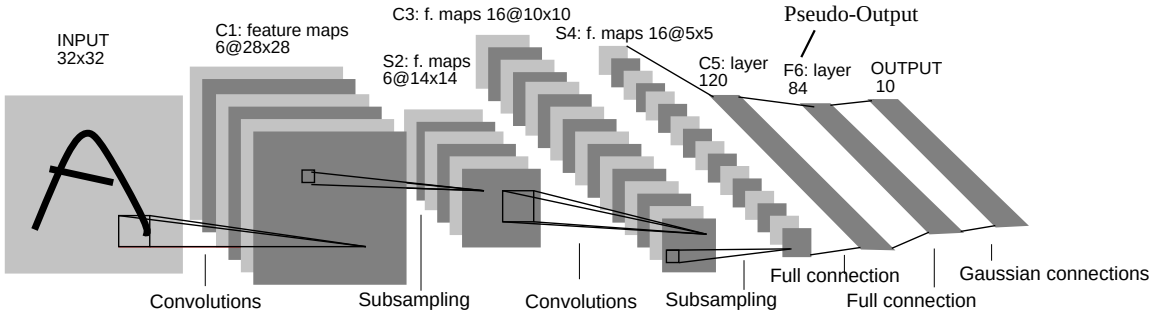


Fig. 2.9 Diagram of LeNet-5 [96].

convolution operator and its derivative is referred to [16, 22, 43]. But it is worth knowing that the backward phase for a convolution operation is also a convolution with a spatially-flipped operator.

A foundation stone: LeNet-5

We have known that a CNN architecture comprises several kinds of neuron layers. Therefore configuring each component, i.e. determining the number of layers, the number of feature maps, the size of local receptive fields, the size of pools, is a complex task. In other words, designing a CNN for a given practical problem indeed becomes an art. Fortunately, the founder of CNN, Yann LeCun, has offered us delicate exemplars such as *LeNet-5* [96]. LeNet is a series of successful CNN applications developed by Yann LeCun since the early 1990's [94, 96], and it has inspired the creation of many successive CNN architectures such as AlexNet [86], ZF Net [179], GoogLeNet [159] and VGGNet [151].

Now we review the principal architecture of *LeNet-5* in Fig. 2.9. Note that the naming style in LeNet-5 is different from ours in previous sections. For two adjacent layers, a name of "convolutional", "pooling" or "fully-connected" is given to the former layer in our case (see Fig. 2.8) but LeNet-5 gives the name to the latter layer. As a result, the first convolutional layer is the input layer in our definition instead of the first hidden layer in LeNet-5. However, this difference does not matter since the name actually indicates the operation between the two layers.

LeNet-5 is composed of 7 layers, not counting the input layer. There are three convolutional layers at the 1st, 3rd and 5th layers, denoted by C1, C3 and C5. Particularly, C1 and C3 are followed by two sub-sampling layers (i.e. pooling layers) S2 and S4. After C5, two more layers are defined to implement classification.

For all the three convolutional layers, C1, C3, C5, the size of local receptive fields is always 5×5. Each feature map is learned on one or more pages of hidden units in the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 2.1 Connection scheme between the layers S2 and C3 of LeNet-5: a cross indicates a connection between the indexed pages in S2 and C3.

S2 \ C3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	X				X	X	X			X	X	X	X		X	X
2	X	X				X	X	X			X	X	X	X		X
3	X	X	X				X	X	X			X		X	X	X
4		X	X	X			X	X	X	X			X		X	X
5			X	X	X			X	X	X	X		X	X		X
6				X	X	X			X	X	X	X		X	X	X

previous layer. Taking a feature map learned on 3 previous pages for example, three 5×5 weight matrices would be used to perform convolution on the three pages in the previous layer, respectively. And then the results are added to a trainable bias. Such a feature map contains 76 $(25 \times 3 + 1)$ parameters.

For all the sub-sampling layers, the size of pools is always 2×2. Especially, LeNet-5 adopts a parametric sub-sampling operation: the four inputs in a pool are added, then multiplied by a trainable coefficient, and added to a trainable bias; finally, the result is passed through a sigmoid function. Thus each feature map of sub-sampling has 5 connections between hidden units but only 2 parameters.

Layer C1 performs a convolution operation with 6 feature maps on a 32×32 input image, resulting in 6 pages of hidden units in the first hidden layer. Since each feature map here has 26 $(25 + 1)$ parameters, C1 contains 156 (6×26) trainable parameters and $122,304$ $(28 \times 28 \times 156)$ connections.

On each of the 6 hidden pages, Layer S2 learns a feature map of sub-sampling. The 2×2 pool area is non-overlapping, thus we observe image down-sampling from 28×28 in C1 to 14×14 in S2. S2 contains 12 (6×2) trainable parameters and $5,880$ $(14 \times 14 \times 6 \times 5)$ connections.

Layer C3 is a convolutional layer with 16 feature maps. Each feature map is learned on several pages of hidden units in the previous layer S2. Table 2.1 shows the connection scheme between the layers S2 and C3. A cross indicates a connection between the indexed pages in S2 and C3. For example, the first page (i.e. the first feature map) in C3 is learned on the first three pages of hidden units in S2. This non-complete connection scheme forces different feature maps in C3 to extract different features from S2 [96]. The number of page connections between the two layers is 60. Thus C3 has $1,516$ $(60 \times 25 + 16)$ trainable parameters and $151,600$ $(10 \times 10 \times 1,516)$ connections between hidden units.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Layer S4 is a sub-sampling layer with 16 feature maps. The sub-sampling operation from C3 to S4 is similar to that between C1 and S2. Thus S4 has 32 (16×2) trainable parameters and 2,000 ($5 \times 5 \times 16 \times 5$) connections.

Layer C5 performs the last convolution operation on the previous layer S4 with 120 feature maps. And each feature map is learned on all the pages in S4. Since the size of hidden pages in S4 is also 5×5, exactly the same with the size of the local receptive fields in LeNet-5, S4 and C5 are actually fully-connected with each other. Particularly, C5 contains 48,120 ($120 \times (25 \times 16 + 1)$) trainable parameters/connections.

The 6th layer F6 is a fully-connected layer having a modified tanh activation function. It contains 84 hidden units and thus 10,164 ($84 \times (120 + 1)$) trainable parameters/connections. Actually, we may consider this layer as the output layer because LeNet-5 directly uses the results of this layer to calculate the final cost. To distinguish this layer from the "OUTPUT" layer in Fig. 2.9, we call F6 the pseudo-output layer. The 7th layer, the "OUTPUT" layer in Fig. 2.9, is composed of Euclidean Radial Basic Function (RBF) units which store classification decision costs for each class.

Let $\mathbf{x}=[x_1, x_2, \cdots, x_{84}]^T$ denote the output of F6, the cost of recognizing a digit as the number $j \in \{0,1,2,3,4,5,6,7,8,9\}$ is calculated as:

$$C_j = (\mathbf{x} - \mathbf{w}_j)^2, \tag{2.25}$$

where $\mathbf{w}_j$ is a vector containing 84 values which form a 7×12 bitmap representing a stylized image of the number j. And this vector $\mathbf{w}_j$ plays the role of a target vector for the layer F6, just like the vector $\mathbf{g}$ does in an MLP (Fig. 2.3) or in an ordinary CNN (Fig. 2.8). This design has considered the shape similarity between characters with different meanings, e.g. the uppercase 'O', lowercase 'o', and zero, have a similar shape of a circle and thus will obtain similar outputs in layer F6. Lecun *et al.* argued that this design of using such distributed codes would be particularly useful in a document recognition system [96] rather than in a system for recognizing isolated digits. Based on the costs in Equation (2.25), an error function is further defined to include a weight decay factor to control the learning procedure of the network parameters [96]. Computing the gradient of the error function with respect to all the weights in all the layers of LeNet-5 is done by Backpropagation [142].

Similar with LeNet-5, most CNN systems employ more than one hidden layers to model high-level abstractions in image data. Hence CNN is a key exemplar of *deep learning* algorithms. In fact, CNN played an important role in the history of deep learning and it was the first deep model to perform well, long before arbitrary deep models were considered viable. It is the CNN that "carried the torch" for the rest of deep learning and paved the way to the acceptance of neural networks in general [16].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

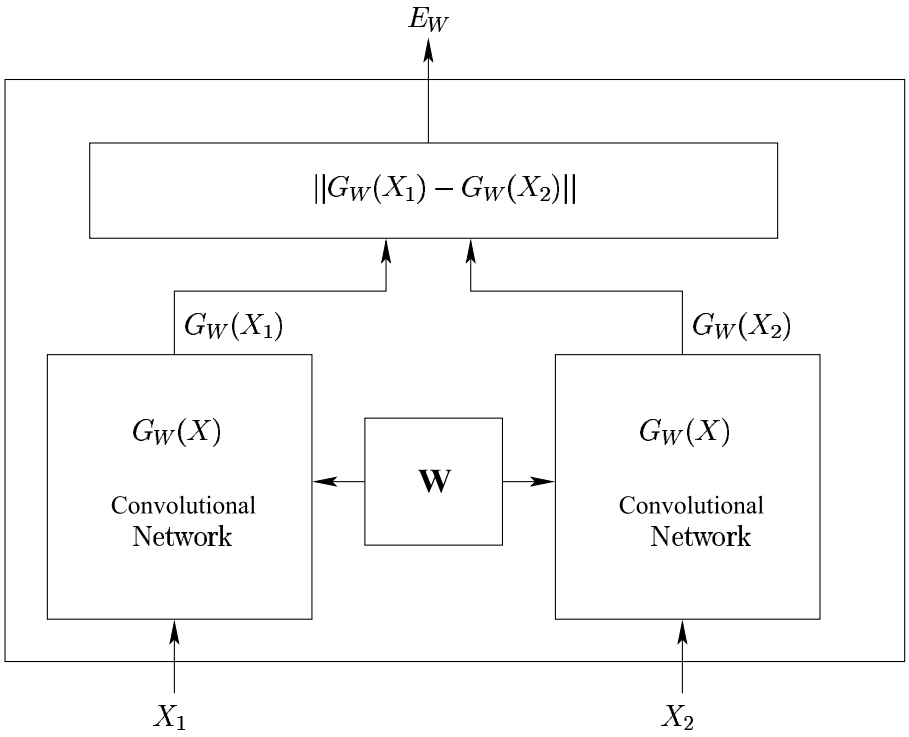


Fig. 2.10 Diagram of the Siamese CNN presented in Chopra's method [32].

2.2.5 Siamese Convolutional Neural Networks

The first works on Siamese CNN were presented around the year 1993. While Baldi and Chauvin [6] applied a Siamese CNN to verify the authenticity of two fingerprints, Bromley *et al.* independently proposed Time Delay Neural Networks (TDNN) for signature verification [24], i.e. to decide if two signatures belong to the same person. About 10 years later (2005), a more complex Siamese CNN architecture was presented by Chopra *et al.* [32] to process face images in an identity verification task, i.e. face verification. After that, similar deep Siamese CNNs were used in different applications such as dimensionality reduction [62] and video analysis [122].

Figure 2.10 shows the siamese architecture used in Chopra's method for face verification [32]. Like a Siamese MLP (Fig. 2.4), a Siamese CNN is a symmetric architecture consisting of two CNNs, where they actually share the same set of weights $\mathbf{W}$. After receiving two images $\mathbf{X}_1$ and $\mathbf{X}_2$ as the inputs, the two CNNs produce two outputs $G_W(\mathbf{X}_1)$ and $G_W(\mathbf{X}_2)$ in the target space. The distance between the two outputs is measured by the Euclidean distance:

$$E_W(\mathbf{X}_1, \mathbf{X}_2) = \|G_W(\mathbf{X}_1) - G_W(\mathbf{X}_2)\|, \tag{2.26}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

where the metric function $E_W(\mathbf{X}_1, \mathbf{X}_2)$ is also called an "energy function" in [32, 97].

Moreover, this pairwise distance in the target space is expected to approximate the "semantic" distance between the raw images: two images of the same class are supposed to yield a small distance in the target space while two images from two different classes should have a large distance. To achieve this goal, a loss function is defined based on this distance metric and the standard Backpropagation algorithm is used to learn the parameters of the CNN. The CNN architecture used in [32] mainly followed LeNet-5. Concretely, the first 6 layers of LeNet-5 were transplanted into the Siamese CNN: C1-S2-C3-S4-C5-F6, but the feature map functions of convolution and pooling were configured by different size of parameters.

Like the Siamese MLP (see Section 2.2.3), the Siamese CNN maintains the two advantages over the single-track neural networks (e.g. the standard MLP, the standard CNN): (1) there is no need to handcraft target vectors for each class, thus dimension of the target space can be arbitrarily specified; (2) the siamese architecture allows weakly supervised training since learning can be performed on data pairs instead of fully labeled data.

Apart from the Siamese MLP and the Siamese CNN, other types of neural networks can be involved in a siamese architecture to perform pairwise comparisons. For example, Nair and Hinton [123] presented Siamese Restricted Boltzmann Machines (RBM) for pairwise face verification.

2.3 Metric Learning

When referring to Siamese Neural Networks in the previous section, we mainly concerned the mapping function, i.e. which type of neural networks should be used for data mapping from the original space to the target space. We have also noticed another important factor that the cost function or the loss function plays an important role in learning a good mapping. The study on choosing a proper metric and developing a metric-based cost function is *Metric Learning*. Actually, a mapping function defines the complexity of a system and a cost function draws the objective of a system.

However, the two functions depend on each other, and an effective Metric Learning system should be a combination of them both. Concretely, specifying a metric contains the idea of learning a data mapping. Given a certain metric, a good data mapping makes the pairwise relationship between data pairs to be better measured by the same metric in the target space than in the original space. From a global view on the complete system, the chosen metric and the mapping function can be reformulated as a single function, representing a new metric for the original data, with which one can distinguish different classes easier.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

In this section, we will present different types of metrics for feature vectors and review a few Metric Learning methods in the current literature. Besides feature vectors, there are also some works on structured data such as strings, DNA sequences [12]. For readers interested in a broader scope on Metric Learning, we recommend a recent survey which has provided an up-to-date and critical review of current Metric Learning methods [13, 14].

Strictly speaking, a metric has to obey four axioms: non-negativity, identity of indiscernibles, symmetry and triangle inequality. In practice, Metric Learning algorithms may ignore one or two axioms and learn a *pseudo-metric*. For example, *Euclidean Distance* and *Cosine Similarity* are two common functions to measure the distance or similarity between two vectors, however, Euclidean Distance is a standard metric obeying the four axioms but Cosine Similarity does not have the triangle inequality property and it violates the axiom of identity of indiscernibles.

According to different metrics used on feature vectors, one can divide Metric Learning into two main families: *distance metric learning* and *similarity metric learning*. One may doubt the difference between a distance and a similarity, the Cosine Similarity is actually related to the Euclidean distance under normalized length conditions[4]. However, this relation is hard to become an equivalency because the length condition may be ignored in practical applications. Actually, the Cosine Similarity, or its relaxed variant the bilinear similarity, has been found as a particularly useful metric for text retrieval [136, 135, 29, 28] and face verification [125, 26].

Before going into detail of these Metric Learning methods, we need to declare two ways of representing a data pair: (1) using a triplet $(\mathbf{x}_i, \mathbf{y}_i, s_i)$ to denote the i_{th} pair in a data set, where $s_i = 1$ (respectively $s_i = -1$) indicates a similar (respectively dissimilar) pair; (2) using a pair $(\mathbf{x}_i, \mathbf{x}_j)$ to denote the i_{th} and j_{th} samples in a data set, forming a pair of samples, and the similarity of this pair would be indicated by additional notation. We mainly use the first style in this thesis.

2.3.1 Distance Metric Learning

As the name suggests, distance metric learning methods use a distance metric, mainly the Euclidean distance, to measure the pairwise relationship between two feature vectors. Currently, most Metric Learning methods learn a linear mapping function $f(\cdot)$ because a linear projection is more convenient to optimize and less prone to over-fitting. Assuming that $\mathbf{x}$ and $\mathbf{y}$ represent two vectors in the original space, $\mathbf{a}$ and $\mathbf{b}$ denote their linear projections in

[4] https://en.wikipedia.org/wiki/Cosine_similarity

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

the target space, the pairwise distances before and after mapping can be measured by:

$$d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\| = \sqrt{(\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})}, \tag{2.27}$$

$$d(\mathbf{a},\mathbf{b}) = \|\mathbf{W}\mathbf{x}-\mathbf{W}\mathbf{y}\| = \sqrt{(\mathbf{W}\mathbf{x}-\mathbf{W}\mathbf{y})^T(\mathbf{W}\mathbf{x}-\mathbf{W}\mathbf{y})}, \tag{2.28}$$

where $\mathbf{W}$ is the transformation matrix of the linear projection, i.e. $\mathbf{a} = f(\mathbf{x},\mathbf{W}) = \mathbf{W}\mathbf{x}$, $\mathbf{b} = f(\mathbf{y},\mathbf{W}) = \mathbf{W}\mathbf{y}$.

In some works [173, 55, 52], Equation (2.28) is written as:

$$\begin{aligned} d_{\mathbf{W}}(\mathbf{x},\mathbf{y}) &= \sqrt{(\mathbf{W}\mathbf{x}-\mathbf{W}\mathbf{y})^T(\mathbf{W}\mathbf{x}-\mathbf{W}\mathbf{y})} = \sqrt{(\mathbf{x}-\mathbf{y})^T\mathbf{W}^T\mathbf{W}(\mathbf{x}-\mathbf{y})} \\ &= \sqrt{(\mathbf{x}-\mathbf{y})^T\mathbf{A}(\mathbf{x}-\mathbf{y})} \\ &= d_{\mathbf{A}}(\mathbf{x},\mathbf{y}), \end{aligned} \tag{2.29}$$

where $\mathbf{A}$ is a positive semi-definite matrix that can be decomposed as $\mathbf{W}^T\mathbf{W}$. This distance metric $d_{\mathbf{A}}(\mathbf{x},\mathbf{y})$ is in a similar formulation with the Mahalanobis distance[5]. Thus the procedure of specifying the matrix $\mathbf{A}$ is considered as learning a Mahalanobis distance.

Generally, the objective of a distance metric learning method is to minimize the distance between a similar pair $(\mathbf{x}_i,\mathbf{y}_i) \in S$ and to separate a dissimilar pair $(\mathbf{x}_i,\mathbf{y}_i) \in D$ with a large distance, where S (respectively D) denotes a set of similar (respectively dissimilar) pairs for training. Towards this objective, researchers have proposed various cost functions and optimization algorithms to find good solutions, i.e. to specify an optimal matrix $\mathbf{A}$ or $\mathbf{W}$. Especially, specifying a positive semi-definite matrix $\mathbf{A}$ is often formulated as a convex problem that has a global optimum and can be solved by iterative algorithms based on matrix decomposition operations [173, 147, 52, 169]. In contrast, learning a linear transformation matrix $\mathbf{W}$ is probably non-convex and solved by ordinary gradient descent in the elements of the matrix $\mathbf{W}$ [55].

Mahalanobis Metric for Clustering (MMC)

The first work of Metric Learning has been presented by Xing *et al.* [173], namely, Mahalanobis Metric for Clustering (MMC). The cost function of MMC is defined as:

$$J = \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in S} d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i), \quad s.t. \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} d_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) > 1. \tag{2.30}$$

[5] https://en.wikipedia.org/wiki/Mahalanobis_distance

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

The objective of this cost function is to minimize the Euclidean distance between a similar pair and to make the distance between a dissimilar pair larger than 1. This cost function has been proven to be *convex*, which enables the authors to derive efficient algorithm to find a global optimal solution [173]. A simple optimization algorithm was proposed, relying on iterative eigenvalue decompositions.

This method is straightforward to follow and can be efficiently computed; but relying on matrix decompositions constrains it to only small dimensional problems because the decomposition of high dimensional matrix is computationally expensive. Moreover, the MMC method focuses on batch learning (i.e. offline learning), where the whole training set must be ready as a batch.

Pseudo-metric Online Learning Algorithm (POLA)

Pseudo-metric Online Learning Algorithm (POLA) [147] is the first work designed for online learning, i.e. having only one or a few samples for training at each iteration. Based on the idea of MMC, POLA introduces a margin factor, the minimum separation between all pairs of similar and dissimilar samples, to the cost function. Concretely, the width of the margin is 2 and a threshold b is defined in two constraints:

- for a similar pair $(\mathbf{x}_i, \mathbf{y}_i) \in S$, the distance between them should be no more than $b-1$: $d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{y}_i) \leq b-1$;
- for a dissimilar pair $(\mathbf{x}_i, \mathbf{y}_i) \in D$, the pairwise distance should be no less than $b+1$: $d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{y}_i) \geq b+1$.

Let $s_i = 1$ denote a similar pair and $s_i = -1$ denote a dissimilar pair, the two constraints can be rewritten as a single constraint:

$$s_i(b - d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{y}_i)) \geq 1. \tag{2.31}$$

And the cost function is an adaptation of the hinge loss,

$$J_i = max(0,\ 1 + s_i(d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{y}_i) - b)), \tag{2.32}$$

where J_i is the cost of the i_{th} sample pair during online learning. Like MMC, POLA relies on matrix decomposition to perform gradient descent, thus it is hard to handle high dimensional problems. Moreover, its perfect assumption of separating all the data with an absolute margin 2 limits its application in practice [13].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

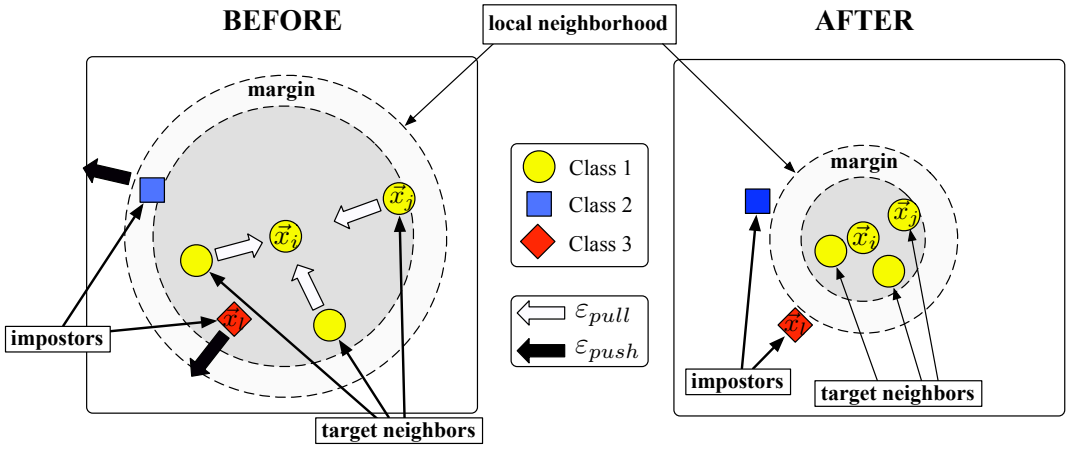


Fig. 2.11 Schematic illustration of one input's neighborhood before training (left) vs. after training (right) for the LMNN method. The distance metric is optimized so that: (1) its K=3 target neighbors lie within a smaller radius after training; (2) differently labeled inputs lie outside this smaller radius by some finite margin. Bold arrows in the left picture indicate the directions of gradients for different neighbors [169].

Large Margin Nearest Neighbors (LMNN)

When POLA has an overcritical assumption of separating all the training pairs with a margin, Large Margin Nearest Neighbors (LMNN), introduced by Weinberger *et al.* [167, 169], pushes the constraints to a few selected training data only. In addition, apart from training on data pairs $(\mathbf{x}_i, \mathbf{x}_j)$, LMNN learns the distance metric also on triplets of data samples, i.e. a group of three samples $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in T$ which includes a similar pair $(\mathbf{x}_i, \mathbf{x}_j) \in S$ and a dissimilar pair $(\mathbf{x}_i, \mathbf{x}_k) \in D$. Usually, with respect to the similar pair, the isolated sample $\mathbf{x}_k$ in the triplet is called an imposter.

Concretely, the cost function of LMNN is composed of two items, a *pulling* force and a *pushing* force:

- *Pulling:* to pull each similar pair closer,

$$J_{pull} = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in S} d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j), \tag{2.33}$$

  note that unlike other methods usually take all possible similar pairs in training, the set S here only contains similar pairs in *local neighborhoods*: for each training sample $\mathbf{x}_i$, it is paired with its K nearest neighbors only (e.g. $K = 3$ in Fig. 2.11).

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

- *Pushing:* to push an imposter far away from the other two similar samples in a triplet, the distance between the two samples should be less than the distance between the imposter and the two samples, i.e. for a triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in T$, we have

$$d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) + 1 \leq d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_k), \tag{2.34}$$

  hence the cost of the pushing part is:

$$J_{push} = \sum_{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in T} max(0,\ d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) + 1 - d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_k)), \tag{2.35}$$

  remind that the function $max(0,\ \cdot)$ denotes the hinge loss function that POLA also used (Equation (2.32)). Minimizing the pushing part aims to move the imposter (the red and blue points in Fig. 2.11) from the neighborhood of $\mathbf{x}_i$.

Combining the two parts J_{pull} and J_{push} has competing effects – to attract target neighbors on one hand and to repel impostors on the other hand (see Fig. 2.11). A weighting coefficient $\mu \in [0, 1]$ balances this goal:

$$J = (1 - \mu) J_{pull} + \mu J_{push}. \tag{2.36}$$

The authors provided a solver based on sub-gradient descent and eigenvalue decomposition to minimize the above cost function. In summary, LMNN involves two main ideas to learn a distance metric: (1) besides data pairs, introducing data triplets into training; (2) in order to perform local optimization, selecting data pairs S and triplets T by *enclosing target neighbors*. Taking advantage of the two ideas, LMNN generally performs very well in practice for supervised classification and clustering problems.

Neighborhood Component Analysis (NCA)

Different from the above methods directly optimize the pairwise distances, Neighborhood Component Analysis (NCA) [55] introduces an idea of transforming the pairwise Euclidean distance to a probability of being neighbors. In the target space, for a pair of vectors $\mathbf{W}\mathbf{x}_i = f(\mathbf{x}_i, \mathbf{W})$ and $\mathbf{W}\mathbf{x}_j = f(\mathbf{x}_j, \mathbf{W})$, the Euclidean distance between them is $d_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j)$ (see Equation (2.28)). NCA defines the probability of the two vector being neighbors as:

$$p_{ij} = \frac{e^{-d^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j)}}{\sum_{k \neq i} e^{-d^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_k)}}, \quad p_{ii} = 0. \tag{2.37}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

And the probability that the i_{th} vector will be correctly classified is:

$$p_i = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in S} p_{ij}, \tag{2.38}$$

where $(\mathbf{x}_i, \mathbf{x}_j) \in S$ includes all the vectors in the same class with $\mathbf{x}_i$. The objective of NCA is to maximize the expected number of correctly classified points. Two candidates for the objective function have been proposed:

$$E = \sum_{i=1}^{N} p_i \quad or \quad E = \sum_{i=1}^{N} ln(p_i), \tag{2.39}$$

where N is the number of points in the training set. Note that maximizing an objective function is equivalent to minimizing its reverse cost function $-E$. Efficiently computed gradient functions $\frac{\partial E}{\partial \mathbf{W}}$ have also been provided in [55] to perform *ordinary gradient descent* (the same as we used for MLP training, see Section 2.2.2).

Compared with MMC and POLA, NCA learns the transformation matrix $\mathbf{W}$ and does not rely on matrix decomposition, thus it can handle large dimensional problems. However, NCA performs non-convex optimization as it uses the exponential function to transform an Euclidean distance to a probability, so its solution may be a local optimum. Besides, NCA needs to hold the whole training set at the very beginning to calculate the pairwise probability p_{ij}, thus it is hardly intractable for online learning, i.e. knowing only one or a few training samples at each iteration.

Maximally Collapsing Metric Learning (MCML)

Shortly after NCA, Globerson and Roweis [52] proposed an alternative convex formulation to deal with the probability p_{ij} (Equation (2.37)). Their method is called Maximally Collapsing Metric Learning (MCML), as they have made a strong assumption on collapsing classes:

- the distance between a similar pair is ideally to be 0, the ideal probability of a similar pair being together is 1: $s_{ij} = 1$;
- the distance between a dissimilar pair should be infinite, so the ideal probability of a dissimilar pair being well separated is 0: $s_{ij} = 0$.

Under this assumption, they employed the Kullback-Leibler (KL) divergence to match the computed probability of each pair to its ideal target:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{N} KL(s_{ij}|p_{ij}) = \sum_{i=1}^{N} \sum_{j=1}^{N} s_{ij} ln(\frac{s_{ij}}{p_{ij}}). \tag{2.40}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

If and only if p_{ij} equals s_{ij}, the yielded cost can be 0. Note that the dissimilar pairs have no contribution to the cost since s_{ij} is set to 0, so this equation can be rewritten as:

$$J = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S} -ln(p_{ij}) = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S} -ln(\frac{e^{-d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j)}}{\sum_{k\neq i} e^{-d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_k)}}), \tag{2.41}$$

which can be separated into two parts as:

$$J = \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in S} d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j) + \sum_{i=1}^{N} ln(\sum_{k\neq i} e^{-d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_k)}), \tag{2.42}$$

where minimizing the first part reduces the distance between similar points, and minimizing the second part enlarges the distance between points from different classes.

Especially, the matrix $\mathbf{A} = \mathbf{W}^T\mathbf{W}$ is positive semi-definite and the cost function is convex: the first part is naturally convex as it involves Euclidean distance only; the second part is a *log-sum-exp* function of affine functions of the matrix $\mathbf{A}$ and is therefore also convex [23]. An iterative optimization algorithm based on eigenvalue decomposition was presented to find the global optimal solution for such a convex cost function [52].

Information-Theoretic Metric Learning (ITML)

Continuing the study on probability inference, Davis *et al.* [39] proposed Information-Theoretic Metric Learning (ITML) by incorporating the *LogDet divergence* into the cost function as a regularization factor[6]. Instead of measuring the probability of a pair being similar in NCA and MCML, ITML parameterizes the probability of a sample $\mathbf{x}$ belonging to a set of multivariate Gaussian distribution:

$$p(\mathbf{x},\mathbf{A}) = \frac{1}{Z} e^{-\frac{1}{2}d^2_{\mathbf{A}}(\mathbf{x},\boldsymbol{\mu})}, \tag{2.43}$$

where $\boldsymbol{\mu}$ is the mean of the Gaussian distribution, Z is a normalizing constant and $\mathbf{A}^{-1}$ can be regarded as the covariance matrix of the distribution. In practice, to prevent over-fitting, a parameter matrix $\mathbf{A}$ is initialized with a predefined matrix $\mathbf{A}_0$ and is constrained to be close to it during learning. The "closeness" of the two matrices is measured by the KL divergence

[6]Regularization refers to a process of introducing additional information in order to solve an ill-posed problem or to prevent over-fitting. Especially, the L2-norm regularization (i.e. the Frobenius norm) is also called "weight decay", in particular in the setting of neural networks

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

of the two Gaussians parameterized by $\mathbf{A}_0$ and $\mathbf{A}$:

$$KL(p(\mathbf{x},\mathbf{A}_0)|p(\mathbf{x},\mathbf{A})) = \int p(\mathbf{x},\mathbf{A}_0) log \frac{p(\mathbf{x},\mathbf{A}_0)}{p(\mathbf{x},\mathbf{A})} d\mathbf{x}, \tag{2.44}$$

which can be represented as the *LogDet divergence*:

$$KL(p(\mathbf{x},\mathbf{A}_0)|p(\mathbf{x},\mathbf{A})) = \frac{1}{2} D_{ld}(\mathbf{A},\mathbf{A}_0) = tr(\mathbf{A}\mathbf{A}_0^{-1}) - log(det(\mathbf{A}\mathbf{A}_0^{-1})) - d, \tag{2.45}$$

where d is the dimension of the input $\mathbf{x}$. Note that if and only if $\mathbf{A} = \mathbf{A}_0$, the above equation results in the minimum 0.

As usual, the objective of ITML is to make the distance between a similar pair smaller and to make the distance between a dissimilar pair larger. Formally, two thresholds u and l are specified to restrict the distances:

$$\begin{cases} d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) \leq u & (\mathbf{x}_i,\mathbf{y}_i) \in S, \\ d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) \geq l & (\mathbf{x}_i,\mathbf{y}_i) \in D, \end{cases} \quad (u \leq l), \tag{2.46}$$

In some cases, there may not exist a feasible solution that perfectly fits all the training data. To prevent such a scenario from occurring, slack variables are incorporated into the formulation:

$$\begin{aligned} & J = \sum_{(\mathbf{x}_i,\mathbf{y}_i) \in S \bigcup D} \varepsilon_i, \\ & s.t. \quad \begin{cases} d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) \leq u + \varepsilon_i & (\mathbf{x}_i,\mathbf{y}_i) \in S, \\ d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) \geq l - \varepsilon_i & (\mathbf{x}_i,\mathbf{y}_i) \in D, \end{cases} \quad (u \leq l), \end{aligned} \tag{2.47}$$

Combining the above equations with the *LogDet divergence* regularization factor, here is the final cost function of ITML:

$$\begin{aligned} & J = \sum_{(\mathbf{x}_i,\mathbf{y}_i) \in S \bigcup D} \varepsilon_i + \lambda D_{ld}(\mathbf{A},\mathbf{A}_0), \\ & s.t. \quad \begin{cases} d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) \leq u + \varepsilon_i & (\mathbf{x}_i,\mathbf{y}_i) \in S, \\ d^2_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) \geq l - \varepsilon_i & (\mathbf{x}_i,\mathbf{y}_i) \in D, \end{cases} \quad (u \leq l), \end{aligned} \tag{2.48}$$

In practice, $\mathbf{A}_0$ is often set to $\mathbf{I}$ (the identity matrix) and thus the regularization aims at keeping the learned distance close to the Euclidean distance. The key feature of the LogDet divergence is that it is finite if and only if $\mathbf{A}$ is positive semi-definite. In other words, minimizing the cost function provides an automatic and cheap way of keeping $\mathbf{A}$ to be positive semi-definite. Therefore, this method does not require costly eigenvalue computations or semi-definite

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

programming. Benefiting form this advantage, the LogDet divergence has been adopted in many other works on Metric Learning [78, 137].

Logistic Discriminative Metric Learning (LDML)

Guillaumin *et al.* [60] proposed Logistic Discriminative Metric Learning (LDML) to model the probability of two vectors being similar by using *the sigmoid function* (Section 2.2.1). Let a triplet $(\mathbf{x}_i, \mathbf{y}_i, s_i)$ denote the i_{th} pair in the training set, where $s_i = 1$ (respectively $s_i = 0$) indicates a similar pair (respectively a dissimilar pair), the probability of $\mathbf{x}_i$ and $\mathbf{y}_i$ being similar is parameterized by:

$$p_i = \frac{1}{1 + e^{(d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{y}_i) - b)}}, \tag{2.49}$$

where b is a bias scalar. We can see that the larger the pairwise distance $d^2_{\mathbf{A}}(\mathbf{x}_i, \mathbf{y}_i)$ is, the smaller the probability is. The cost function is simply the inverse maximum log-likelihood of all possible training pairs:

$$J = -\sum_{i=1}^{N} [s_i ln(p_i) + (1 - s_i) ln(1 - p_i)], \tag{2.50}$$

where N is the number of training pairs. This equation is smooth and convex, the maximum likelihood estimations for the matrix $\mathbf{A}$ and the bias b are obtained by using a projected gradient method [19]. In practice, LDML has shown its effectiveness on the problem of pairwise face verification [60].

2.3.2 Similarity Metric Learning

Besides distances, similarities are another kind of metrics widely used to measure pairwise relationship between two feature vectors. Actually, a similarity metric is a *pseudo-metric* as it may violate the axiom of the triangle inequality. However, for metric learning applications, similarities were preferred over distances in many practical situations, such as information retrieval [5] and document analysis [152, 172].

Like the Mahalanobis distance, a general similarity metric can be also parameterized by a matrix $\mathbf{A}$. For any two vectors $\mathbf{x}$ and $\mathbf{y}$ in the original space, the similarity between their projections in the target space is measured by:

$$s_{\mathbf{A}}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{A} \mathbf{y}}{N(\mathbf{x}, \mathbf{y})}, \tag{2.51}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

when the matrix $\mathbf{A}$ is supposed to be positive semi-definite, we have $\mathbf{A} = \mathbf{W}^T\mathbf{W}$, and the matrix $\mathbf{W}$ acts as a linear transformation matrix that maps the original space (e.g. $\mathbf{x}$) to the target space (e.g. $\mathbf{a} = \mathbf{W}\mathbf{x}$):

$$s_{\mathbf{W}}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^T\mathbf{W}^T\mathbf{W}\mathbf{y}}{N(\mathbf{x},\mathbf{y})} = \frac{(\mathbf{W}\mathbf{x})^T\mathbf{W}\mathbf{y}}{N(\mathbf{x},\mathbf{y})}, \tag{2.52}$$

where $N(\mathbf{x},\mathbf{y})$ in the two equations is a normalization term to map the similarity function to a particular interval, e.g. [-1,1]. Specifically, when $N(\mathbf{x},\mathbf{y}) = 1$, $s_{\mathbf{W}}(\mathbf{x},\mathbf{y})$ is the bilinear similarity function [29]; when $N(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{W}\mathbf{x})^T\mathbf{W}\mathbf{x}}\sqrt{(\mathbf{W}\mathbf{y})^T\mathbf{W}\mathbf{y}}$, $s_{\mathbf{W}}(\mathbf{x},\mathbf{y})$ is the Cosine Similarity function [125]. Generally, the objective of a similarity metric learning method is to increase the similarity between a similar pair and to decrease the similarity between a dissimilar pair.

Similarity Learning Algorithm (SiLA)

Similarity Learning Algorithm (SiLA) is the first work proposed by Qamar *et al.* [136] to learn a similarity metric for k-Nearest Neighbors (kNN) classification. They have presented the general formulation of similarity metric learning (Equation (2.51)), but they actually restricted themselves to the Cosine Similarity in the experimental evaluations.

SiLA does not impose any constraint on the matrix $\mathbf{A}$, it is not required to be positive, or even symmetric. Instead, the only constraint on $\mathbf{A}$ is that its Frobenius norm equals 1, i.e. $\|\mathbf{A}\| = 1$. Similar with LMNN (Section 2.3.1), SiLA also learns a metric from triples. Remind that a triplet $(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k) \in T$ comprises a pair of similar vectors $(\mathbf{x}_i,\mathbf{x}_j) \in S$ and a dissimilar pair $(\mathbf{x}_i,\mathbf{x}_k) \in D$. With respect to the similar pair $(\mathbf{x}_i,\mathbf{x}_j)$, the isolated sample $\mathbf{x}_k$ is called an imposter or an outlier.

The cost for the i_{th} sample is defined as a hinge loss:

$$J_i = max(0, \gamma - (\sum_{(\mathbf{x}_i,\mathbf{x}_j,\mathbf{x}_k)\in T_i} s_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_j) - s_{\mathbf{A}}(\mathbf{x}_i,\mathbf{x}_k))), \tag{2.53}$$

where the set of triplets T_i is composed of selected nearest neighbors of the i_{th} sample (like target neighbors in LMNN), and the positive constant γ is the margin separating the similar pairs from the outliers. The overall cost for all the samples is:

$$J = \sqrt{\sum_{i=1}^{N} J_i^2}, \tag{2.54}$$

where N is the number of all training samples.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Different from LMNN relying on sub-gradient descent and eigenvalue decomposition for an offline optimization, SiLA optimizes the similarity metric with an online learning algorithm based on voted perceptrons [48]. Compared with several distance metric learning methods such as LMNN and ITML, SiLA has demonstrated that the Cosine Similarity is preferred over the Euclidean distance on several data collections such as Iris and Balance from the UCI dataset [20].

Generalized Cosine Learning Algorithm (gCosLA)

The same authors of SiLA, Qamar and Gaussier, proposed Generalized Cosine Learning Algorithm (gCosLA) [135], which works directly on the Cosine Similarity. But different from SiLA, gCosLA learns a positive semi-definite matrix $\mathbf{A}$. Remind that for any two vectors $\mathbf{x}$ and $\mathbf{y}$ in the original space, the pairwise Cosine Similarity metric between them is defined as:

$$s_{\mathbf{A}}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^T\mathbf{A}\mathbf{y}}{\sqrt{\mathbf{x}^T\mathbf{A}\mathbf{x}}\sqrt{\mathbf{y}^T\mathbf{A}\mathbf{y}}}. \tag{2.55}$$

The basic assumption of gCosLA is that the similarity between a similar pair should be always larger than the similarity between a dissimilar pair. Formally, assuming that we are given three vectors $\mathbf{x},\mathbf{y},\mathbf{z}$, where $\mathbf{x}$ and $\mathbf{y}$ are similar, but dissimilar with $\mathbf{z}$, a margin of width 2γ is defined to separate them:

$$s_{\mathbf{A}}(\mathbf{x},\mathbf{y}) - s_{\mathbf{A}}(\mathbf{x},\mathbf{z}) \geq 2\gamma. \tag{2.56}$$

By introducing another threshold b, the equation can be rewritten as two constraints for similar pairs $(\mathbf{x}_i,\mathbf{y}_i) \in S$ and dissimilar pairs $(\mathbf{x}_i,\mathbf{y}_i) \in D$, respectively:

$$\begin{aligned} s_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) &\geq b+\gamma, \quad (\mathbf{x}_i,\mathbf{y}_i) \in S, \\ s_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i) &\leq b-\gamma, \quad (\mathbf{x}_i,\mathbf{y}_i) \in D. \end{aligned} \tag{2.57}$$

If we use $l_i = 1$ (respectively $l_i = -1$) to denote that a pair of data $(\mathbf{x}_i,\mathbf{y}_i)$ being similar (respectively dissimilar), we can write a single-line constraint as:

$$l_i(b - s_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i)) + \gamma \leq 0, \tag{2.58}$$

and the cost function of gCosLA is:

$$J = \sum_{i=1}^{N} max(0, l_i(b - s_{\mathbf{A}}(\mathbf{x}_i,\mathbf{y}_i)) + \gamma), \tag{2.59}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

where the function $max(0, \cdot)$ denotes the hinge loss function and N is the number of training pairs. The authors followed POLA and proposed both online and batch learning algorithms to solve the above minimization problem. Compared with SiLA, gCosLA is generally more accurate on classification problems.

Online Algorithm for Scalable Image Similarity (OASIS)

Chechik *et al.* [28, 29] proposed an Online Algorithm for Scalable Image Similarity (OASIS) that learns a bilinear similarity with a focus on large-scale problems. Formally, the normalization term in Equation (2.51) is set to 1, and the bilinear similarity between two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ is:

$$s_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{A} \mathbf{x}_j. \tag{2.60}$$

OASIS shares two common issues with SiLA: (1) the matrix $\mathbf{A}$ is not required to be positive semi-definite; (2) the learning procedure is performed on triplets of vectors. For a triplet $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in T$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, but dissimilar with $\mathbf{x}_k$, a constraint is defined as:

$$s_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) - s_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_k) \geq 1, \tag{2.61}$$

and the cost function is simply the sum of the hinge loss for each triplet:

$$J = \sum_{(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k) \in T} max(0, 1 - s_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_j) + s_{\mathbf{A}}(\mathbf{x}_i, \mathbf{x}_k)). \tag{2.62}$$

This equation is similar with the pushing part of the LMNN method [169] (Section 2.3.1).

In order to minimize this loss, OASIS employs the Passive-Aggressive algorithm [35] iteratively to optimize the matrix $\mathbf{A}$. Experiments comparing OASIS with distance metric learning methods such as LMNN demonstrated the superiority of learning the bilinear similarity over learning a distance metric for the problem of image retrieval. Unfortunately, the authors did not compare OASIS with SiLA or any other similarity metric learning methods. However, since the bilinear similarity has no need to compute a normalization term (Equation (2.51)), we believe that OASIS is naturally efficiently computable.

Cosine Similarity Metric Learning (CSML)

Besides SiLA and gCosLA, Cosine Similarity Metric Learning (CSML) [125] is another method focusing on learning a Cosine Similarity metric. Different from SiLA and gCosLA that optimize the matrix $\mathbf{A}$, CSML explicitly learns the transformation matrix $\mathbf{W}$ (note that

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

$\mathbf{W}^T\mathbf{W} = \mathbf{A}$). Firstly, the Cosine Similarity metric between a pair of vectors $(\mathbf{x}_i, \mathbf{y}_i)$ is:

$$s_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i) = \frac{(\mathbf{W}\mathbf{x})_i^T \mathbf{W}\mathbf{y}_i}{\sqrt{(\mathbf{W}\mathbf{x})_i^T \mathbf{W}\mathbf{x}_i}\sqrt{(\mathbf{W}\mathbf{y})_i^T \mathbf{W}\mathbf{y}_i}} = \frac{(\mathbf{W}\mathbf{x})_i^T \mathbf{W}\mathbf{y}_i}{\|\mathbf{W}\mathbf{x}_i\|\|\mathbf{W}\mathbf{y}_i\|}. \tag{2.63}$$

The cost function of CSML is defined as:

$$J = -\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in S} s_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i) + \alpha \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} s_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i) + \beta\|\mathbf{W} - \mathbf{W}_0\|^2. \tag{2.64}$$

This cost function intuitively presents the objective of CSML:

- Minimizing the first part $-\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in S} s_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i)$ aims to increase the similarity between each similar pair.
- Minimizing the second part $\alpha\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} s_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i)$ aims to decrease the similarity between each dissimilar pair, the coefficient α weights the contribution from the dissimilar pairs, with respect to the contribution from the similar pair (the first part).
- The third part $\beta\|\mathbf{W} - \mathbf{W}_0\|$ acts as a regularization factor, like the *LogDet divergence* regularization factor in ITML (Section 2.3.1), it restricts the learned matrix $\mathbf{W}$ close to a predefined matrix $\mathbf{W}_0$. The coefficient β weights its effect to the whole cost function. In practice, this part guarantees the CSML algorithm to obtain a better matrix $\mathbf{W}$ than $\mathbf{W}_0$ to specify the pairwise similarity.

CSML was applied for pairwise face verification and a batch gradient descent algorithm, i.e. the Conjugate Gradient Descent (CGD) [115], was adopted to optimize the matrix $\mathbf{W}$. The gradient of the cost function is calculated by:

$$\frac{\partial J}{\partial \mathbf{W}} = -\sum_{(\mathbf{x}_i,\mathbf{y}_i)\in S} \frac{\partial s_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i)}{\partial \mathbf{W}} + \alpha \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in D} \frac{\partial s_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i)}{\partial \mathbf{W}} + \beta(\mathbf{W} - \mathbf{W}_0), \tag{2.65}$$

where

$$\frac{\partial s_{\mathbf{W}}(\mathbf{x}_i, \mathbf{y}_i)}{\partial \mathbf{W}} = \frac{1}{\|\mathbf{W}\mathbf{x}_i\|\|\mathbf{W}\mathbf{y}_i\|}[(\frac{(\mathbf{W}\mathbf{x}_i)^T\mathbf{W}\mathbf{y}_i}{\|\mathbf{W}\mathbf{x}_i\|^2}\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{y}_i)\mathbf{x}_i^T + (\frac{(\mathbf{W}\mathbf{x}_i)^T\mathbf{W}\mathbf{y}_i}{\|\mathbf{W}\mathbf{y}_i\|^2}\mathbf{W}\mathbf{y}_i - \mathbf{W}\mathbf{x}_i)\mathbf{y}_i^T]. \tag{2.66}$$

The proof of this partial derivative of the Cosine Similarity function is referred to Appendix A. Note that CSML performs non-convex optimization resulting in a local optimal solution, however, experiments in [125] showed that CSML achieved competitive performance on the problem of face verification.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

2.3.3 Other Advances in Metric Learning

Besides the above methods mainly based on the Mahalanobis distance or the Cosine Similarity, many interesting advances have been made to Metric Learning. For example, investigating other metrics such as the χ^2-distance [82] and the Fisher information distance [165] for measuring feature vectors; involving both distance metric learning and similarity metric learning in a single task [26]; examining the effect of regularization factors such as LogDet divergence regularization [39] and Fantope regularization [90] in metric learning for robust classification; apart from the usual pairwise and triplet constraints, introducing quadruplet-wise (Qwise) constraints [89] to exploit fine data relationship such as class ranking; studying nonlinear transformations instead of the common used linear matrices in the above methods [62]; or developing specific metrics for other kind of data such as histogram data [82] and strings [12].

Relevant Component Analysis (RCA)

Relevant Component Analysis (RCA) [148, 7] provides a simple way to specify a transformation matrix $\mathbf{W}$, which performs well in both distance metric learning and similarity metric learning. Firstly, the proposed RCA algorithm computes the *within chunklet covariance matrix* for a set of training data:

$$\mathbf{C} = \frac{1}{N}\sum_{j=1}^{n}\sum_{i=1}^{n_j}(\mathbf{x}_{ji}-\boldsymbol{\mu}_j)(\mathbf{x}_{ji}-\boldsymbol{\mu}_j)^T, \tag{2.67}$$

where N is the number of all samples, n is the number of chunklets, n_j counts the number of samples in the j_{th} chunklet, and $\boldsymbol{\mu}_j$ denotes the mean of the j_{th} chunklet. It is noteworthy that the idea of "within chunklet" involves similar pairs only for learning a metric. By applying Cholesky decomposition [132] or eigenvalue decomposition [26], the transformation matrix is simply $\mathbf{W} = \mathbf{C}^{-\frac{1}{2}}$.

On one hand, Bar-Hillel *et al.* [7] has proven that RCA gives an optimal solution to the problem of minimizing within class Euclidean distances. On the other hand, under Gaussian assumptions, RCA can be interpreted as the maximum-likelihood estimator of the *within class covariance matrix* [65, 64], which is usually combined with the Cosine Similarity metric to measure the similarities between data [26, 8]. In general, RCA has been empirically justified to be effective in learning either the Euclidean distance metric [7] or the Cosine Similarity metric [26, 8].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Dimensionality Reduction by Learning an Invariant Mapping (DrLIM)

Dimensionality Reduction by Learning an Invariant Mapping (DrLIM) [62] proposed to learn a nonlinear distance metric in a Siamese CNN architecture (Section 2.2.5), where the so called "energy function" (i.e. a specific kind of cost function) was indeed developed on the Euclidean distance. Concretely, the parameter set $\mathbf{W}$ that configures the distance metric is no longer a single matrix for a linear mapping, but a set of transformation matrices and bias vectors that realizes a nonlinear mapping:

$$d_{\mathbf{W}}(\mathbf{x},\mathbf{y}) = \|f(\mathbf{x},\mathbf{W}) - f(\mathbf{y},\mathbf{W})\| = \sqrt{(f(\mathbf{x},\mathbf{W}) - f(\mathbf{y},\mathbf{W}))^T (f(\mathbf{x},\mathbf{W}) - f(\mathbf{y},\mathbf{W}))}. \quad (2.68)$$

Let $s_i = 1$ (respectively $s_i = 0$) denote a similar pair (respectively a dissimilar pair), the cost of the i_{th} training pair is:

$$J_i = s_i \frac{1}{2} d^2_{\mathbf{W}}(\mathbf{x}_i,\mathbf{y}_i) + (1-s_i)\frac{1}{2} max^2(0, m - d_{\mathbf{W}}(\mathbf{x}_i,\mathbf{y}_i)), \quad (2.69)$$

where m defines a margin in the hinge loss function. Similar with other distance metric learning methods (Section 2.3.1), we can see that the first half of this equation aims to minimize the distance between a similar pair in the target space, and the second half aims to separate a dissimilar pair with a margin m. Being a Siamese NN, the DrLIM method can be efficiently trained via the Backpropagation algorithm [142] in an online mode (Section 2.2.2).

Multiple Metrics Large Margin Nearest Neighbor (M^2-LMNN)

All the above distance or similarity metric learning methods learn a single matrix $\mathbf{A}$ or $\mathbf{W}$ for a certain problem, Multiple Metric Large Margin Nearest Neighbor (M^2-LMNN) [168, 169] pioneered the study of learning multiple metrics via a single formulation. Formally, for any two given vectors $\mathbf{x}$ and $\mathbf{y}$, several Mahalanobis distances may be specified:

$$d_{\mathbf{A}_i}(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T \mathbf{A}_i (\mathbf{x}-\mathbf{y})}, \quad i = 1,\ldots,K, \quad (2.70)$$

where K defines the number of metrics that need to be learned. In practice, these K learned distances $d_{\mathbf{A}_1}(\mathbf{x},\mathbf{y}),\ \ldots,\ d_{\mathbf{A}_K}(\mathbf{x},\mathbf{y})$ can be *sorted* or *averaged* as the final measurement of distance, in order to determine nearest neighbors and label test samples. Although introducing more metrics into learning leads to a higher computational cost, M^2-LMNN significantly improved the classification performance over standard LMNN that learns a single metric. Besides LMNN, other Metric Learning methods have also benefited from learning multiple metrics [131, 36, 71].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Similarity Metric Learning over the Intra-personal Subspace (Sub-SML)

When classical methods chose to learn either a distance metric or a similarity metric, Cao *et al.* [26] proposed to *simultaneously* learn a Mahalanobis distance metric and a bilinear similarity metric for the problem of face verification. They call their method Sub-SML, short for Similarity Metric Learning over the Intra-personal Subspace.

Concretely, the distance is parameterized by a matrix $\mathbf{A}$ as $d_{\mathbf{A}}(\mathbf{x},\mathbf{y})$ (Equation (2.28)) and the bilinear similarity is parameterized by another matrix $\mathbf{B}$ as $s_{\mathbf{B}}(\mathbf{x},\mathbf{y})$ (Equation (2.60)). A *generalized similarity* containing them both is defined as:

$$g_{(\mathbf{A},\mathbf{B})}(\mathbf{x},\mathbf{y}) = s_{\mathbf{B}}(\mathbf{x},\mathbf{y}) - d_{\mathbf{A}}(\mathbf{x},\mathbf{y}). \tag{2.71}$$

Let $l_i = 1$ (respectively $l_i = -1$) denote that a pair of vectors $(\mathbf{x}_i,\mathbf{y}_i)$ being similar (respectively dissimilar), the cost of this pair is defined as a hinge loss:

$$J_i = max(0, 1 - l_i g_{(\mathbf{A},\mathbf{B})}(\mathbf{x}_i,\mathbf{y}_i)), \tag{2.72}$$

which can be explained as two constraints:

- when $l_i = 1$, we have $1 - g_{(\mathbf{A},\mathbf{B})}(\mathbf{x}_i,\mathbf{y}_i) \leq 0$, which means that the generalized similarity of a similar pair should be larger than 1;
- when $l_i = -1$, we have $1 + g_{(\mathbf{A},\mathbf{B})}(\mathbf{x}_i,\mathbf{y}_i) \leq 0$, which means that the generalized similarity of a dissimilar pair should be less than -1.

In other words, a default margin of width 2 (from -1 to 1) is set. Combining the cost of all training pairs with a regularization factor based on Frobenius norms, the cost function of Sub-SML is

$$J = \sum_{i=1}^{N} max(0, 1 - l_i g_{(\mathbf{A},\mathbf{B})}(\mathbf{x}_i,\mathbf{y}_i)) + \frac{\lambda}{2}(\|\mathbf{A}-\mathbf{I}\|^2 + \|\mathbf{B}-\mathbf{I}\|^2), \tag{2.73}$$

where N is the number of all training pairs and $\mathbf{I}$ is the identity matrix. The formulation is a convex optimization problem that guarantees the existence of its global solution. Moreover, the authors provided the *dual formulation* of the Sub-SML cost which can be efficiently solved by an accelerated gradient-based algorithm [10].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

2.4 Conclusion and Open Problems

In this chapter, we have reviewed the literature of Siamese Neural Networks and Metric Learning. Generally, the study of Siamese Neural Networks focuses on designing a good architecture of neural networks, i.e. specifying a mapping function. In contrast, Metric Learning emphasizes learning a good metric, i.e. formulating a cost function. However, Siamese Neural Networks and Metric Learning are not isolated from each other. On one hand, a Siamese NN requires a metric-based cost function, e.g. on the Euclidean distance in [32, 62]. On the other hand, most Metric learning methods take a linear mapping function as the default setting [173, 147, 169, 55, 52, 39, 60, 136, 135, 29, 125, 7, 26].

Therefore, we regard Siamese Neural Networks and Metric Learning as *a unifying study* of designing a good architecture to learn a good metric. So far, we have known many exemplars in this chapter: for a mapping function, the depth of the architecture can be shallow or deep, namely, shallow linear transformations (most methods in Section 2.3), multi-layer neural networks (e.g. the MLP in Section 2.2.3) or deep neural networks (e.g. the CNN in Section 2.2.5); for a cost function, we have a lot of candidates based on either a distance metric (Section 2.3.1) or a similarity metric (Section 2.3.2).

Can we build up a general system with these components, providing perfect solutions to all the practical problems? Apparently, the answer to this question would be negative since most problems are data-driven and the effectiveness of a method is mainly verified by empirical evaluations. For designing a good Metric Learning system, a few problems remain open.

A mapping function: linear vs. nonlinear

The selection of a linear or nonlinear function for a problem reveals the trade-off between *under-fitting* and *over-fitting* to the given training data. Problems in a "well-represented" feature space are linearly separable, so a linear function has less risk of over-fitting than the nonlinear one. However, when it is difficult to craft discriminative feature representations for a problem, the linearity also limits the function's capacity of realizing complex mappings and thus under-fitting occurs, i.e. the linear model can not capture the underlying trend of the training data.

Over-fitting is always the biggest challenge for nonlinear mappings: they fit the training data well but fail to predict the test data correctly, since the nonlinearity also captures the noise of the training data. Therefore, extra generalization terms such as weight decay [100], dropout [153] are required to reduce the influence of over-fitting. Besides, nonlinear formulations cost more computational resource in a machine than linear ones.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

In practice, according to the principle of Occam's razor [116], if a linear solution and a nonlinear solution are equally effective to a problem, the linear one should be preferred due to its simplicity.

A cost function: convex vs. non-convex

Even we have chosen a linear mapping function, we still have to determine the convexity of a cost function. Generally, a convex cost function holds a global optimal solution but a non-convex function has more than one local optima.

Relatively, the major advantage of a convex formulation is that it can be efficiently solved by *convex optimization* [23] algorithms and the only solution sounds unique to the users. However, a local optimum of solvable non-convex functions may be also a good solution to practical problems, e.g. the cost functions of NCA [55] and CSML [125] are non-convex (Section 2.3). Recent theoretical and empirical results strongly suggest that local optima are not a serious issue in general [99]: regardless of initial conditions, a non-convex system nearly always reaches local solutions of very similar quality.

Mapping function vs. Cost function: which one plays a more important role?

The last question is that between the design of an architecture and the formulation of a metric-based cost, which one plays a more important role in a Metric Learning system? We believe that there is no simple answer, and an effective Metric Learning system should be a collaboration of both.

But it should be noted that the trend of *deep learning* [15, 99, 16, 107] has attracted more and more attention to constructing deep architectures for non-trivial, large-scale problems. Configuring the structure of a system and designing an effective deep architecture is indeed an art: we need to carefully choose the type of layers, the number of layers, the number of nodes in each layer, the connectivity mode among the layers, the choice of activation functions, etc.

Consequently, when we develop our own Metric Learning system, we take into account both the mapping function and the cost function. In following chapters, we will first propose a novel similarity-based cost function naturally related to the triangle inequality theorem and explain its objective by geometrical illustration. We will then integrate the cost function with three different mapping functions, namely, a linear transformation, an MLP or a CNN, respectively. We will evaluate these linear and nonlinear systems in different applications such as pairwise face verification, speaker verification, kinship verification, object classification and data visualization.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Chapter 3

Triangular Similarity Metric Learning

3.1 Introduction

In the previous chapter, we have reviewed the literature of linear Metric Learning methods and also their non-linear variants, i.e. Siamese Neural Networks. We have concluded that a good Metric Learning system should be a collaborative product of designing a mapping architecture and formulating a metric-based cost function.

The current literature has offered us many exemplars: for a mapping function, the depth of the architecture can be shallow or deep, namely, shallow linear transformations [13], multi-layer neural networks [142, 31, 177, 18] or deep neural networks [6, 24, 32, 62, 122]; for a cost function, we have a lot of options based on a distance metric [173, 147, 169, 55, 52, 39, 60], a similarity metric [136, 135, 29, 125] or a hybrid metric concerning both distance and similarity [7, 26].

In general, more works in the current literature focused on learning a distance metric rather than learning a similarity metric. Among all the few similarity metric learning methods, most of them concerned the Cosine Similarity metric [136, 135, 125] or its relaxed variant, the bilinear similarity [28, 29]. In this thesis, we contribute to the study on similarity metric learning and propose an alternative metric which is equivalent to the Cosine Similarity metric but has a nicer geometrical interpretation of learning the similarity. This novel metric is naturally related to the well-known *triangle inequality theorem*, so we call it the *Triangular Similarity*. Moreover, we will develop an efficient and effective cost function to learn a Triangular Similarity metric. By examining its gradient, we will discover that the cost function can be easily enrolled in a linear or nonlinear architecture of neural networks.

This chapter focuses on introducing and illustrating the methodology of our Triangular Similarity Metric Learning (TSML) approach, the main contributions of this chapter are summarized as below:

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

- We propose the *Triangular Similarity* and illustrate its equivalence to the Cosine Similarity in measuring a data pair.
- We develop the *triangular loss function* and show its connection to the Mean Squared Error (MSE) function of traditional neural networks.
- We visualize the mapping objective of the proposed TSML system.

3.2 Triangular Similarity

For any two given vectors $\mathbf{a}$ and $\mathbf{b}$, the Triangular Similarity between them is measured by:

$$tri(\mathbf{a},\mathbf{b}) = \frac{1}{2}\|\frac{\mathbf{a}}{\|\mathbf{a}\|} + \frac{\mathbf{b}}{\|\mathbf{b}\|}\|, \tag{3.1}$$

apparently, the value of this similarity lies in the range [0, 1]. This function results in 1 if and only if the two vectors $\mathbf{a}$ and $\mathbf{b}$ are towards the same direction, and yields 0 if and only if the directions of the two vectors are exactly opposite.

Now we relate the Triangular Similarity to the Cosine Similarity:

$$\begin{aligned} tri(\mathbf{a},\mathbf{b}) &= \frac{1}{2}\|\frac{\mathbf{a}}{\|\mathbf{a}\|} + \frac{\mathbf{b}}{\|\mathbf{b}\|}\| \\ &= \frac{1}{2}\sqrt{(\frac{\mathbf{a}}{\|\mathbf{a}\|} + \frac{\mathbf{b}}{\|\mathbf{b}\|})^T(\frac{\mathbf{a}}{\|\mathbf{a}\|} + \frac{\mathbf{b}}{\|\mathbf{b}\|})} \\ &= \frac{1}{2}\sqrt{(\frac{\mathbf{a}}{\|\mathbf{a}\|})^2 + (\frac{\mathbf{b}}{\|\mathbf{b}\|})^2 + 2\frac{\mathbf{a}^T\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}} \\ &= \frac{1}{2}\sqrt{2 + 2\frac{\mathbf{a}^T\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}} \\ &= \frac{1}{2}\sqrt{2 + 2cos(\mathbf{a},\mathbf{b})} \\ &= \sqrt{\frac{1 + cos(\mathbf{a},\mathbf{b})}{2}}, \end{aligned} \tag{3.2}$$

where $cos(\mathbf{a},\mathbf{b})$ is the standard Cosine Similarity function. We can see that the relation between $tri(\mathbf{a},\mathbf{b})$ and $cos(\mathbf{a},\mathbf{b})$ is a *consecutive* and *bijective* function[1] $f(z) = \sqrt{(1+z)/2}$ in its effective domain $z \geq -1$, indicating that they are equivalent in measuring a similarity.

[1] In mathematics, a bijective function is a function between the elements of two sets, where every element of one set is paired with exactly one element of the other set, and every element of the other set is paired with exactly one element of the first set.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

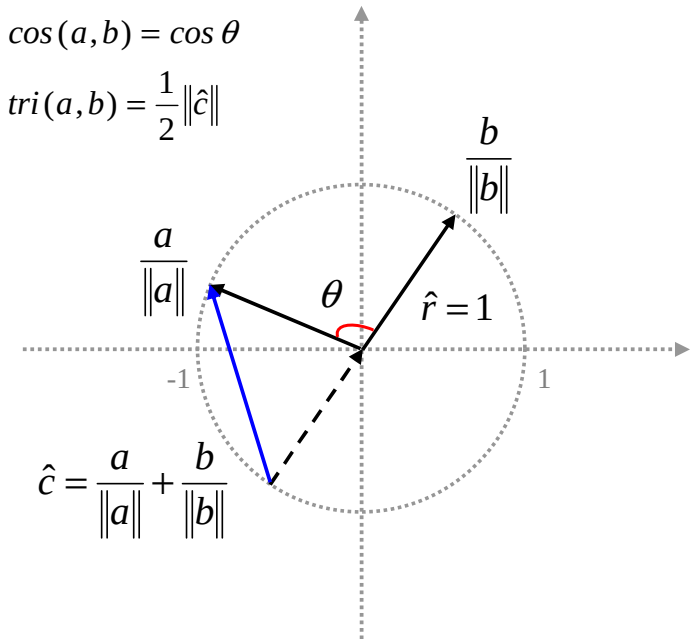


Fig. 3.1 Equivalence between Triangular Similarity and Cosine Similarity. While the Cosine Similarity simply calculates the *cosine* of the angle θ between the two vectors, the Triangular Similarity measures the half length of the directed chord (the blue line). The three vectors $\frac{\mathbf{a}}{\|\mathbf{a}\|}$, $\frac{\mathbf{b}}{\|\mathbf{b}\|}$ and $\hat{\mathbf{c}}$ compose an *isosceles triangle*. The two similarity functions compose a one-to-one correspondence thus the equivalence is confirmed.

A more intuitive interpretation of the relationship between the two similarities is illustrated in Fig. 3.1. In a Cartesian coordinate system, $\frac{\mathbf{a}}{\|\mathbf{a}\|}$ and $\frac{\mathbf{b}}{\|\mathbf{b}\|}$ represent two vectors lying on the unit circle, i.e. a circle with a radius of one. And their sum determines a directed chord on the circle (the blue line), denoted by a new vector $\hat{\mathbf{c}}$. The Cosine Similarity simply calculates the *cosine* of the angle θ between the two vectors, and the Triangular Similarity halves the length of the directed chord. When the angle θ decreases from π to 0, the value of the Cosine Similarity increases from the minimum -1 to the maximum 1, and the value of the Triangular Similarity raises from 0 to 1. Note that the three vectors $\frac{\mathbf{a}}{\|\mathbf{a}\|}$, $\frac{\mathbf{b}}{\|\mathbf{b}\|}$ and their sum $\hat{\mathbf{c}}$ compose an *isosceles triangle*, that is why we call this similarity measurement as the *Triangular Similarity*.

3.3 Triangular Loss Function

For both the Cosine Similarity and the Triangular Similarity, the scale of the factors $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$, i.e. the length of the vectors, should be taken care of. Otherwise, it may raise up numerical instability problems [67] especially when $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ are too small. This

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

hidden problem can be avoided or relieved by many strategies. For example, adopting regularization terms to prevent $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ from degenerating to 0 [18], normalizing the inputs by a whitening transformation [77, 185] so that all the input variables have unit variance [100], or simply performing an L2 normalization to let all the vectors have unit length [26, 150]. Empirical experiences showed that these strategies usually bring up to faster convergence [100] and better performance [26, 185] on machine learning applications.

In this work, we propose a *soft L2 normalization* to constrain the length of the vectors. Different from the above strategies that perform normalization as a preprocessing step before feeding the inputs in the metric learning system, we constrain the length of the vectors to a constant r by a regularization function:

$$\min \ \ (\|\mathbf{a}\| - r)^2 + (\|\mathbf{b}\| - r)^2. \tag{3.3}$$

Minimizing the above function is able to make the values of $\|\mathbf{a}\|$ and $\|\mathbf{b}\|$ approaching a predefined scalar r, but in fact not all the vector lengths can be exactly r, so we consider it as a *soft* length normalization.

With the indicative assumption of $\|\mathbf{a}\| \approx \|\mathbf{b}\| \approx r$, we can simplify the Cosine Similarity and the Triangular Similarity. For the Cosine Similarity, its function can be rewritten as:

$$cos(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a}^T\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \approx (\frac{1}{r})^2\mathbf{a}^T\mathbf{b}, \tag{3.4}$$

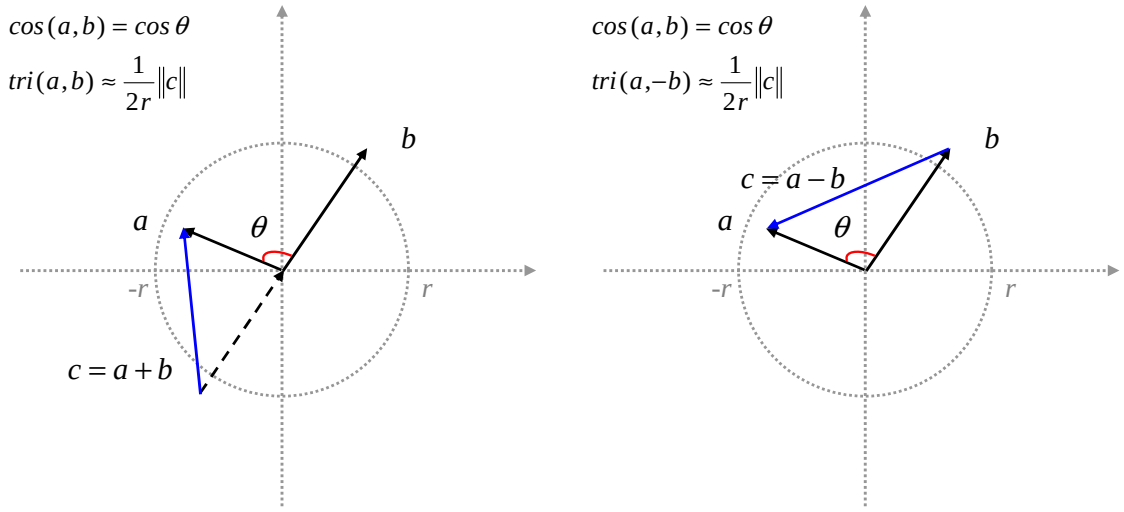
where $\mathbf{a}^T\mathbf{b}$ is the bilinear similarity. This approximation indicates that when vectors have approximate lengths, the bilinear similarity can be an equally effective but more efficient substitute of the Cosine Similarity in practical applications [28, 29, 41, 26]. Analogously, the Triangular Similarity can be simplified as:

$$tri(\mathbf{a},\mathbf{b}) = \frac{1}{2}\|\frac{\mathbf{a}}{\|\mathbf{a}\|} + \frac{\mathbf{b}}{\|\mathbf{b}\|}\| \approx \frac{1}{2r}\|\mathbf{a}+\mathbf{b}\|. \tag{3.5}$$

Instead of the isosceles triangle in Fig. 3.1, the above simplified Triangular Similarity concerns a normal triangle lying around a circle with a radius of r (see Fig. 3.2 (a)). This triangle is determined by the two vectors $\mathbf{a}$ and $\mathbf{b}$, completed with their sum $\mathbf{c} = \mathbf{a}+\mathbf{b}$ as the third side. Additionally, the two vectors $\mathbf{a}$ and $\mathbf{b}$ determine another triangle where the third side is the difference of $\mathbf{a}$ and $\mathbf{b}$, i.e. $\mathbf{c} = \mathbf{a}-\mathbf{b}$ (Fig. 3.2 (b)). With respect to the operations of sum or subtraction, we name the two triangles as the positive triangle and the negative triangle, respectively.

Like all the similarity metric learning methods, the objective of learning a Triangular Similarity metric is to increase the similarity between a similar pair and to decrease the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) Positive Triangle (b) Negative Triangle

Fig. 3.2 The simplified Triangular Similarity concerns normal triangles lying around a circle with a radius of r. A pair of vectors $\mathbf{a}$ and $\mathbf{b}$ determines two triangles: the positive triangle (left) illustrates the Triangular Similarity between $\mathbf{a}$ and $\mathbf{b}$; the negative triangle illustrates the Triangular Similarity between $\mathbf{a}$ and $-\mathbf{b}$.

similarity between a dissimilar pair. An intuitive geometrical interpretation of the objective is provided in Fig. 3.2.

- When $\mathbf{a}$ and $\mathbf{b}$ are labeled as being similar, increasing the pairwise similarity means minimizing the inter-vector angle θ, which can be realized by maximizing the length of the third side $\mathbf{c}$ in the positive triangle (Fig. 3.2 (a)).
- When $\mathbf{a}$ and $\mathbf{b}$ are a dissimilar pair, we need to separte the two vectors with a larger angle θ. This can be also achieved by maximizing the length of the third side $\mathbf{c}$ in the negative triangle (Fig. 3.2 (b)).

The cost function

Finally, let $s_i = 1$ (respectively -1) denote a pair of vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ being similar (respectively dissimilar). With the soft length normalization factors $(\|\mathbf{a}_i\| - r)^2$, $(\|\mathbf{b}_i\| - r)^2$, and the lengths of the three sides, $\|\mathbf{a}_i\|$, $\|\mathbf{b}_i\|$, $\|\mathbf{c}_i\|$, the triangular loss of this pair is defined as:

$$J_i = \frac{1}{2}[\boxed{(\|\mathbf{a}_i\| - r)^2 + (\|\mathbf{b}_i\| - r)^2}] + r(\boxed{\|\mathbf{a}_i\| + \|\mathbf{b}_i\| - \|\mathbf{c}_i\|}), \tag{3.6}$$

where r is a constant constraint for the vector length; $\mathbf{c}_i = \mathbf{a}_i + s_i\mathbf{b}_i$, representing the simplified Triangular Similarity in a positive triangle or a negative triangle. It is interesting to find that

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

the second part of this equation naturally obeys the *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. $\|\mathbf{a}_i\| + \|\mathbf{b}_i\| - \|\mathbf{c}_i\| \geq 0$.

Moreover, the coefficients of the two parts are set to $\frac{1}{2}$ and r, in order to further simplify the formulation of Equation (3.6) as:

$$J_i = \frac{1}{2}\|\mathbf{a}_i\|^2 + \frac{1}{2}\|\mathbf{b}_i\|^2 - r\|\mathbf{c}_i\| + r^2. \tag{3.7}$$

The gradient function

In Metric Learning systems, the vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ are outputs of a certain mapping function $f(\cdot)$ parameterized by a set of parameters (see Fig. 3.3). We now deduce the gradient of the triangular loss function (Equation (3.7)) with respect to the parameter set $\mathbf{W}$.

First of all, the derivative of the vector norm is (see the proof in Appendix A):

$$\frac{\partial\|\mathbf{a}\|}{\partial\mathbf{W}} = \frac{\partial\|\mathbf{a}\|}{\partial\mathbf{a}}\frac{\partial\mathbf{a}}{\partial\mathbf{W}} = \frac{\mathbf{a}}{\|\mathbf{a}\|}\frac{\partial\mathbf{a}}{\partial\mathbf{W}}. \tag{3.8}$$

Thus the derivative of the triangular loss is:

$$\begin{aligned}\frac{\partial J_i}{\partial\mathbf{W}} &= \mathbf{a}_i\frac{\partial\mathbf{a}_i}{\partial\mathbf{W}} + \mathbf{b}_i\frac{\partial\mathbf{b}_i}{\partial\mathbf{W}} - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}\frac{\partial\mathbf{c}_i}{\partial\mathbf{W}} \\ &= \mathbf{a}_i\frac{\partial\mathbf{a}_i}{\partial\mathbf{W}} + \mathbf{b}_i\frac{\partial\mathbf{b}_i}{\partial\mathbf{W}} - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}\frac{\partial\mathbf{a}_i + s_i\mathbf{b}_i}{\partial\mathbf{W}} \\ &= (\mathbf{a}_i - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\frac{\partial\mathbf{a}_i}{\partial\mathbf{W}} + (\mathbf{b}_i - r\frac{s_i\mathbf{c}_i}{\|\mathbf{c}_i\|})\frac{\partial\mathbf{b}_i}{\partial\mathbf{W}}.\end{aligned} \tag{3.9}$$

Since the partial derivatives $\frac{\partial\mathbf{a}_i}{\partial\mathbf{W}}$ and $\frac{\partial\mathbf{b}_i}{\partial\mathbf{W}}$ are controlled by the specific mapping function $f(\cdot)$, the minimal cost can be obtained at the zero gradient when $\mathbf{a}_i = r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}$ and $\mathbf{b}_i = r\frac{s_i\mathbf{c}_i}{\|\mathbf{c}_i\|}$. In other words, the gradient function has $r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}$ and $r\frac{s_i\mathbf{c}_i}{\|\mathbf{c}_i\|}$ as targets for $\mathbf{a}_i$ and $\mathbf{b}_i$, respectively. Figure 3.4 illustrates that: for a similar pair (when $s_i = 1$), $\mathbf{a}_i$ and $\mathbf{b}_i$ are mapped to the same vector in parallel with the third side of the positive triangle (the red solid line); for a dissimilar pair (when $s_i = -1$), $\mathbf{a}_i$ and $\mathbf{b}_i$ are mapped to two opposite vectors in parallel with the third side of the negative triangle (the blue solid line).

Most importantly, this gradient function confirms that though the Triangular Similarity in the cost function is only a simplified version as we have assumed the vectors having approximate lengths, we can still achieve the objective of closing a similar pair and separating a dissimilar pair: (1) for two similar vectors, the gradient defines an identical target between them; (2) for two dissimilar vectors, the gradient projects them to opposite directions.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

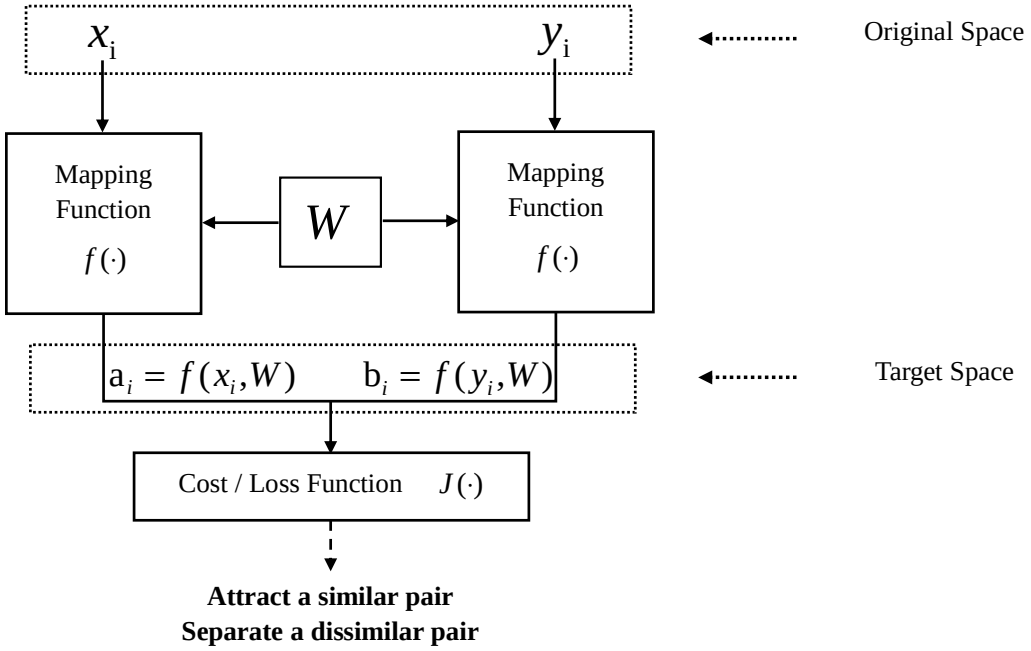


Fig. 3.3 The siamese architecture of Metric Learning.

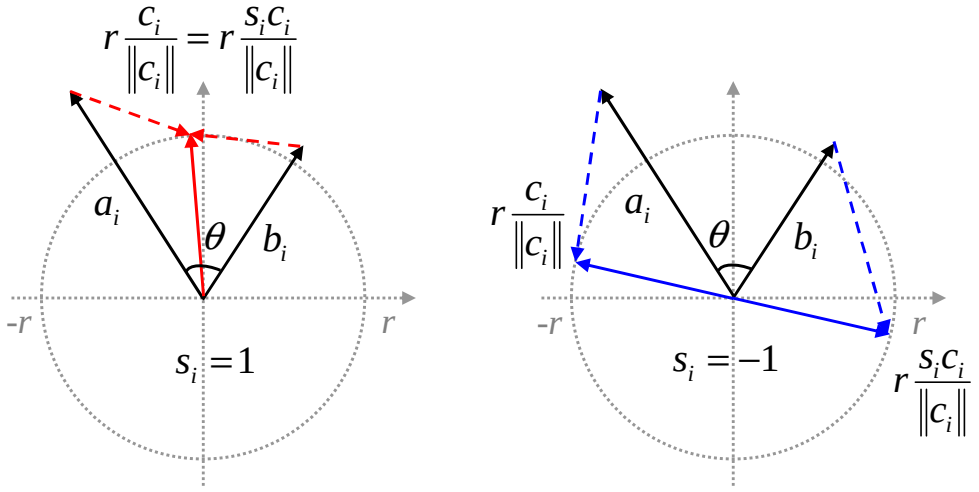


Fig. 3.4 The minimal cost can be obtained at the zero gradient when $\mathbf{a}_i = r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}$ and $\mathbf{b}_i = r\frac{s_i\mathbf{c}_i}{\|\mathbf{c}_i\|}$: for a similar pair (when $s_i = 1$), $\mathbf{a}_i$ and $\mathbf{b}_i$ are mapped to the same vector along the red solid line; for a dissimilar pair (when $s_i = -1$), $\mathbf{a}_i$ and $\mathbf{b}_i$ are mapped to two opposite vectors along the blue solid line.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

3.4 Relation to Traditional Neural Networks

As we have mentioned in previous chapters, the structure of our Metric Learning method is indeed the symmetric architecture in Siamese Neural Networks [24, 32]. The mapping function $f(\cdot)$ in Fig. 3.3 can be realized by any traditional neural networks, from linear single layer perceptrons [141] to nonlinear Multi-layer Perceptrons (MLP) [142], to deep nonlinear Convolutional Neural Networks (CNN) [96]. Furthermore, besides the common relation of the mapping function, the proposed *triangular loss function* also has natural connection to the Mean Squared Error (MSE) function, i.e. the most commonly used cost function in traditional neural networks [142, 95, 100].

3.4.1 Relation to the Mean Squared Error Function

For classification problems, the Mean Squared Error (MSE) loss function must be the earliest and the most popular cost function used in traditional neural networks, either in an MLP [142] or in a CNN [96]. It simply measures the difference between a computed output of a network and its desired target.

Formally, when we are given a training sample $\mathbf{x}_i$ and its predefined target $\mathbf{g}_i$, we first compute its output by the mapping function, i.e. $\mathbf{a}_i = f(\mathbf{x}_i, \mathbf{W})$, where $\mathbf{W}$ denotes the set of parameters in the mapping function. The error with respect to this training sample is defined as the squared Euclidean distance between $\mathbf{a}_i$ and $\mathbf{g}_i$:

$$J_i = \frac{1}{2}(\mathbf{a}_i - \mathbf{g}_i)^2, \tag{3.10}$$

and the partial derivative of the cost J_i with respect to the set of parameters $\mathbf{W}$ is:

$$\frac{\partial J_i}{\partial \mathbf{W}} = (\mathbf{a}_i - \mathbf{g}_i)\frac{\partial \mathbf{a}_i}{\partial \mathbf{W}}. \tag{3.11}$$

Usually, these predefined target values are typically binary for classification problems. For example, for a 4-class classification problem, we usually set unit vectors $[1,0,0,0]^T$, $[0,1,0,0]^T$, $[0,0,1,0]^T$, $[0,0,0,1]^T$ as target vectors for the 4 classes, respectively. Note that the dimension of the output vectors equals the number of classes.

Comparing this function with the gradient function of the triangular loss (Equation (3.9)), we find that the gradient function of the triangular loss is exactly *a double copy* of the MSE gradient: (1) the single output $\mathbf{a}_i$ in traditional neural networks is paired with a partner $\mathbf{b}_i$ to learn the pairwise relationship between data in a siamese architecture; (2) the hand-crafted target $\mathbf{g}_i$ is replaced by temporal targets $r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|}$ and $r\frac{s_i\mathbf{c}_i}{\|\mathbf{c}_i\|}$ which are automatically specified

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

by the two vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ themselves. This is indeed an advantage that the dimension of the output vectors is no longer required to be equal to the number of classes, and thus the proposed metric learning system is applicable for flexible dimensionality reduction. Furthermore, with the similar gradient formulations, typical optimization techniques and practical tricks of training neural networks [129] can be directly applied to optimize our triangular loss function. More details will be given as below.

3.4.2 Non-Convexity and Backpropagation

Different from most Metric Learning methods [173, 147, 169, 52, 39, 60, 29, 7, 26] that each holds a convex cost function and adopts a linear mapping function, our triangular loss function is non-convex and the mapping function can be either linear or nonlinear.

Generally, a global optimal solution is guaranteed to a linear and convex optimization problem [23, 11]. In contrast, among more than one local optima in a non-convex problem, there is no theories or formula to guarantee that the cost function will certainly converge to a good solution [100]. However, recent theoretical and empirical results strongly suggest that local optima are not a serious issue in general [99]: regardless of initial conditions, a non-convex system nearly always reaches local solutions of very similar quality.

Taking advantage of the connection between the triangular loss and the MSE cost, like the traditional neural networks, we directly employ the standard Backpropagation algorithm [142] to perform gradient descent. The update equation for gradient descent can be written as:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \mu \frac{\partial J^{(t)}}{\partial \mathbf{W}}, \tag{3.12}$$

where μ is the learning rate in an online gradient descent learning algorithm, $J^{(t)}$ is the cost at the t_{th} iteration. By successive training iterations, the parameters $\mathbf{W}$ is iteratively adjusted until reaching a relatively stable status, i.e. a local optimal solution.

Backpropagation can be very slow particularly for multi-layer networks where the cost surface is typically *non-quadratic*, *non-convex*, and high dimensional with many local minima and/or flat regions. The convergence may be quite slow, or even does not occur at all. However, a number of tricks such as normalizing the inputs, choosing good learning rates, initializing the weights, can greatly improve the chances of finding a good solution while also decreasing the convergence time often by orders of magnitude. Lecun *et al.* [100] have discussed these tricks and have examined the underlying theories of efficient Backpropagation. Nowadays, the popularity of *deep learning* [15, 99, 16, 107] has also demonstrated the effectiveness of deep non-linearity.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

3.4.3 Batch Gradient Descent or Stochastic Gradient Descent

Once we have defined a cost function and its gradient, the Back-propagation algorithm [142] applies gradient descent techniques to minimize the overall error for all the training data iteratively. There are mainly three modes to perform gradient descent: stochastic gradient descent, batch gradient descent, or the trade-off between them, mini-batch gradient descent. Concretely, stochastic gradient descent uses only one training sample in each iteration while batch gradient descent uses all training samples in each iteration. Mini-batch gradient descent, as the name suggests, takes several training samples in each iteration.

Stochastic gradient descent is generally preferred for large-scale Backpropagation. Stochastic gradient descent is most often much faster than batch gradient descent particularly on large redundant datasets. The reason is simple to show that stochastic gradient descent computes the gradient based on only one training sample while batch gradient descent has to average the gradient over all the training samples.

Stochastic gradient descent is particularly useful to model a function changing over time, a quite common scenario in industrial applications where the data distribution changes gradually over time (e.g. due to wear and tear of the machines). Since batch gradient descent always considers the whole training data, it is difficult to detect and follow the changes and thus it may result in rather bad solutions. In contrast, stochastic gradient descent, as a kind of online learning algorithm, can track the changes and yield good approximation results.

Despite the advantages of stochastic gradient descent, there are still reasons to use batch gradient descent [100]. Batch gradient descent can be involved in some advanced optimization algorithms to accelerate the learning speed, such as the Conjugate Gradient Descent (CGD) algorithm [115] and the Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm [108]. With these acceleration techniques that is hard to operate in stochastic gradient descent, the accelerated batch gradient descent can be much faster for small and medium scale problems, i.e. offline learning problems where the whole training set is ready as a batch. Besides, compared with the stochastic gradient descent technique, these advanced algorithms have no need to manually pick a learning rate.

Nowadays, we are more likely to face a large scale problem with an overlarge training set, it may be impossible to load all the training data into memory in a single iteration. In this case, one may prefer stochastic gradient descent or its variant, mini-batch gradient descent. Like stochastic gradient descent, mini-batch gradient descent is also applicable for online learning. And as the trade-off between stochastic gradient descent and batch gradient descent, mini-batch gradient descent is even a better choice than stochastic gradient descent for many online optimization problems [100].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

In summary, a proper choice of a gradient descent mode depends on the practical conditions in a problem, such as the data scale, the underlying distribution of the data. More detailed discussion in practical applications will be given in later chapters.

3.4.4 Various Mapping Functions

Last but not least, like traditional neural networks, we can incorporate various mapping functions with our triangular loss function for different applications. We simply give three typical options for the mapping function and summarize their characteristics here.

For a simple and general problem where we already have discriminative feature representations for objects, the Triangular Similarity with a linear mapping may be enough to describe the concerned semantic similarity between objects. In this case, according to the principle of Occam's razor [116], we prefer a linear single layer of perceptrons [141], i.e. a matrix, as the mapping function due to its simplicity.

If the data have some nonlinear underlying distribution that a linear similarity metric is unable to capture, we can adopt Multi-layer Perceptrons (MLP) [142] to perform nonlinear projections on the inputs of feature vectors.

For data that have a known *grid-like topology* [16] and do not have an effective vector representation, such as images and speech time-series, the Convolutional Neural Networks [96] may be the desirable choice for the mapping function to automatically extract features on the 2-dimensional inputs.

3.5 Visualization of the Objective

The objective of our triangular loss function seems too ideal for a dissimilar pair: in Fig. 3.4, the two vectors of a dissimilar pair (when $s_i = -1$), $\mathbf{a}_i$ and $\mathbf{b}_i$, are mapped to be exactly opposite to each other. However, in a limited space which contains a large quantity of classes, it is impossible to have all the dissimilar pairs oppositely separated. For example, when there are 3 different vectors in the 2-dimensional space, we can find at least one pair of vectors with the angle less than 180° (i.e. non-ideally opposite). However, the triangular loss function is able to balance the pairwise angles for all the data pairs and result in a relatively stable solution. In this section, in order to have an intuitive sense of the triangular loss function, we visualize the mapping results of some toy data.

We randomly select some points from different Gaussian distributions, i.e. normally distributed data, in the 2-dimensional input space. Similar and dissimilar pairs are generated as training data for our Triangular Similarity Metric Learning (TSML) system. We use a

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

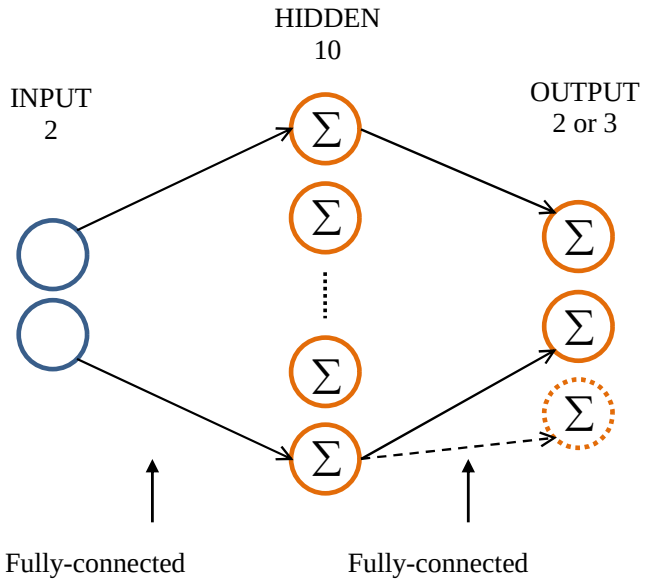


Fig. 3.5 Diagram of a 3-layer MLP, with only one input layer, one hidden layer and one output layer. Adjacent layers are fully connected and the activation function is the tanh function. TSML employs this 3-layer MLP as the mapping function.

3-layer MLP as the mapping function to project the original normally distributed data into a target space of 2-dimension or 3-dimension.

Figure 3.5 shows the naive MLP to realize the mapping from the input to the output, size of the only hidden layer is set to 10. Size of the output layer is set to 2 or 3, indicating the 2-dimensional or 3-dimensional target space, respectively. Adjacent layers are fully connected and the activation function is the tanh function. For a pair of inputs, the triangular loss is calculated on their outputs and Equation (3.9) is used in the Backpropagation algorithm to update the weights of this MLP. The length parameter r in Equation (3.9) is simply set to 1. As the scale of this toy problem is quite small, we employ the advanced L-BFGS algorithm to perform batch gradient descent. Specifically, we used a MATLAB implementation of L-BFGS provided by Mark Schmidt [144].

3.5.1 Example 1: Two Classes

We first illustrate an example of data originating from two Gaussian distributions. In the original 2-dimensional space, centers of the two classes are at $(0,1)$ and $(1,0)$, respectively. The standard variation on each dimension is simply set to 0.1 for the two groups of normally distributed data. We randomly generate 10 points for each class as the training data, and 100

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

points for each class as the test data. Figure 3.6 (a) plots the raw training data (left) and test data (right), respectively.

Since the TSML method performs learning on data pairs, we collect all the possible pairs between the training data. For the 20 training samples, the number of all sample pairs is simply $190 = 20 \times (20-1)/2$, where 100 of them are dissimilar pairs and the rest 90 are similar pairs.

To initialize the parameters of the MLP, we use a simple normalized random initialization method in [53], which is considered to be helpful for the tanh networks. The initialized MLP produces outputs for all the data samples and yields a new data distribution in the 2-dimensional target space, shown in Fig. 3.6 (b). Comparing it with the raw distribution, we can see that the center of each class has been moved and data in each class have been assembled by a certain degree.

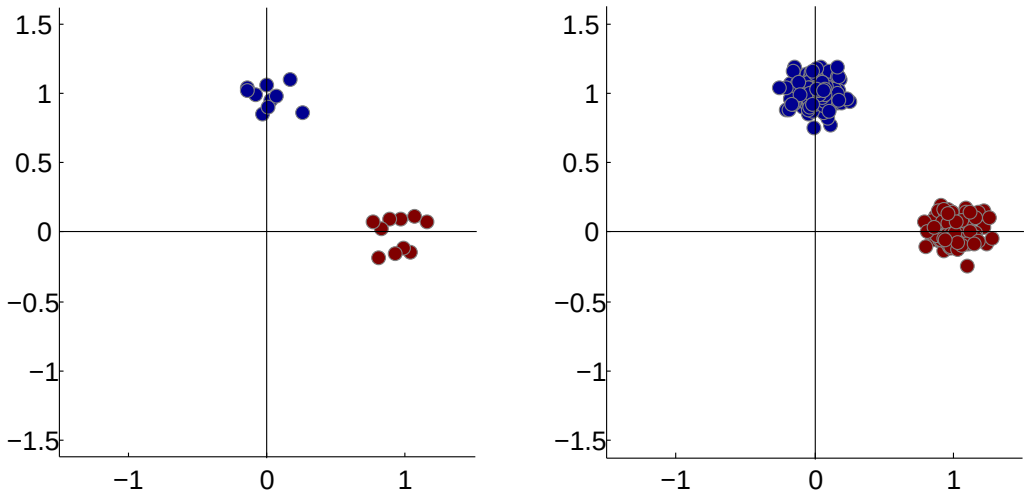
Iterative learning is performed by the TSML method on all the training pairs. We obtain an optimal model when the algorithm reaches convergence. Like the initialized model, the optimal model projects all the data samples into the 2-dimensional target space. And the results after learning is shown in the bottom picture, Fig. 3.6 (c). The objective of the triangular loss has been perfectly achieved: (1) the lengths of all the vectors are approaching the predefined parameter $r = 1$; (2) every similar pair are closed with an intersection angle of $0°$, i.e. points in the same class are mapped to an identical position; (2) every dissimilar pair are separated maximally with an intersection angle of $180°$, and thus the two classes locate oppositely on two sides of the origin $(0,0)$.

3.5.2 Example 2: Four Classes

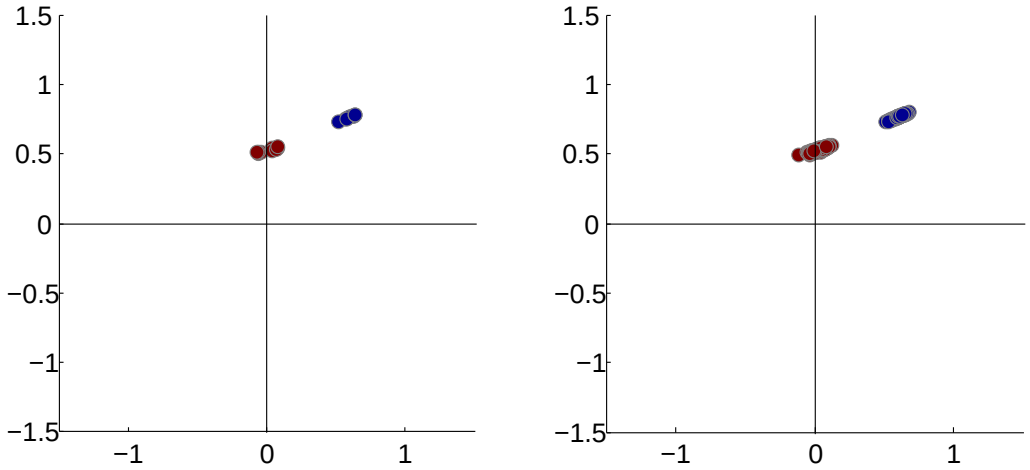
We now present a more complex example of four classes of normally distributed data. In the original 2-dimensional space (see Fig. 3.7 (a)), the centers of the four classes are at $(0,0.5)$, $(0,1.3)$, $(0,-1.3)$, $(0,-0.5)$, respectively. The standard derivation on each dimension is still set to 0.1. For each class, the number of training samples is 10, and the number of test samples is 100. Training pairs are generated by the 40 training samples in order to perform nonlinear metric learning. Normalized initialization is taken and the dimension of the target space is first set to 2.

Figure 3.7 (b) shows the projection using the initialized model, and Fig. 3.7 (c) illustrated the projection using the optimal model learned by the proposed TSML method. Comparing these two projections, we conclude this optimal solution as: (1) although the lengths of all the vectors can not reach 1, they are roughly equal; (2) points in the same class are mapped to the same position; (3) in the 2-dimensional coordinate system, the four different classes occupy a quadrant each, sharing the $360°$ around the origin. Thus any two neighboring classes have

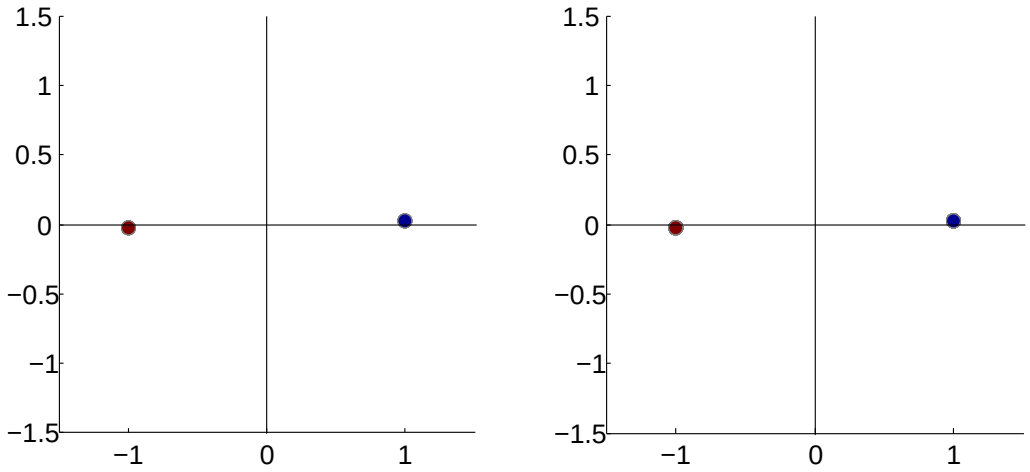
Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) The raw data



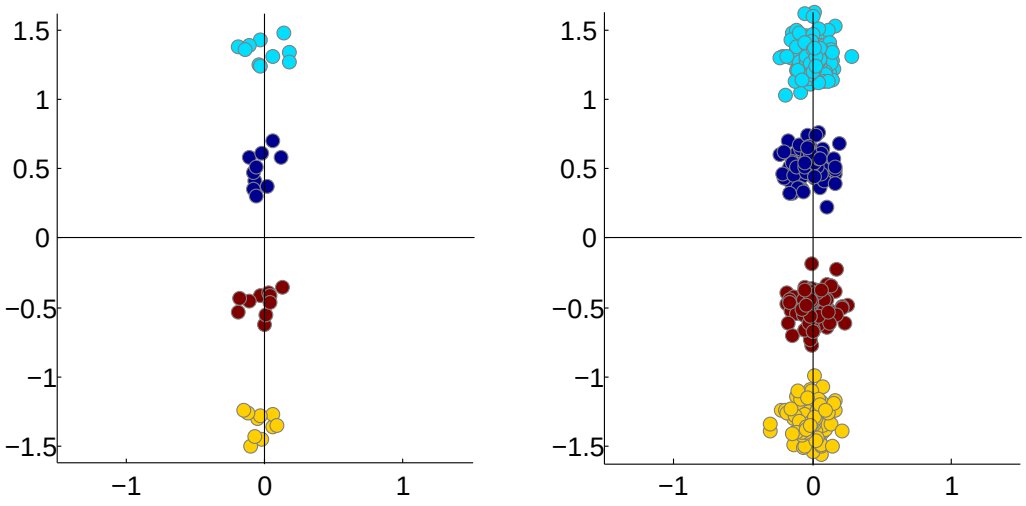
(b) After normalized initialization



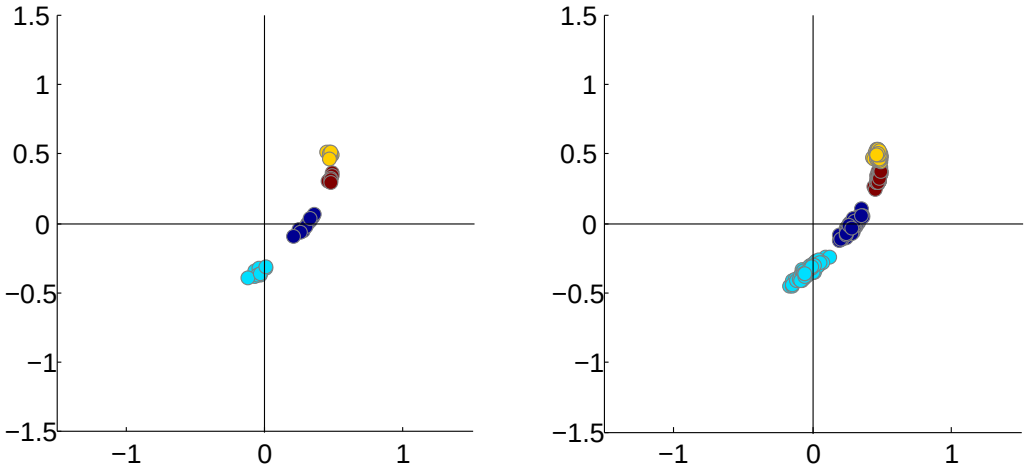
(c) After convergence

Fig. 3.6 Illustration of the 2-class toy problem, showing the distribution of the training data (left column) and the test data (right column) at different stages.

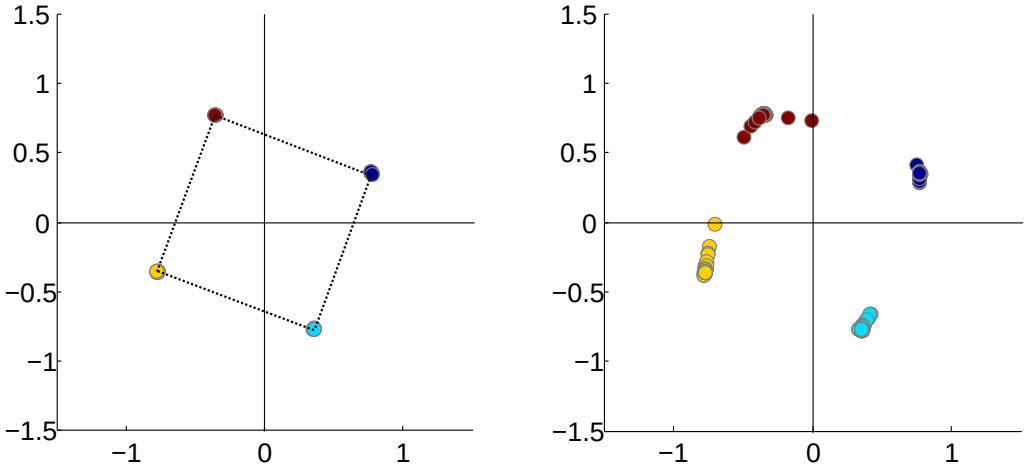
Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) The raw data



(b) After normalized initialization



(c) After convergence

Fig. 3.7 Illustration of the 4-class toy problem, showing the distribution of the projections of the training data (left column) and the test data (right column) in the 2-dimensional target space.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

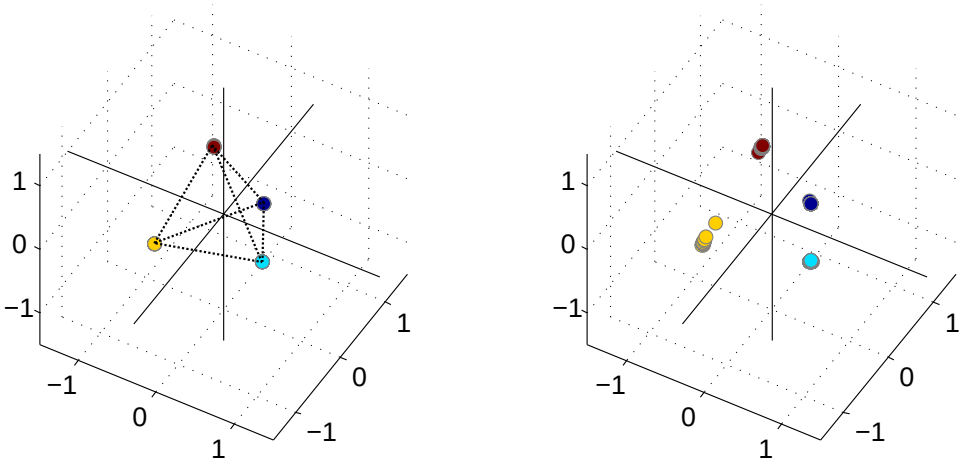


Fig. 3.8 Illustration of the 4-class toy problem, showing the distribution of the projections of the training data (left column) and the test data (right column) in the 3-dimensional target space.

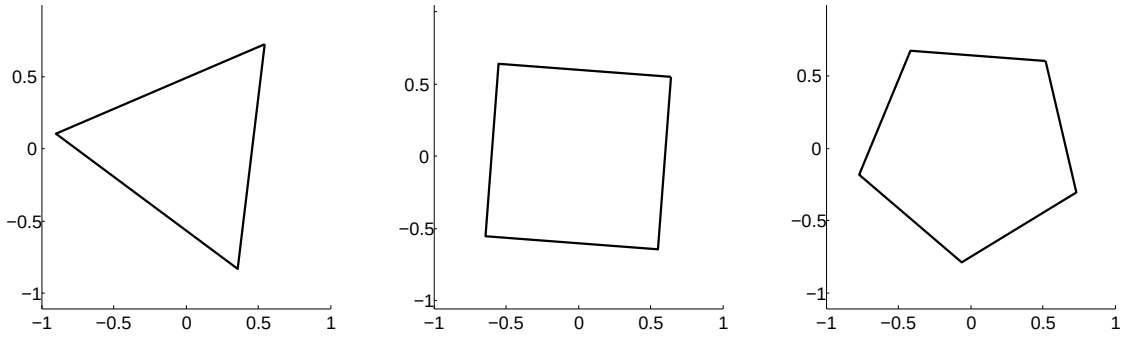
an intersection angle of 90°. Especially, the four classes of the training data (see the left part in Fig. 3.7 (c)) can be considered as four vertexes of a square. After all, the TSML method has accomplished the mission of closing similar pairs and separating dissimilar pairs.

We know that three points can only determine a surface but four points are enough to construct a body. Thus for this 4-class example, we also view its projection in the 3-dimensional target space, we simply set the size of the output layer to 3. Figure 3.8 shows the optimal projection learned by the TSML method. We can see that points of the same class have been concentrated into one cluster and different clusters have been maximally separated. Ideally on the training data, the four clusters construct a *regular tetrahedron* with the origin as the polyhedron center (see the left part in Fig. 3.8). We regard this regular tetrahedron as the ideal final state in the 3-dimensional space for the 4-class toy problem.

Final States: Polygons and Polyhedrons

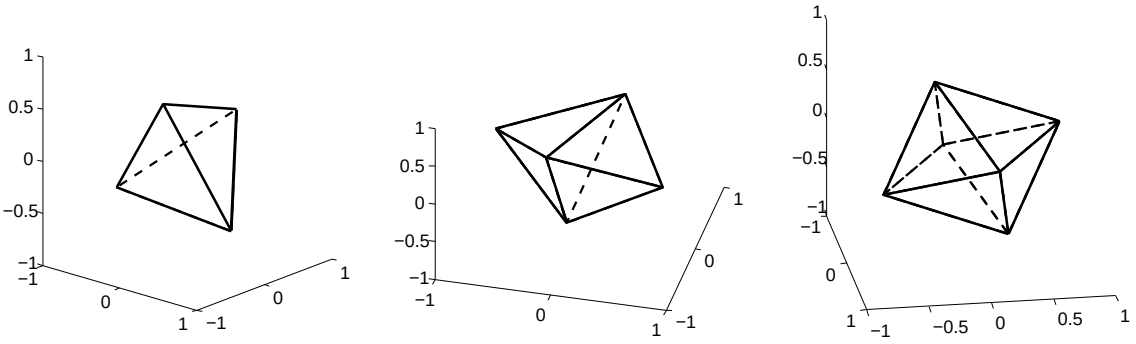
When there is no variance in each class of the training data, any toy problem can reach an ideal optimal solution via the proposed TSML method. And we can anticipate that the ideal final state of n clusters is an n-side regular polygon in the 2-dimensional space, or a convex polyhedron in the 3-dimensional space. Examples of some learned geometrical objects are shown in Fig. 3.9 and Fig. 3.10. For instance, the ideal final state of 5 clusters is a pentagon or a double-pyramid.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) Projection of 3 clusters (b) Projection of 4 clusters (c) Projection of 5 clusters

Fig. 3.9 Ideal projections by the proposed TSML method, showing regular polygons in the 2-dimensional space.



(a) Projection of 4 clusters (b) Projection of 5 clusters (c) Projection of 6 clusters

Fig. 3.10 Ideal projections by the proposed TSML method, showing convex polyhedrons in the 3-dimensional space.

3.6 Conclusion

In this chapter, we first proposed the *Triangular Similarity*, a novel similarity measurement equivalent to the Cosine Similarity in measuring a data pair. With the soft length normalization strategy, we showed how to simplify the Triangular Similarity and the Cosine Similarity. Based on the simplified Triangular Similarity, we further developed the *triangular loss* in order to perform metric learning, i.e. to increase the similarity between two vectors in the same class and to decrease the similarity between two vectors of different classes. After that, by examining the gradient function of the triangular loss, we found that its gradient has similar formulation with the gradient function of the Mean Squared Error (MSE), which has been widely used in neural networks for classification problems. As a result, it allows us to employ the standard Backpropagation algorithm to perform optimization for the proposed TSML

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

system. Moreover, compared with other metric learning methods, an intuitive geometrical interpretation of the metric learning objective is given.

We also used toy problems of normally distributed data to show the TSML system's capability of learning optimal projections. Concretely, the proposed learning system is able to assemble data of the same class to a single point, and also maximally separate different classes by projecting them with a balanced distribution around the origin of the target space. Taking the 4-class toy problem for example, the four classes are mapped to the vertexes of a square in the 2-dimensional space, or to the vertexes of a regular tetrahedron in the 3-dimensional space. Either the square or the regular tetrahedron has the origin as the center.

In the following chapters, we will apply the proposed TSML system to practical applications such as pairwise face verification, object classification and data visualization, in order to demonstrate its effectiveness for processing different kinds of datasets.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Chapter 4

Applications on Pairwise Verification

4.1 Introduction

We first apply the Triangular Similarity Metric Learning (TSML) method to a series of pairwise verification problems. The objective of pairwise verification is to verify a pair of data if they are from the same class or not. Formally, two samples of the same class are called a similar pair; otherwise, two samples from two different classes are called a dissimilar pair or a different pair. The objective of pairwise 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 [13], i.e. methods that automatically learn a metric from a set of data pairs.

The definition of similarity or dissimilarity differs from one application to another. Taking the comparison of two face images for example: (1) in the task of face verification, two face images of the same person are considered to be similar; (2) in the task of gender verification, two face images capturing two males or two females is a similar pair; (3) in the task of kinship verification, a similar pair of face images must indicate a blood relationship between the two persons in the images, such as father and son, mother and daughter. Hence, a good metric learning method must be able to capture the intrinsic relationship between the concerned semantic contents of two objects.

Compared with the traditional identification task in which a decision of acceptance or rejection is made by comparing a sample to models (or templates) of each class [118, 138], pairwise verification is more challenging because of the impossibility of building robust models with enough training data for each class [75]. Actually, there may be only one training sample available for some classes in the experimental setting of pairwise verification. Besides, individuals in training and testing should be mutually exclusive, i.e. the testing set comprises only *unseen samples* that are not part of the training set. This strict experimental setting has been adopted in popular benchmarks such as the dataset 'Labeled Faces in the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Wild' (LFW) [75][1] for pairwise face verification and the dataset 'Kinship Face in the Wild' (KinFaceW) [112][2] for pairwise kinship verification.

Both the setting of limited training data for some classes and the setting of mutually exclusive training and test sets bring up to significant over-fitting problems: a model fits the training data easily but fails to predict the test data well. Thus we prefer the linear version of our TSML system for these pairwise verification problems since the linear one is able to fit the training data and it suffers less influence from over-fitting compared with the nonlinear ones. Empirically, state-of-the-art metric learning methods learned linear mappings to realize generalization [125, 8, 150, 26]. We will also demonstrate that without any additional regularization factor, a linear mapping generally achieves superior performance than a slightly deeper nonlinear mapping on the problems of pairwise verification.

The contribution of this chapter is that we apply the linear TSML system on different applications to show its effectiveness for the specific problem of pairwise verification:

- we apply the linear TSML method on the LFW dataset in terms of the evaluation of pairwise face verification.
- we apply the same TSML method for pairwise kinship verification on the KinFaceW dataset.
- we establish our own pairwise speaker verification protocol and demonstrate that TSML is also effective to process speech data besides the above face images.
- we compare the linear TSML to its nonlinear variants in order to investigate the importance of the linearity on the problem of pairwise verification.

4.2 Pairwise Face Verification

Pairwise face verification belongs to the category of face recognition. Actually, early face recognition datasets and protocols focused on another problem of identification where identity information about a set of individuals is collected as the training data. At test time, one needs to specify the identity information for new images, i.e. whether the individual in a test image has appeared in the training set or not; and if so, which identity in the training set is represented by the test image. In order to assess the performance of identity classification, identification datasets [117, 51, 149] require most of the test images to have captured the

[1] http://vis-www.cs.umass.edu/lfw/index.html
[2] http://www.kinfacew.com/index.html

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

individuals in the training data, and it is optional to have a few unseen test images as strangers to the training set.

In contrast, the problem of pairwise verification is to analyze two face images and decide whether they represent the same person or two different people. It is usually assumed that neither of the face images shows a person from any previous training set. Compared with identification, pairwise verification poses a more general problem of finding a universally effective solution to measure the concerned semantic similarity or difference between two face images with unconstrained individual assembly. Many well-known benchmarks have used pairwise verification as the evaluation protocol [140, 75, 170] but the LFW dataset [75] must be the most popular one since it has provided detailed evaluation standards and various protocols for different evaluation purposes.

4.2.1 The LFW Protocols and Related Work

With respect to pairwise face verification, the LFW dataset holds the setting of limited training data for some classes and the setting of mutually exclusive training and test sets. Originally, the LFW dataset proposed a *restricted* training protocol where only a few specified data pairs are allowed for training, a challenging setting in order to seek effective learning algorithms which have the ability to discover principles from a small number of training examples, just like the human beings [93]. Besides the restricted protocol, the unrestricted protocol is also provided to allow the creation of additional training pairs by combining the images in LFW [75].

Recently, as many researchers started using additional training data from outside LFW to improve performance or even inhibited the use of training data to realize unsupervised learning, new protocols were developed to maintain fair comparisons among methods [74].

Since the LFW dataset was released in 2007, more than 70 papers have been published and several commercial systems have reported their performance on this benchmark. Generally, methods using additional labeled training data achieved higher accuracies than those relying on LFW only. In particular, recent deep learning techniques have approached the accuracy of 100% under the LFW evaluation standard. Almost all the published methods employed deep Convolutional Neural Networks (CNN) to process face images and to learn robust face representation on additional large labeled training datasets, such as the DeepFace from Facebook using the non-public SFC dataset [161]; the DeepID family (DeepID [157], DeepID2 [155], DeepID2+ [158], DeepID3 [156]) using the CelebFaces dataset [109] [3] or/and the WDRef dataset [30]; the Face++ systems using the Megvii Face Classification

[3] http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Database [47, 186]; the FaceNet from Google using a 260-million image dataset [145], the Tencent-BestImage commercial system using their BestImage Celebrities Face dataset[4].

However, with limited training data, these deep methods are more prone to over-fitting and thus result in inferior performance in a test evaluation. Actually, under the restriction of no outside labeled training data, tremendous efforts have been put on developing robust face descriptors [162, 1, 81, 38, 171, 76, 150, 8, 30, 146, 130, 143, 103] and metric learning methods [169, 39, 59, 125, 72, 178, 26, 8, 69, 71]. Popular face descriptors include eigenfaces [162], Gabor wavelets [38], SIFT [111, 81], Local Binary Patterns (LBP) [1], etc. Especially, LBP and its variants, such as center-symmetric LBP (CSLBP) [66], multi-block LBP (MBLBP) [180], three patch LBP (TPLBP) [171] and over-complete LBP (OCLBP) [8], have been proven to be effective at describing facial texture. Since face verification needs an appropriate way to measure the difference or similarity between two images, many researchers have been studying metric learning which aims at automatically specifying a metric from data pairs. Readers are referred to the Chapter 2 for the literature review of major metric learning methods.

Besides, other efforts have been made to face frontalization (i.e. pose alignment) [17, 176, 102, 187, 63] or multiple descriptor fusion [36, 130, 4], in order to further improve face verification performance on the LFW dataset.

4.2.2 Linear Triangular Similarity Metric Learning

As we have discussed in Section 4.1, the definition of pairwise verification, as well as the restricted protocols in the LFW dataset, has posed a problem of limited training data for some classes and of mutually exclusive training and test sets. These settings make more variations between the training set and the test set, and thus over-fitting is more inclined to happen for a learning algorithm.

For the possible over-fitting problem, we propose to use a linear mapping in our TSML system since the linearity indicates a certain degree of generalization which can reduce the influence of over-fitting. Empirically, state-of-the-art metric learning methods also learned linear mappings to realize generalization [8, 150, 26], such as the most relevant work to ours, the Cosine Similarity Metric Learning [125] method. In this section, we present our linear Triangular Similarity Metric Learning system and analyze its superiority over the CSML method.

With the i_{th} training pair $(\mathbf{x}_i, \mathbf{y}_i)$ from the LFW dataset, a linear mapping parameterized by a matrix $\mathbf{W}$ produces two outputs $(\mathbf{a}_i, \mathbf{b}_i)$ where $\mathbf{a}_i = \mathbf{W}\mathbf{x}_i$ and $\mathbf{b}_i = \mathbf{W}\mathbf{y}_i$. In Chapter 3.3,

[4] http://bestimage.qq.com/

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

we have defined the triangular loss of this pair as:

$$J_i = \frac{1}{2}\|\mathbf{a}_i\|^2 + \frac{1}{2}\|\mathbf{b}_i\|^2 - r\|\mathbf{c}_i\| + r^2, \tag{4.1}$$

where r is a constant constraint on the vector length; $\mathbf{c}_i = \mathbf{a}_i + s_i\mathbf{b}_i$, representing the simplified Triangular Similarity in a positive triangle ($s_i = 1$) or a negative triangle ($s_i = -1$). And the gradient of this linear function is:

$$\begin{aligned}\frac{\partial J_i}{\partial \mathbf{W}} &= (\mathbf{W}\mathbf{x}_i - r\frac{\mathbf{W}\mathbf{x}_i + s_i\mathbf{W}\mathbf{y}_i}{\|\mathbf{W}\mathbf{x}_i + s_i\mathbf{W}\mathbf{y}_i\|})\mathbf{x}_i^T + (\mathbf{W}\mathbf{y}_i - s_i r\frac{\mathbf{W}\mathbf{x}_i + s_i\mathbf{W}\mathbf{y}_i}{\|\mathbf{W}\mathbf{x}_i + s_i\mathbf{W}\mathbf{y}_i\|})\mathbf{y}_i^T \\ &= (\mathbf{a}_i - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\mathbf{x}_i^T + (\mathbf{b}_i - s_i r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\mathbf{y}_i^T.\end{aligned} \tag{4.2}$$

The derivation of this equation can be proven in two ways: one is simply based on matrix derivatives and provided in Appendix A; the other is by regarding the linear system as a single-layer neural networks of linear perceptrons and following the chain rule of the Backpropagation algorithm [142] to calculate error propagation.

Comparison with Cosine Similarity Metric Learning

As the proposed linear TSML method has natural relation to the CSML method in [125], we compare them here and show that the gradient function of TSML is simpler than the gradient function of CSML. Remind that the CSML cost on a pair of vectors $(\mathbf{a}_i, \mathbf{b}_i)$ is defined as:

$$J_i = -s_i cos(\mathbf{a}_i, \mathbf{b}_i) = -s_i\frac{\mathbf{a}_i^T\mathbf{b}_i}{\|\mathbf{a}_i\|\|\mathbf{b}_i\|}, \tag{4.3}$$

where s_i is the similarity label of a data pair as in Equation (4.1). We can see that minimizing the CSML cost aims at minimizing the Cosine Similarity between a dissimilar pair and also maximizing the similarity between a similar pair (see discussion on CSML in Chapter 2). Moreover, it is shown in Appendix A that the gradient of the CSML cost is:

$$\begin{aligned}\frac{\partial J_i}{\partial \mathbf{W}} &= \frac{s_i}{\|\mathbf{W}\mathbf{x}_i\|\|\mathbf{W}\mathbf{y}_i\|}[(\frac{(\mathbf{W}\mathbf{x}_i)^T\mathbf{W}\mathbf{y}_i}{\|\mathbf{W}\mathbf{x}_i\|^2}\mathbf{W}\mathbf{x}_i - \mathbf{W}\mathbf{y}_i)\mathbf{x}_i^T + (\frac{(\mathbf{W}\mathbf{x}_i)^T\mathbf{W}\mathbf{y}_i}{\|\mathbf{W}\mathbf{y}_i\|^2}\mathbf{W}\mathbf{y}_i - \mathbf{W}\mathbf{x}_i)\mathbf{y}_i^T] \\ &= \frac{s_i}{\|\mathbf{a}_i\|\|\mathbf{b}_i\|}[(\frac{\mathbf{a}_i^T\mathbf{b}_i}{\|\mathbf{a}_i\|^2}\mathbf{a}_i - \mathbf{b}_i)\mathbf{x}_i^T + (\frac{\mathbf{a}_i^T\mathbf{b}_i}{\|\mathbf{b}_i\|^2}\mathbf{b}_i - \mathbf{a}_i)\mathbf{y}_i^T].\end{aligned} \tag{4.4}$$

Comparing the gradient functions of TSML and CSML, i.e. Equation (4.2) vs. Equation (4.4), we find that the gradient of TSML needs fewer matrix multiplications than that of CSML, this makes TSML more efficient than CSML during gradient descent.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Concretely, except from the four fundamental operations of arithmetic, i.e. addition, subtraction, multiplication and division on scalar forms, matrix multiplications are the major consumptions in computing the gradients. We can see that the two gradient functions have common parts such as the computation of $\mathbf{W}\mathbf{x}_i$ and $\mathbf{W}\mathbf{y}_i$, but they differ in the computation of vector inner products: CSML (Equation (4.4)) requires to compute three inner products, namely, $(\mathbf{W}\mathbf{x})_i^T\mathbf{W}\mathbf{x}_i$, $(\mathbf{W}\mathbf{y})_i^T\mathbf{W}\mathbf{y}_i$ and $(\mathbf{W}\mathbf{x})_i^T\mathbf{W}\mathbf{y}_i$; TSML (Equation (4.2)) only needs to compute one inner product, $(\mathbf{W}\mathbf{x}_i + s_i\mathbf{W}\mathbf{y}_i)^T(\mathbf{W}\mathbf{x}_i + s_i\mathbf{W}\mathbf{y}_i)$. As a result, TSML runs faster than CSML in practical computing. A runtime comparison experiment of CSML and TSML will be presented in Section 4.2.5 to show that TSML is more efficient than CSML while they are actually equally effective on the problem of face verification.

Overall Cost and Gradient

If we have more than one training pair in each iteration for gradient descent, we average the cost on all training pairs as the overall cost:

$$J = \frac{1}{N}\sum_{i=1}^{N} J_i = \frac{1}{N}\sum_{i=1}^{N}[\frac{1}{2}\|\mathbf{a}_i\|^2 + \frac{1}{2}\|\mathbf{b}_i\|^2 - r\|\mathbf{c}_i\| + r^2], \tag{4.5}$$

where N is the number of all possible similar pairs and dissimilar pairs for training. And the corresponding gradient function is:

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{1}{N}\sum_{i=1}^{N}[(\mathbf{a}_i - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\mathbf{x}_i^T + (\mathbf{b}_i - s_i r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\mathbf{y}_i^T]. \tag{4.6}$$

It is noteworthy that in this overall case, we can not obtain the optimal solution by simply computing the zero gradient. Indeed, a specific vector $\mathbf{a}_i$ appears in more than one training pairs (with different $\mathbf{b}_i$ each), that means taking zero gradient would set many different targets for $\mathbf{a}_i$. Hence it is impossible for $\mathbf{a}_i$ to satisfy all the targets at the same time. In fact, $\mathbf{a}_i$ will be moved towards a balanced point of all the targets rather than any one of them. And an optimal solution minimizes and balances the cost from all the training pairs.

We have adopted the linear mapping as a kind of weapon to fight against the possible over-fitting problem in the pairwise face verification task. Additionally, following [125, 26], we add a regularization term to the cost function as another means to prevent the occurrence of over-fitting:

$$J = \frac{1}{N}\sum_{i=1}^{N}[\frac{1}{2}\|\mathbf{a}_i\|^2 + \frac{1}{2}\|\mathbf{b}_i\|^2 - r\|\mathbf{c}_i\| + r^2] + \frac{\lambda}{2}(\mathbf{W} - \mathbf{W}_0)^2, \tag{4.7}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

where $\mathbf{W}_0$ is a predefined constant matrix. Moreover, $\mathbf{W}_0$ is also the initialization for $\mathbf{W}$, i.e. $\mathbf{W}$ is set to be $\mathbf{W}_0$ before optimization. A positive parameter λ adjusts the effects of the regularization term: the larger the parameter λ is, the closer $\mathbf{W}$ is to $\mathbf{W}_0$. And the best $\mathbf{W}$, denoted by $\mathbf{W}_\star$, which performs the best on a validation set, is selected by tuning the parameter λ. This regularization guarantees the learned optimal transformation matrix $\mathbf{W}_\star$ to realize a better mapping than the initial matrix $\mathbf{W}_0$ does [125]. The gradient of the final cost is:

$$\frac{\partial J}{\partial \mathbf{W}} = \frac{1}{N}\sum_{i=1}^{N}[(\mathbf{a}_i - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\mathbf{x}_i^T + (\mathbf{b}_i - s_i r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\mathbf{y}_i^T] + \lambda(\mathbf{W} - \mathbf{W}_0). \tag{4.8}$$

With this cost function and its gradient, we can apply Backpropagation algorithm [142] in either a stochastic gradient descent mode or a batch gradient descent mode, to find an optimal solution.

4.2.3 The LFW Dataset and Face Descriptors

The LFW dataset contains numerous annotated images from the web. The standard protocol only allows to use the View 2 subset of LFW for performance evaluation. In View 2, to do 10-fold cross validation, all the 5749 people in the dataset are divided into 10 subsets where the individuals are mutually exclusive. The total number of images for all the people is 13,233, however, the number of images for each individual varies from 1 to 530. Table 4.1 summarizes the data distribution of individuals and images in the 10 subsets.

To encourage fair and meaningful comparisons, LFW fixed the sample pairs in each fold for test evaluation. Specifically, 600 image pairs are provided in each fold of the LFW dataset, where 300 are similar and the other 300 are dissimilar [75]. Whatever the training conditions are, the recorded test evaluation is always the mean decision accuracy of the 10-fold cross validation, accompanied by its standard error in the 10 repetitions.

Under the constraint of no outside labeled training data, LFW defined two different training settings: the restricted setting in which only the fixed sample pairs are available for training; in contrast, under the unrestricted setting, it is possible to generate an arbitrary number of sample pairs since it is permitted to use the identity information.

Instead of using the original LFW images in experiments, the LFW web site provides two additional versions where the faces have been aligned. The first is referred to as LFW-funneled [73]. The alignment method in [73] was shown to improve classification rates over some of the early landmark-based face alignment methods, but was less effective than the later landmark methods, such as that in the second version, the LFW-a dataset [160]. However, works using the LFW-funneled data are in the category *with no outside data*, but

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.1 Distribution of individuals and images in the 10 subsets, where the individuals in different subsets are mutually exclusive. Ind.: individuals, Im.: images.

Index	1	2	3	4	5	6	7	8	9	10	Total
No. of Ind.	601	555	552	560	567	527	597	601	580	609	5749
No. of Im.	1369	1367	1089	1324	1016	1166	1690	1222	1207	1783	13233

works on the LFW-a dataset are categorized as *with label-free outside data* because LFW-a has used a trained commercial alignment system to align face images.

In this section, we use the images of the LFW-a version, the original size of each 8-bit grayscale image is 250×250. Following [125, 8, 26], at the preprocessing step, we simply crop an image to remove the background, leaving a 150×80 (height $\times$ width) face image. The next step after preprocessing is extracting facial features from the cropped images.

Face Descriptors

To represent a face image in the LFW-a dataset, we use four different face descriptors: Gabor wavelets [38], LBP [1], SIFT [111] and OCLBP [8]. Additionally, since the operation of square-rooting has been demonstrated to be useful for face verification [125, 8, 26], square roots of all the face descriptors are also evaluated. Details of feature extraction are listed as below.

Gabor wavelets: we extract Gabor wavelets with 5 scales and 8 orientations on each downsampled image. The downsampling rate is 10×10 for all the 150×80 images, thus the dimension of an extracted Gabor vector is 4800 $(= 5\times8\times15\times8)$.

Local Binary Patterns: we use the uniform LBP [128] to represent face images. The uniform LBP is denoted as $LBP^{u2}_{p,r}$, where $u2$ stands for 'uniform', (p,r) means to sample p points over a circle with a radius r. The dimension of an uniform pattern is 59. Concretely, each 150×80 image is divided into non-overlapping 10×10 blocks and uniform LBP patterns $LBP^{u2}_{8,1}$ are extracted from all the blocks. We concatenate all the LBP patterns into a single feature vector, whose dimension is 7080 $(= 15\times8\times59)$.

SIFT: we directly use the feature data provided by [60]. The SIFT descriptors were computed at fixed points on the face (corners of the mouth, eyes, and nose) located by a facial feature detector in [46]. They computed 128-dimensional SIFT descriptors at 3 scales, centered on 9 points, leading to a 3456 $(= 3 \times 9 \times 128)$ dimensional face vector.

Over-complete Local Binary Patterns: besides LBP, we also used a variant of LBP, OCLBP, to improve the overall performance on face verification [8]. Unlike LBP, OCLBP adopts overlapping to adjacent blocks. Formally, the configuration of OCLBP is denoted

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

as $S:(a,b,v,h,p,r)$: an image is divided into $a \times b$ blocks with vertical overlap of v and horizontal overlap of h, and then uniform pattern $LBP^{u2}_{p,r}$ are extracted from all the blocks. Moreover, OCLBP is composed of several different configurations. For example, Oren *et al.* [8] used three configurations: $S:(10,10,\frac{1}{2},\frac{1}{2},8,1),(14,14,\frac{1}{2},\frac{1}{2},8,2),(18,18,\frac{1}{2},\frac{1}{2},8,3)$. The three configurations consider three block sizes: $10\times 10, 14\times 14, 18\times 18$, and half overlap rates along the vertical and horizontal directions.

Concretely, *they shift the images to produce overlaps*. For instance, a cropped 150×80 image is divided into $15\times 8 = 120$ blocks with the size 10×10. Shifting the image to the left by a step $10\times\frac{1}{2}=5$ also produces 120 blocks; and shifting downwards produces another 120 blocks. Hence there are totally 360 blocks under the configuration $S:(10,10,\frac{1}{2},\frac{1}{2},8,1)$. Similarly, there are 198 blocks under the configuration $S:(14,14,\frac{1}{2},\frac{1}{2},8,2)$ and 135 blocks under the configuration $S:(18,18,\frac{1}{2},\frac{1}{2},8,3)$. In summary, the dimension of their OCLBP vectors is $40,887$ $((360+198+135)\times 59)$.

In our work, *we shift the block window to produce overlaps*. Taking the 10×10 block window for example, with the shifting step $10\times\frac{1}{2}=5$ to the left and downwards, the total number of 10×10 blocks is $(\frac{150}{5}-1)\times(\frac{80}{5}-1)=435$. Similarly, shifting the 14×14 window produces 231 blocks and shifting the 18×18 window produces 128 blocks. The dimension of our OCLBP vectors is $46,846$ $((435+231+128)\times 59)$.

Apparently, the two forms of OCLBP contains LBP as a subpart. Comparing the experimental results using the two different OCLBP, we found no significant difference. For example, WCCN gets 86.83% using our 46,846-d OCLBP, which is very close to 87.23% in [8] using the 40,887-d OCLBP.

Preprocessing: Dimensionality Reduction

Usually, automatic learning is performed on thousands of feature vectors. Directly taking the original facial vectors for learning causes computational problem. For example, the time required for multiplications between 46846-d OCLBP vectors would be unacceptable. Therefore, before learning, we apply Whitened Principle Component Analysis (WPCA) for dimensionality reduction [125, 8] as a preprocessing step. On the one hand, performing PCA reduces the original dimension; on the other hand, whitening makes new feature vectors more discriminative for face verification: the PCA reduced feature is normalized by the eigenvalues over all the dimensions, thus the negative influences of the large eigenvectors are reduced while the discriminating details of the smaller eigenvectors are enhanced [42]. In addition, normalizing the inputs by a whitening transformation [77, 185] makes all the input variables uncorrelated and having unit variance, which usually leads to faster convergence for gradient descent [100].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

In our experiments, following [26], all the original feature vectors are transformed to new vectors with dimension 300. The transformation matrix of WPCA is computed only using similar pairs from a training set, namely, 4800 feature vectors. Usually, data selection for WPCA does not significantly affect the performance after transformation. The only point worth noting is that we should select enough data, for example, under the experimental setting in this work, an appropriate number of feature vectors for whitening is more than 1000 [125, 26]. In our experiments, we use a MATLAB implementation of PCA provided by the PhD toolbox[5].

4.2.4 Experimental Settings

We apply the linear TSML method for face verification on the LFW dataset [75]. In this section, we introduce the scheme of our verification system in detail. All the experiments are performed under the LFW restricted configuration with label-free outside data: aligned images of the LFW-a dataset are adopted; only the provided 6000 pairs of data are used in evaluation (see Section 4.2.3).

Restricted Training

We experiment only on the restricted configuration of LFW, i.e. training and evaluating only on the specified 3000 similar pairs and 3000 dissimilar pairs. Concretely, all data of the restricted configuration are separated into 3 partitions: a training set, a validation set and a test set. We learn a model on the training set, choose the best model that achieves the highest performance on the validation set, and report the performance on the testing set using the best model.

We perform cross-validation on the 10 folds: there are overall 10 experiments, in each repetition, 4800 pairs from 8 folds are used for training, 600 pairs from another fold are used for validation and 600 pairs from the last fold are used for testing. For example, the first experiment uses subsets (1,2,3,4,5,6,7,8) for training, subset 9 for validation and subset 10 for testing; the second experiment uses (2,3,4,5,6,7,8,9) for training, subset 10 for validation and subset 1 for testing. At last, we report the mean accuracy ($\pm$standard error of the mean) of the 10 experiments.

Batch Gradient Descent

Under the setting of restricted training, in each of the 10 folds cross-validation, only 4800 pairs of 300-dimensional vectors are used for training. For this small training scale, advanced

[5]http://luks.fe.uni-lj.si/sl/osebje/vitomir/face_tools/PhDface/index.html

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

algorithms such as the Conjugate Gradient Descent (CGD) algorithm [115] and the Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm [108] can be used to perform batch gradient descent. Compared with the standard stochastic gradient descent algorithm, these advanced algorithm have no need to manually pick a learning rate and they are usually much faster in convergence. In our experiments, we used a MATLAB implementation of L-BFGS provided by Mark Schmidt [144] to minimize either the triangular loss function or the CSML cost function.

Matrix Initialization

In Equations (4.7) and (4.8), we need to specify the initialization matrix $\mathbf{W}_0$ before starting gradient descent. In our experiments, we use two kinds of initialization: the first one is the commonly used identity matrix $\mathbf{I}$ [125, 26], the second one is the Within Class Covariance Normalization (WCCN) matrix $\mathbf{T}$ that provides a better start. We directly set $\mathbf{W} = \mathbf{W}_0 = \mathbf{I}$ or $\mathbf{W} = \mathbf{W}_0 = \mathbf{T}$ before learning, and then optimize $\mathbf{W}$ by the L-BFGS algorithm.

WCCN has been well studies as Relevant Component Analysis (RCA) (see discussion on RCA in Chapter 2) [148, 7] or Intra-personal Subspace [166]. It has been used for object classification [7], speaker recognition [65] and face recognition [166]. Recently, it was introduced to improve the discrimination for pairwise face verification [8]. At first, a within class covariance matrix $\mathbf{C}$ is defined as:

$$\mathbf{C} = \frac{1}{t}\sum_{i=1}^{t}\frac{1}{m_i}\sum_{j=1}^{m_i}(\mathbf{x}_{ij}-\boldsymbol{\mu}_i)(\mathbf{x}_{ij}-\boldsymbol{\mu}_i)^T, \tag{4.9}$$

where t is the number of different classes, m_i is the number of instances in the ith class, $\mathbf{x}_{ij}$ is the jth instance in the ith class and $\boldsymbol{\mu}_i$ is the mean of the ith class. Decomposition on the matrix $\mathbf{C}$ produces eigenvalues $\lambda_1,\ldots,\lambda_k$ and eigenvectors $\mathbf{V} = \{\mathbf{v}_1,\ldots,\mathbf{v}_k\}$. And the WCCN matrix is:

$$\mathbf{T} = diag(\lambda_1^{-\frac{1}{2}},\ldots,\lambda_k^{-\frac{1}{2}})\mathbf{V}^T. \tag{4.10}$$

Under the restricted configuration of LFW, since we have no class information of any image, we regard each similar pair as a mini-class of its own for computing WCCN transformation [8]. It is worth noting that all the eigenvalues and eigenvectors are retained, i.e. we do not perform dimensionality reduction in this step.

Learning and Evaluation

Once we set up the initialization, we use the advanced optimization algorithm L-BFGS [108] to compute the optimal solution. Let $\mathbf{W}_\star$ denote the optimal matrix and $(\mathbf{a},\mathbf{b})$ represents

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

a pair of output vectors. Similarity between the two vectors can be measured by either the Triangular Similarity or the Cosine Similarity as we have proven their equivalence in Chapter 3. In our experiments, the final decision is made by comparing to a threshold γ: if $cos(\mathbf{a},\mathbf{b}) \geq \gamma$, $\mathbf{a}$ and $\mathbf{b}$ are similar; otherwise, they are dissimilar. We record the percentage of right decisions on the validation set, i.e.

$$accuracy = \frac{number\ of\ right\ decisions}{total\ number\ of\ pairs}. \tag{4.11}$$

After that, we select the best matrix $\mathbf{W}_\star$ and the best threshold γ that obtain the best accuracy on the validation set. At last, we report accuracy on the testing set using the best $A_\star$ and γ.

For the proposed TSML method, we have only two parameters to tune in the experiments: the regularization term λ and the decision threshold γ. The tuning range of λ was from 10^{-4} to 10^{-3} with a step size of 2×10^{-5}; the tuning range of γ was from -1 to 1 with a step size of 0.001. Without loss of generality, the length parameter r in Equation (4.8) is set to 1.

4.2.5 Results and Analysis

At the beginning, we directly perform evaluation on the 300-d whitened feature vectors, i.e. using the identity matrix $\mathbf{I}$ to realize a linear mapping. We consider this evaluation as the baseline. We also implemented state-of-the-art methods With-Class Covariance Normalization (WCCN) [26, 8] and CSML [125] in our experiments as a comparison. Concretely, four different methods were included in the first experiment:

- Baseline: evaluating directly on the whitened feature vectors, i.e. using the identity matrix $\mathbf{I}$ to realize a linear mapping;
- WCCN [26, 8]: performing WCCN on the whitened feature vectors and using the WCCN matrix $\mathbf{T}$ to realize a linear mapping;
- CSML-I: performing CSML on the whitened feature vectors with initialization matrix $\mathbf{W}_0 = \mathbf{I}$ and using the learned transformation matrix $\mathbf{W}_\star$ to realize a linear mapping;
- TSML-I: performing TSML on the whitened feature vectors with initialization matrix $\mathbf{W}_0 = \mathbf{I}$ and using the learned transformation matrix $\mathbf{W}_\star$ to realize a linear mapping.

Table 4.2 summarizes experimental results of the above four methods. We can see that all the three methods, WCCN, CSML-I, TSML-I, have obtained better results than the baseline, which means that all of them have learned a better transformation than the simple identity matrix. Generally, WCCN achieves the best performance on all the different kinds of features.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.2 Face verification accuracy (%) (±standard error of the mean) on LFW-a under the restricted configuration using four different methods: the baseline, WCCN, CSML-I, TSML-I. Dimension of the whitened feature vectors is 300.

Method	LBP		OCLBP		SIFT		Gabor	
	original	square root	original	square root	original	square root	original	square root
Baseline	77.17±0.49	79.73±0.38	80.43±0.25	81.55±0.44	76.88±0.42	77.52±0.49	75.28±0.45	77.25±0.32
CSML-I	79.30±0.33	82.78±0.39	82.58±0.43	85.08±0.57	81.60±0.32	82.37±0.35	78.03±0.55	80.00±0.69
TSML-I	78.35±0.49	81.80±0.56	82.23±0.51	84.38±0.43	80.85±0.56	81.40±0.47	76.27±0.59	79.27±0.46
WCCN	**80.40±0.39**	**84.23±0.33**	**83.75±0.51**	**86.83±0.37**	**82.72±0.39**	**84.17±0.25**	**78.68±0.62**	**81.52±0.65**

Table 4.3 Face verification accuracy (%) (±standard error of the mean) on LFW-a under the restricted configuration using five different methods: WCCN, CSML-Sim-I, CSML-Sim-WCCN, TSML-Sim-I, TSML-Sim-WCCN. Dimension of the whitened feature vectors is 300.

Method	LBP		OCLBP		SIFT		Gabor	
	original	square root	original	square root	original	square root	original	square root
WCCN	80.40±0.39	84.23±0.33	83.75±0.51	86.83±0.37	82.72±0.39	84.17±0.25	78.68±0.62	81.52±0.65
CSML-Sim-I	**83.18±0.71**	85.17±0.60	85.27±0.60	87.03±0.50	84.73±0.51	85.95±0.37	81.45±0.54	83.53±0.42
CSML-Sim-T	82.88±0.74	85.17±0.61	85.15±0.76	***87.18±0.46***	**85.00±0.53**	85.82±0.32	**81.62±0.61**	**83.58±0.52**
TSML-Sim-I	82.63±0.68	85.40±0.52	**85.27±0.73**	87.02±0.40	84.50±0.67	**86.08±0.44**	80.63±0.54	83.00±0.47
TSML-Sim-T	82.40±0.80	**85.58±0.58**	84.98±0.75	87.10±0.43	84.83±0.58	85.70±0.43	80.82±0.52	83.35±0.57

Table 4.4 Time cost (±standard error of the mean) in **milliseconds** of CSML-Sim-I and TSML-Sim-I on LFW-a under the restricted configuration. Rel. Impr.: Relative Improvement.

Method	LBP		OCLBP		SIFT		Gabor	
	original	square root	original	square root	original	square root	original	square root
CSML-Sim-I	91.38±1.23	92.46±1.40	92.26±0.76	93.71±1.11	91.42±0.62	93.30±1.03	92.13±0.70	90.96±0.78
TSML-Sim-I	73.41±0.71	73.79±0.62	74.74±1.03	74.77±0.79	74.69±0.44	74.75±0.64	74.64±0.86	74.45±0.72
Rel. Impr.	19.67%	20.19%	18.99%	20.21%	18.30%	19.88%	18.98%	18.15%

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

For example, WCCN obatins the highest accuracy of 86.83% using the square-rooted OCLBP face descriptor .

Learning on Similar Pairs Only

Why does WCCN obtain better results than CSML and TSML? The major difference between WCCN and the other two methods is that WCCN concerns only intra-personal variance but ignores the inter-personal information [166, 8, 26]. In other words, WCCN performs learning on similar pairs only but CSML and TSML take into account both similar and dissimilar pairs. Indeed, according to the setting of mutually exclusive training and testing in LFW, the identities in the validation and testing set are different from those in the training set, hence the discriminative information learned between individuals in the training set is not a good prediction of that between individuals in the testing set, i.e. a kind of over-fitting occurs. It is also shown in Appendix B that under the configuration of restricted training, similar pairs generally contribute more to the gradient than dissimilar pairs, resulting in better performance on pairwise face verification.

Consequently, like WCCN, we now train CSML and TSML on similar pairs only. Moreover, since the WCCN transformation matrix performs so well, we take it as another initialization besides the identity matrix. In summary, we experiment with four new different methods for pairwise face verification on LFW-a:

- CSML-Sim-I: training CSML on similar pairs only with initialization matrix $\mathbf{W}_0 = \mathbf{I}$ and using the learned transformation matrix $\mathbf{W}_\star$ to realize a linear mapping;
- CSML-Sim-T: training CSML on similar pairs only with initialization matrix $\mathbf{W}_0 = \mathbf{T}$ and using the learned transformation matrix $\mathbf{W}_\star$ to realize a linear mapping;
- TSML-Sim-I: training TSML on similar pairs only with initialization matrix $\mathbf{W}_0 = \mathbf{I}$ and using the learned transformation matrix $\mathbf{W}_\star$ to realize a linear mapping;
- TSML-Sim-T: training TSML on similar pairs only with initialization matrix $\mathbf{W}_0 = \mathbf{T}$ and using the learned transformation matrix $\mathbf{W}_\star$ to realize a linear mapping.

Table 4.3 summarizes experimental results of WCCN and the above four methods. Compared with WCCN, the metric learning methods CSML-Sim-I, CSML-Sim-T, TSML-Sim-I and TSML-Sim-T further improve the decision accuracies. For example, TSML-Sim-T gets 85.58%, which outperforms the 84.23% of WCCN on the square-rooted LBP.

Interestingly, comparing the two different kinds of initialization (Table 4.3: CSML-Sim-I vs. CSML-Sim-T, TSML-Sim-I vs. TSML-Sim-T), we observe that all of them obtain

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

comparable results. Remind that TSML-Sim-I takes the identity matrix as initialization and TSML-Sim-T takes the WCCN matrix as initialization. We deduce that either of the two initialization matrices is acceptable for the linear metric learning methods. The difference is that *different initializations imply different lower bounds*: for example, TSML-Sim-I sets the results of the baseline as the lower bound, and TSML-Sim-T sets the results of WCCN as the lower bound. However, though the two initializations set different lower bounds for learning, they may have comparable upper bounds, so we have observed similar results.

Efficiency of TSML

Comparing the TSML approaches with the CSML approaches, one may notice that they actually perform equally well on all kinds of face descriptors (Table 4.3). For example, on the square-rooted OCLBP descriptor, CSML-Sim-I, CSML-Sim-T, TSML-Sim-I and TSML-Sim-T obtain verification accuracies of 87.03%, 87.18%, 87.02% and 87.10%, respectively. However, our proposed linear TSML (Equation (4.8)) is theoretically more efficient than CSML. In order to prove that, we carry out a runtime comparison between CSML [125] and the proposed TSML.

We perform the efficiency comparison on all the 300-d whitened features: with fixed regularization parameter $\lambda = 0$, we calculate the gradient of CSML-Sim-I and TSML-Sim-I on the training set for only once and record their time consumption, respectively. Like the reported accuracy on face verification, we report the average time ($\pm$standard error of the mean) spent for 10 repetitions. This comparison was performed on a machine with a 4-core CPU, 8 GB RAM and 64-bit operating system. Table 4.4 summarizes the time cost on each feature in milliseconds. Generally, TSML relatively improves the efficiency by about 20% over CSML. For example, on the square-rooted OCLBP, calculating the gradient of CSML-Sim-I for once averagely costs 93.71 milliseconds, in contrast, calculating the gradient of TSML-Sim-I averagely costs only 74.77 milliseconds.

Comparison with the State-of-the-Art

Fusion on different descriptors generally leads to performance gain [125, 8, 26, 178, 69]. Therefore we perform fusion on the similarity scores of the proposed TSML approaches and compare it with the state-of-the-art methods (Table 4.5).

After learning a linear model, one can produce similarity scores for all the pairs of vectors on all possible descriptors. For each pair of vectors, all the corresponding similarity scores compose a new short vector which can be used to predict the final decision: a pair of vectors

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.5 Face verification accuracy (%) (±standard error of the mean) on LFW-a under the restricted configuration using different methods.

Method	Accuracy
DML-eig-fusion [178]	85.65±0.56
CSML-fusion [125]	88.00±0.37
PAF [176]	87.77±0.51
WCCN-fusion [8]	**91.10±0.59**
Sub-SML-fusion [26]	89.73±0.38
DDML-fusion [69]	**90.68±1.41**
LM3L [71]	89.57±1.53
TSML-fusion (this work)	**89.80±0.47**

is similar or not. We consider it as a two-class classification problem and employ a linear support vector machine (SVM) [27] to perform the classification.

Since LBP is a subpart of OCLBP [8], we abandon LBP and collect similarity scores on the other three descriptors and their square roots for fusion. Both of the two methods, TSML-Sim-I and TSML-Sim-T, are used to produce similarity scores. Thus we have totally 12 similarity scores for each pair, representing a new feature vector. After that, we train an SVM model on the validation set and make prediction on the test set. The fusion result of the proposed methods is 89.80%, which occupies the third position among the state-of-the-art methods under the restricted configuration (Table 4.5 and Fig. 4.1). Available ROC curves of more approaches are present in the result page of the LFW data set[6].

Other methods that perform well on the face verification problem (Table 4.5), for example WCCN-fusion, rely on a fusion of high-dimensional features (i.e. the 96520-d Scattering descriptor), or they are naturally slower because they perform more complex optimization, e.g. DDML-fusion [69] integrates deep neural networks with distance metric learning. Apart from fusion on multiple features, employing specific advanced processing steps to face verification can also produce promising results, such as the facial landmark extraction and 3D model fitting in PAF [176].

4.3 Pairwise Kinship Verification

The objective of pairwise kinship verification is to determine whether there is a kin relation between a pair of given face images. The kinship is defined as a relationship between two persons who are biologically related with overlapping genes. There are four different types of

[6] http://vis-www.cs.umass.edu/lfw/results.html#ImageRestrictedLF

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

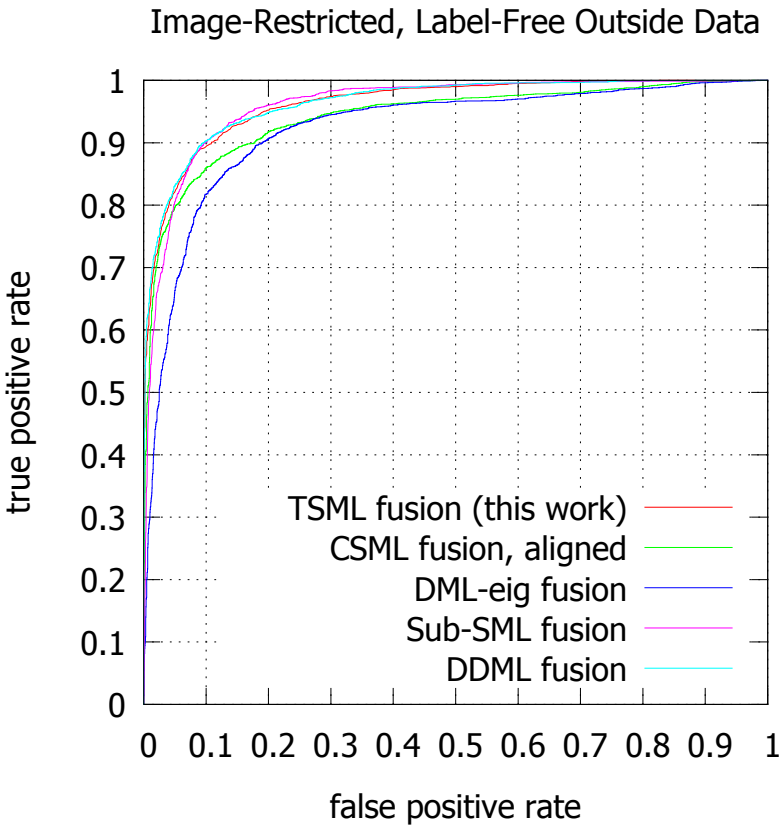


Fig. 4.1 ROC curves of the proposed TSML-fusion method (red line) and the other state-of-the-art methods on LFW under the restricted configuration with label-free outside data (LFW-a).

close kin relations: Father-Son (F-S), Father-Daughter (F-D), Mother-Son (MS) and Mother-Daughter (M-D). Comparing with face verification [75], kinship verification via faces is a relatively new problem in face analysis, but it can find many potential applications such as family album organization, genealogical research, missing family members search, and social media analysis. To conduct research along this direction, Lu *et al.* [114] constructed the Kinship Face in the Wild (KinFaceW) dataset for studying the problem of kinship verification from unconstrained face images.

4.3.1 The KinFaceW Protocols and Related Work

Following the LFW dataset [75], the KinFaceW dataset holds the setting of limited training data for some classes and the setting of mutually exclusive training and test sets. Three different settings are defined: (1) unsupervised setting, i.e. no labeled kin relation information is used; (2) image-restricted setting, i.e. only the given kin relation information is used in the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

training splits; (3) image-unrestricted setting, i.e. the identity information of the person is available to potentially form additional dissimilar/negative pairs for training.

Generally, all the metric learning methods used in face verification are also applicable for this kinship verification task. For example, Lu *et al.* [114] proposed Neighborhood Repulsed Metric Learning (NRML) as one of the earliest method applied on kinship verification and demonstrated that its performance was comparable to that of human observers. After that, the authors also investigated other approaches for kinship verification [174, 175, 71]. Two competitions have been organized to fairly evaluate and compare different kinship verification algorithms: one was held in conjunction with the International Joint Conference on Biometrics 2014, Clearwater, Florida, USA [113]; the other one was held in conjunction with the IEEE International Conference on Automatic Face and Gesture Recognition 2015, Ljubljana, Slovenia [112]. We participated in the second competition with the proposed linear TSML system and achieved the best performance under the image-restricted setting.

4.3.2 The KinFaceW Dataset and Face Descriptors

The KinFaceW dataset provides two kinship subsets: KinFaceW-I and KinFaceW-II. All the face images were collected from Internet, including some public figures as well as their parents or children. Face images were captured under uncontrolled environments in the two subsets with no restriction in terms of pose, lighting, background, expression, age, ethnicity, or partial occlusion. For ease of use, the data providers [114] manually labeled the coordinates of the eyes position of each face image, aligned and cropped facial regions into 64×64 to remove background.

There are four kin relations in the two subsets: Father-Son (F-S), Father-Daughter (F-D), Mother-Son (M-S), and Mother-Daughter (M-D). In KinFaceW-I, there are 156, 134, 116, and 127 pairs of kinship images for these four relations, respectively. For KinFaceW-II, each relation contains 250 pairs of kinship images. Apart from that the numbers of images in each subset are different, the major difference of KinFaceW-I and KinFaceW-II is that any two relative faces were acquired from different photos in KinFaceW-I but most relative faces in KinFaceW-II were captured from the same photo. In other words, environment conditions such as lighting differ more significantly between face pairs in KinFaceW-I than that in KinFaceW-II.

For the image-restricted setting, in addition to the similar pairs for each relation, equal numbers of dissimilar pairs were also provided for training. A dissimilar pair or a negative pair is generated by randomly combining each parent face image with a child image who is not his/her true child. Hence KinFaceW-I actually provides 1066 ($= 2 \times (156 + 134 + 116 + 127)$) data pairs for training and testing, and KinFaceW-II contains 2000 pairs of similar or

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

dissimilar face images. Finally, all the data pairs are recommended to be split into five non-overlapping folds to perform a 5-fold cross validation. For the image-unrestricted setting, more dissimilar pairs can be generated but the similar pairs are held the same. Like in the previous section of face verification, we focus on the image-restricted setting here.

Face Descriptors

To represent a face image in the KinFaceW dataset, we use four different face descriptors: LBP [1], Histogram of Gradients (HOG) [37], OCLBP [8] and Fisher Vectors [150]. Evaluation is directly performed on square roots of all the descriptors because the operation of square-rooting has been demonstrated to be useful [125, 8, 26, 150]. Details of feature extraction are listed as below.

Local Binary Patterns: we use the uniform LBP [128] to represent face images. The uniform LBP is denoted as $LBP^{u2}_{p,r}$, where $u2$ stands for 'uniform', (p,r) means to sample p points over a circle with a radius r. The dimension of an uniform pattern is 59. Concretely, each 64×64 image is divided into non-overlapping 8×8 blocks and uniform LBP patterns $LBP^{u2}_{8,1}$ are extracted from all the blocks. We concatenate all the LBP patterns into a single feature vector, whose dimension is 3776 $(=8\times8\times59)$.

Histogram of Gradients: we first divide each 64×64 image into non-overlapping 4×4 blocks, and then split the original image into non-overlapping 8×8 blocks. The 2-scale block division produces 320 $(=16\times16+8\times8)$ blocks in total. Subsequently, we extract a 9-dimensional HOG feature for each block and concatenate them into a single feature vector, whose dimension is 2880 $(=320\times9)$.

Over-complete Local Binary Patterns: we also used a variant of LBP, OCLBP, to improve the overall performance [184]. Unlike LBP, OCLBP adopts overlapping to adjacent blocks. Formally, the configuration of OCLBP is denoted as $S:(a,b,v,h,p,r)$: an image is divided into $a\times b$ blocks with vertical overlap of v and horizontal overlap of h, and uniform pattern $LBP^{u2}_{p,r}$ are extracted from all the blocks. Moreover, for the 64×64 images in the KinFaceW dataset, our OCLBP is composed of several different configurations: $S:(8,8,\frac{1}{2},\frac{1}{2},8,1),(12,12,\frac{1}{2},\frac{1}{2},8,2),(16,16,\frac{1}{2},\frac{1}{2},8,3)$. Concretely, the three configurations consider three block sizes: $8\times8,12\times12,16\times16$ with half overlap rates along both the vertical and horizontal directions. We shift the block window to produce overlaps. Taking the 8×8 block window for example, with the shifting step $8\times\frac{1}{2}=4$ to the left and downwards, the total number of 8×8 blocks is $(\frac{64}{4}-1)\times(\frac{64}{4}-1)=225$. Similarly, shifting the 12×12 window produces 100 blocks and shifting the 16×16 window produces 49 blocks. The dimension of our OCLBP vectors is $22,066$ $((225+100+49)\times59)$, where 59 is the dimension of the uniform LBP in each block.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Fisher Vectors: Fisher Vector (FV) faces [150] is one of the state-of-the-art descriptors used in image classification, so we adopt it here for kinship verification. The FV construction starts by extracting dense SIFT [111] from images. Specifically, for the 64×64 images in the KinFaceW dataset, 12×12 pixels patches are sampled with a stride of one pixel and for each patch a 128-dimensional SIFT representation is computed. After that, all the SIFTs are passed through square-rooting and L2 normalization to let all the vectors have unit length. Following [150], this process is repeated at five scales, with a scaling factors of $\sqrt{2}$ on each image, resulting in an 128×4397 dimensional feature vector. Moreover, PCA is applied to the square-rooted SIFTs, reducing its dimensionality from 128 to 64. Since introducing spatial information has been proved to be useful for improving the performance of image descriptor [150], we augment the PCA-SIFT vectors with their spatial coordinates: $[\frac{x}{w} - \frac{1}{2}; \frac{y}{h} - \frac{1}{2}]$, where (x, y) is the patch center of the corresponding SIFT vector, w and h are the width and height of the face image. Finally, the size of the dense features for each image is 66×4397. To simplify reproducibility, we use the VLFEAT package [163] to perform non-linear FV encoding on these dense feature vectors. We first use the *vl_gmm* command in the VLFEAT package to train a Gaussian Mixture Model with 512 Gaussians on the dense features. Based on the GMM, we call the *vl_fisher* command to compute the FV coefficients for each image. The dimension of an FV face is 67,584. Once again, to improve the final performance, all the FV faces are passed through square-rooting and L2 normalization.

4.3.3 Experiments and Analysis

We apply the same linear TSML method in Section 4.2.2 for kinship verification on the KinFaceW dataset [114]. All the experiments are carried out under the image-restricted configuration: only the specific pairs of training data are used in evaluation. A MATLAB implementation of L-BFGS [108, 144] is used to perform batch gradient descent for optimizing the triangular loss function, where the mapping matrix is initialized with the WCCN matrix (see Section 4.2.4). Due to limited data in the KinFaceW dataset, no validation set is separated and the best decision accuracy on the testing set is recorded. We report the average accuracy of a 5-fold cross validation as the final score for each method.

We first applied WPCA to whiten the raw feature vectors and reduce the vector dimension to 100, we evaluated on the whitened feature vectors and regarded it as the baseline solution [125, 8]. The transformation matrix of WPCA is computed using all the samples in the training set. We also implemented state-of-the-art methods WCCN [26, 8] as a comparison. Concretely, three different methods are included in this experiment:

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.6 Kinship verification accuracy (%) on (a) KinFaceW-I and (b) KinFaceW-II under the restricted configuration using different methods with different features.

(a) Results on KinFaceW-I

Feature	Method	F-S	F-D	M-S	M-D	Mean
Square-rooted LBP	Baseline	78.20	69.05	72.01	74.13	73.35
	WCCN	82.08	75.34	76.68	77.99	78.02
	TSML-Sim-T	82.08	75.73	77.12	77.99	**78.23**
Square-rooted HOG	Baseline	77.26	69.39	69.75	79.61	74.00
	WCCN	83.04	73.50	74.09	83.47	78.53
	TSML-Sim-T	82.73	73.49	75.40	83.47	**78.77**
Square-rooted OCLBP	Baseline	77.89	70.14	71.92	78.04	74.50
	WCCN	81.75	77.22	77.95	82.70	79.91
	TSML-Sim-T	82.08	77.61	78.39	82.30	**80.09**
Square-rooted FV	Baseline	78.84	76.50	77.10	80.24	78.17
	WCCN	82.72	80.63	81.87	84.58	82.45
	TSML-Sim-T	83.04	80.63	82.30	84.98	***82.74***

(b) Results on KinFaceW-II

Feature	Method	F-S	F-D	M-S	M-D	Mean
Square-rooted LBP	Baseline	76.40	67.80	73.60	73.00	72.70
	WCCN	84.80	78.20	81.80	78.20	80.75
	TSML-Sim-T	84.80	78.40	82.00	79.40	**81.15**
Square-rooted HOG	Baseline	75.80	67.40	74.00	72.20	72.35
	WCCN	83.80	77.00	81.00	78.80	80.15
	TSML-Sim-T	84.60	77.60	81.00	80.00	**80.80**
Square-rooted OCLBP	Baseline	78.60	70.20	75.60	75.00	74.85
	WCCN	87.60	79.80	82.40	81.40	82.80
	TSML-Sim-T	87.80	80.40	83.60	81.80	**83.40**
Square-rooted FV	Baseline	81.40	72.60	77.60	79.60	77.80
	WCCN	89.20	83.20	86.20	85.00	85.90
	TSML-Sim-T	89.40	83.60	86.20	85.00	***86.05***

Table 4.7 Participants in the FG 2015 Kinship Verification Evaluation. Our team is labeled as LIRIS, indicating results of the TSML-Sim-T method.

Team	Country	Label
Politecnico di Torino	Italy	Polito
LIRIS, University of Lyon	France	**LIRIS**
Universidad de Las Palmas de Gran Canaria	Spain	ULPGC
Nanjing University of Aeronautics and Astronautics	China	NUAA
Bar Ilan University	Israel	BIU

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.8 Kinship verification accuracy (%) on KinFaceW-I under the restricted configuration.

Team	F-S	F-D	M-S	M-D	Mean
Polito	85.30	85.80	87.50	86.70	**86.30**
LIRIS	83.04	80.63	82.30	84.98	82.74
ULPGC	71.25	70.85	58.52	80.89	70.01
NUAA	86.25	80.64	81.03	83.93	82.96
BIU	86.90	76.48	73.89	79.75	79.25
SILD (LBP)	78.22	69.40	66.81	70.10	71.13
SILD (HOG)	80.46	72.39	69.82	77.10	74.94

Table 4.9 Kinship verification accuracy (%) on KinFaceW-II under the restricted configuration in the FG 2015 Kinship Verification Evaluation.

Team	F-S	F-D	M-S	M-D	Mean
Polito	84.00	82.20	84.80	81.20	83.10
LIRIS	89.40	83.60	86.20	85.00	**86.05**
ULPGC	85.40	75.80	75.60	81.60	80.00
NUAA	84.40	81.60	82.80	81.60	82.50
BIU	87.51	80.82	79.78	75.63	80.94
SILD (LBP)	78.20	70.00	71.20	67.80	71.80
SILD (HOG)	79.60	71.60	73.20	69.60	73.50

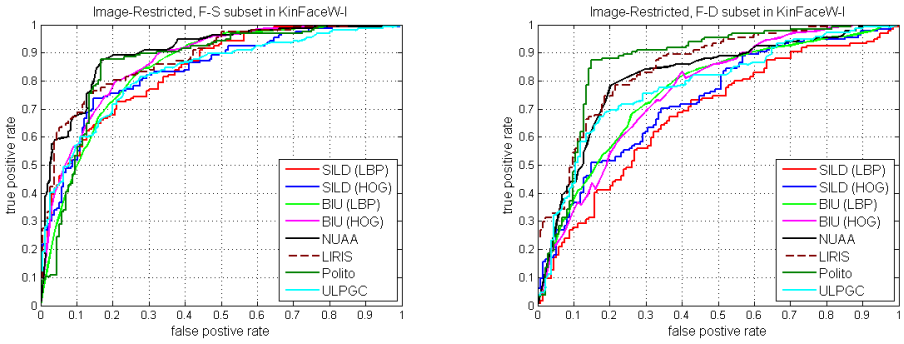
- Baseline: performing WPCA on the raw feature vectors;
- WCCN: performing WCCN on the whitened feature vectors;
- TSML-Sim-T: training TSML on similar pairs only with initialization matrix $\mathbf{W}_0 = \mathbf{T}$, where $\mathbf{T}$ is the WCCN matrix.

Tables 4.6 summarizes all the experimental results on the two datasets KinFaceW-I and KinFaceW-II. The proposed TSML method using FV faces achieves superior performance on the problem of kinship verification. For example, TSML-Sim-T obtains 86.05% on the KinFaceW-II dataset, which has even surpassed the human ability [114]. The other method, WCCN, has also achieved better results than the baseline model.

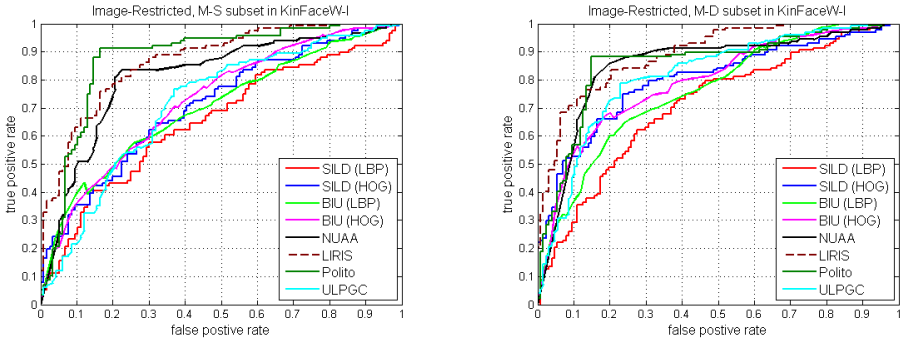
Comparison with the State-of-the-Art

The proposed TSML method is compared with state-of-the-art methods in the FG 2015 Kinship Verification in the Wild Evaluation [112]. Table 4.7 shows the five teams participating in this evaluation, note that our team is labeled as LIRIS, indicating results of the TSML-Sim-T method. Tables 4.8 and 4.9 present the verification accuracies of different methods

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

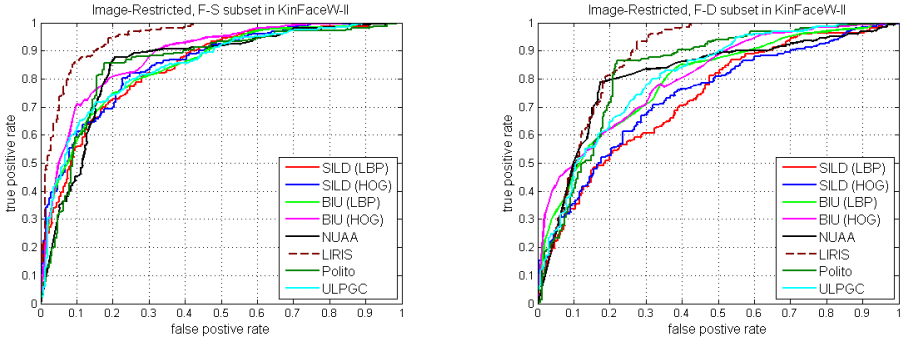


(a) F-S (b) F-D

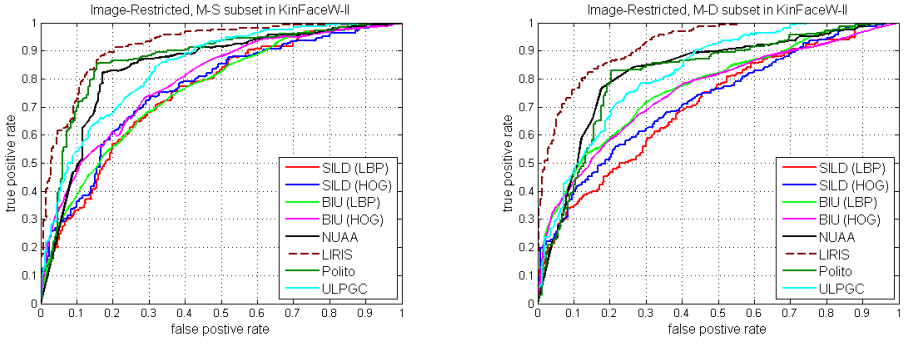


(c) M-S (d) M-D

Fig. 4.2 ROC curves of different methods under the image-restricted setting on KinFaceW-I.



(a) F-S (b) F-D



(c) M-S (d) M-D

Fig. 4.3 ROC curves of different methods under the image-restricted setting on KinFaceW-II.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

on the two data sets, Fig. 4.2 and Fig. 4.3 picture ROC curves for all the results. We can see that compared with the baseline SILD method [80], all participated methods show better verification performance on the two data sets. In this image-restricted experiment, TSML-Sim-T (i.e. the team LIRIS) achieves the first place (86.03%) on KinFaceW-II and the third place (82.74%) on KinFaceW-I. This results indicates the effectiveness of linear TSML on kinship verification as well as that on face verification in the previous section.

4.4 Linearity in Pairwise Verification

In previous two sections, we have shown the effectiveness of the linear TSML approach for pairwise face verification and pairwise kinship verification, where the two tasks deal with face images of human individuals. For the restricted setting on training, we argued that linearity indicates the property of generalization which reduces the risk of over-fitting. Though the linear model indeed achieves competitive performance with state-of-the-art methods, this argument has not been supported by experimental justification. In this section, we will directly compare linear models and nonlinear models by experiments.

Furthermore, previous experiments restricted the TSML systems to face images only, one may doubt that if the proposed method is also effective on other kinds of data. Therefore besides face verification, we will introduce another task of pairwise speaker verification, that takes speech utterances as the input.

Another interesting question is that compared with existing data-restricted experimental protocols, how much performance gain can be obtained by unrestricted training? In order to perform fair comparison of the restricted and unrestricted protocols, we will utilize stochastic gradient descent instead of the advance L-BFGS algorithm we used before. This is because the L-BFGS is only applicable for batch gradient descent in small-scale training and thus limits itself to the data-restricted setting only, but stochastic gradient descent can be applied for both restricted and unrestricted training.

The contributions of this section with respect to our previous work are the following:

- we establish a pairwise speaker verification protocol based on the data from the NIST 2014 i-Vector machine learning challenge, which is the first protocol for pairwise speaker verification. Both the pairwise face verification protocol of the LFW dataset and this speaker verification task aim at verifying identity information by individuals' biometric features.

- we present the TSML method in both linear and non-linear formulations for pairwise identity verification problems, i.e. pairwise face verification and pairwise speaker

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

verification. A comprehensive evaluation comparing the different formulations has shown that the linear model should be preferred due to its superior performance and its simplicity.

- we also study the influence of limited training data. Generally, compared with unlimited training, the limited case suffers from over-fitting. However, we find that with limited data, training the models on similar pairs only considerably reduces the effect of over-fitting.

4.4.1 Linear and Nonlinear Triangular Similarity Metric Learning

Firstly, we introduce the general TSML formulations. Given the i_{th} pair $(\mathbf{x}_i, \mathbf{y}_i)$ from a training set, a TSML system delivers it through a certain mapping function and produces two outputs $(\mathbf{a}_i, \mathbf{b}_i)$. Assuming the mapping function is parameterized by a set of weights $\mathbf{W}$, we have $\mathbf{a}_i = f(\mathbf{x}_i, \mathbf{W})$ and $\mathbf{b}_i = f(\mathbf{y}_i, \mathbf{W})$. Let $s_i = 1$ (respectively -1) denote that this pair is similar (respectively dissimilar), the general triangular loss for this pair is defined as:

$$J_i = \frac{1}{2}\|\mathbf{a}_i\|^2 + \frac{1}{2}\|\mathbf{b}_i\|^2 - r\|\mathbf{c}_i\| + r^2, \tag{4.12}$$

where r is a constant constraint on the vector length; and $\mathbf{c}_i = \mathbf{a}_i + s_i\mathbf{b}_i$. When the set of weights $\mathbf{W}$ is simply a transformation matrix, it is a linear TSML system; when the set of weights $\mathbf{W}$ represents nonlinear neural networks such as MLP, it is a nonlinear TSML system. Whichever system we choose for learning a metric, the gradient of the triangular loss with respect to the parameter set $\mathbf{W}$ is the same:

$$\frac{\partial J_i}{\partial \mathbf{W}} = (\mathbf{a}_i - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\frac{\partial \mathbf{a}_i}{\partial \mathbf{W}} + (\mathbf{b}_i - r\frac{s_i\mathbf{c}_i}{\|\mathbf{c}_i\|})\frac{\partial \mathbf{b}_i}{\partial \mathbf{W}}. \tag{4.13}$$

From the point of view of neural networks, different mapping functions are 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 $\mathbf{W}$. For a given input $\mathbf{z} \in R^d$, the output is simply $f(\mathbf{z}, \mathbf{W}) = \mathbf{W}\mathbf{z}$, which is exactly the case we have discussed in Section 4.2.2. According to Equation (4.2), differential of the

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

parameter matrix with respect to the i_{th} pair is:

$$\frac{\partial J_i}{\partial \mathbf{W}} = (\mathbf{a}_i - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\mathbf{x}_i^T + (\mathbf{b}_i - s_i r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\mathbf{y}_i^T. \tag{4.14}$$

Single layer of nonlinear neurons

Besides the parameter matrix $\mathbf{W}$, nonlinear neurons involve a bias term, and a nonlinear activation function, e.g. the tanh function [100]. For a given input $\mathbf{z} \in R^d$, the output is:

$$f(\mathbf{z}, \mathbf{W}) = tanh(\mathbf{W}\mathbf{z} + \mathbf{h}), \tag{4.15}$$

where $\mathbf{h}$ denotes the bias term of the neurons. This equation can be rewritten as:

$$f(\mathbf{z}', \mathbf{W}') = tanh(\mathbf{W}'\mathbf{z}'), \tag{4.16}$$

where $\mathbf{z}' = [\mathbf{z}; 1]$ and $\mathbf{W}' = [\mathbf{W} \ \ \mathbf{h}]$. Remind that derivative of the tanh function is $tanh'(\mathbf{z}) = 1 - tanh^2(\mathbf{z})$. Based on the differential of the linear case in Equation (4.14), differential of the parameters $\mathbf{W}' : \{\mathbf{W}, \mathbf{h}\}$ here is:

$$\begin{aligned} \frac{\partial J_i}{\partial \mathbf{W}'} &= (\mathbf{a}_i - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})\frac{\partial \mathbf{a}_i}{\partial \mathbf{W}'} + (\mathbf{b}_i - r\frac{s_i\mathbf{c}_i}{\|\mathbf{c}_i\|})\frac{\partial \mathbf{b}_i}{\partial \mathbf{W}'} \\ &= (1 - \mathbf{a}_i \odot \mathbf{a}_i) \odot (\mathbf{a}_i - r\frac{\mathbf{c}_i}{\|\mathbf{c}_i\|})[\mathbf{x}_i; 1]^T + (1 - \mathbf{b}_i \odot \mathbf{b}_i) \odot (\mathbf{b}_i - r\frac{s_i\mathbf{c}_i}{\|\mathbf{c}_i\|})[\mathbf{y}_i; 1]^T, \end{aligned} \tag{4.17}$$

where the notation $\odot$ means element-wise multiplication. Details of this equation is referred to the chain rule used in Backpropagation [100].

Multiple layers of nonlinear neurons

A single layer of neurons is hard to represent the power of nonlinear mapping. But 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 [142]. Thus we adopt a 3-layer MLP, containing one input layer and two layers of nonlinear neurons, to realize the nonlinear mapping.

With a parameter set $\mathbf{W} : \{\mathbf{W}^{(1)}, \mathbf{h}^{(1)}, \mathbf{W}^{(2)}, \mathbf{h}^{(2)}\}$, for a given input $\mathbf{z} \in R^d$, the output through the 3-layer MLP is:

$$f(\mathbf{z}, \mathbf{W}) = tanh(\mathbf{W}^{(2)}tanh(\mathbf{W}^{(1)}\mathbf{z} + \mathbf{h}^{(1)}) + \mathbf{h}^{(2)}). \tag{4.18}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Similarly with Equation (4.17), according to the chain rule in the Backpropagation algorithm [100], we can calculate differentials for each parameter $\{\mathbf{W}^{(1)}, \mathbf{h}^{(1)}, \mathbf{W}^{(2)}, \mathbf{h}^{(2)}\}$ with respect to a training pair. More details are referred to the discussion on MLP and Backpropagation in Chapter 2. For all the three linear and nonlinear TSML systems, we employ the same stochastic gradient descent to update their weights until reaching an optimal solution.

4.4.2 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 the Conjugate Gradient Descent (CGD) algorithm [115] and the Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm [108] could be used as well but their analysis would go beyond the scope of this section. We adopt *early-stopping* [134] to prevent 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* [100] term to speed up training. The momentum λ is empirically set to be 0.99 for all the experiments. Note that the input vectors will be passed through L2 normalization before training, i.e. the length of input vectors are normalized to 1. Without loss of generality, the length parameter r in the triangular loss function is set to 1.

Initializing the weights

For the linear mapping, like in [125, 26, 184], we initialize the transformation matrix with the identity matrix. For the nonlinear mappings, we use the normalized random initialization [53] that is considered to be helpful for the tanh networks. Concretely, weights of each layer are initialized with an uniform distribution as:

$$\{\mathbf{W}^{(j)}, \mathbf{h}^{(j)}\} \sim U[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}], \tag{4.19}$$

where $\{\mathbf{W}^{(j)}, \mathbf{h}^{(j)}\}$ 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.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Algorithm 1: Stochastic Gradient Descent for TSML

input : Training set; Validation set;
output: Parameter set $\mathbf{W}_\star$
paramters: 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**
  $\mathbf{W}_0 \leftarrow \mathbf{I}$; *% **I** is the identity matrix*
if *nonlinear mapping* **then**
  randomly initialize $\mathbf{W}_0$ according to Equation (4.19);
$\Delta\mathbf{W}_0 \leftarrow 0$;
Perform L2 normalization on the training set;
Perform L2 normalization on the validation set;
% optimization by back propagation
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 the cost J according to Equation (4.12);
  % back propagation
  Calculate the gradient $\frac{\partial J}{\partial \mathbf{W}_{t-1}}$ according to Equation (4.13);
  % updating using momentum
  $\Delta\mathbf{W}_t = \lambda\Delta\mathbf{W}_{t-1} + \frac{\partial J}{\partial \mathbf{W}_{t-1}}$;
  $\mathbf{W}_t \leftarrow \mathbf{W}_{t-1} + \alpha\Delta\mathbf{W}_t$;
  % checking on the validation set regularly
  if $(P_t \mod F_t) == 0$ **then**
    compute the Decision Accuracy according to Equation (4.20);
% output the best matrix on the validation set
$\mathbf{W}_\star \leftarrow$ the $\mathbf{W}_t$ gives the best result on the validation set;
return $W_\star$.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

4.4.3 Datasets and Feature Vectors

In order to validate the generality of the proposed TSML method, we carry out pairwise identity verification experiments on two datasets in different domains: the LFW image dataset for pairwise face verification [75] and the NIST i-vector dataset for pairwise speaker verification [58].

The LFW dataset

Recently, high-dimensional Fisher Vector (FV) faces, which combine dense feature sampling with improved Fisher Vector encoding, have achieved striking results on pairwise face verification [150]. We have also confirmed its superiority for the problem of pairwise kinship verification (see Section 4.3.3). Thus we use FV to represent face images in the current experiments.

The LFW dataset is the one we have used in Section 4.2. However, instead of using the LFW-a images [160], we concern cropped 150×150 'funneled' images of LFW [73] because data of FV faces are directly provided by [150] on the LFW-funneled images[7] (Data for the setting 3). The process of extracting FV descriptors is referred to Section 4.3.3, dimension of a FV vector is 67,584. Moreover, following [125, 8], in order to transform the high dimensional FV vectors into a new space of a tractable scale, we apply WPCA to reduce the vector dimension to 500.

The NIST i-vector dataset

For speaker recognition, most popular features are developed on generative models such as Gaussian Mixture Model-Universal Background Model (GMM-UBM) [139]. 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 [83]. Moreover, inspired by the joint factor analysis, a new feature called i-vector is developed [40]. Unlike JFA models the speaker variability in the high dimensional space of GMM supervectors, 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 directly applied to speaker verification [88].

We use the data of the NIST 2014 Speaker i-Vector Challenge [58], which consist of i-vectors derived from conversational telephone speech data in the NIST Speaker Recognition Evaluations (SRE's) 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

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

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.10 Distribution of individuals and speech utterances in the 10 subsets, where the individuals are mutually exclusive. Ind.: individuals, Utt.: utterances.

Index	1	2	3	4	5	6	7	8	9	10	Total
No. of Ind.	496	496	496	496	496	496	496	496	496	494	4958
No. of Utt.	3660	3664	3568	3741	3702	3566	3605	3636	3744	3686	36572

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 consist of a development set for building models and a test set for evaluation.

We only select the development data of this Challenge and establish an experimental protocol of pairwise speaker verification. There are 36,572 speech utterances in total in this experiment, belonging to 4,958 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 perform a 10-fold cross validation. Table 4.10 shows the distribution of individuals and speech utterances in the 10 subsets.

4.4.4 Experiments and Analysis

A. Experimental Setup

We carried out experiments on the LFW image dataset for pairwise face verification [75] and on the NIST i-vector dataset for pairwise speaker verification [58]. On both of the two datasets, we performed cross-validation on 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 4.4.2, some training data are separated as an independent validation set to do *early-stopping*.

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 compare different approaches on the same test data. Specifically, 600 image pairs are provided in each fold of the LFW dataset, where 300 are similar and the other 300 are dissimilar [75]. 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.

Restricted and unrestricted training: following [75], we adopted two different training settings in our experiments: the restricted setting in which only the fixed sample pairs are allowed for training; in contrast, under the unrestricted setting, it is possible to generate more training pairs since it is permitted to use the identity information.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.11 Proportion of the triplets $\{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}$ satisfying that $cos(\mathbf{x}_i, \mathbf{y}_i) > cos(\mathbf{x}_i, \mathbf{z}_i)$, using 1230 randomly selected triplets. The mapping is linear and trained on data of LFW under the restricted setting.

Status	Before mapping	After mapping
Proportion	92.85%	96.34%

Maximal decision accuracy: like the minimal Decision Cost Function (minDCF) in [58], we define a Decision Accuracy (DA) function to measure the overall verification performance on a set of data pairs:

$$DA(\gamma) = \frac{number\ of\ right\ decisions\ (\gamma)}{total\ number\ of\ pairs}, \tag{4.20}$$

where the threshold γ is used to make a decision on the Cosine Similarity values: $cos(\mathbf{a}, \mathbf{b}) > \gamma$ means $(\mathbf{a}, \mathbf{b})$ is 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 ($\pm$standard error of the mean) of the 10 experiments.

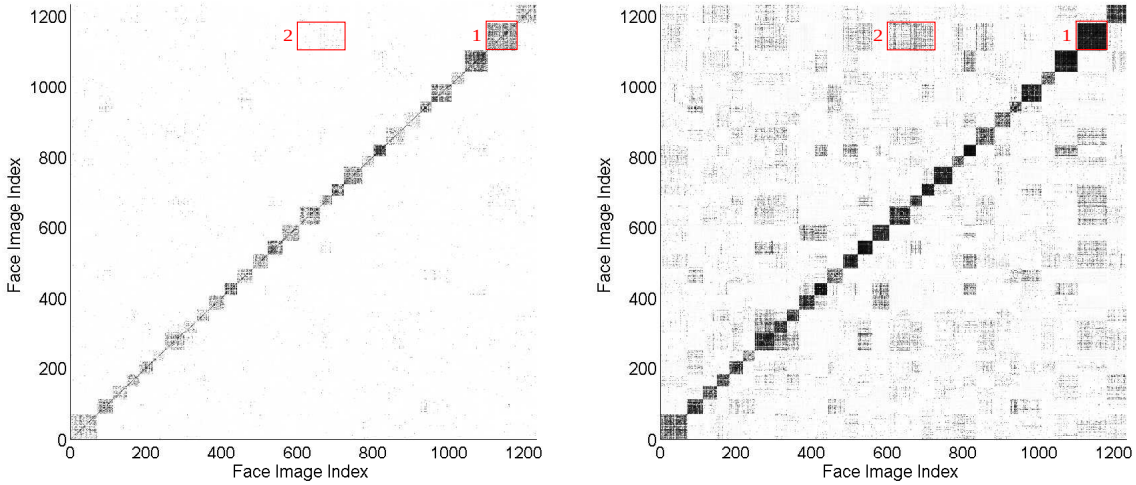
B. Learning a better metric

From the point of view of feature representation, the objective of Metric Learning is to learn a new vector representation $f(\mathbf{x})$ as a substitute of the original representation $\mathbf{x}$ that better suits a specific metric, e.g. the Triangular Similarity or the Cosine Similarity in this work. In this section, we illustrate the similarity changes before and after mapping. Concrete experiments will be given in later sections.

Taking the task of pairwise face verification for example, the original inputs are FV vectors representing pairs of face images, e.g. $(\mathbf{x}_i, \mathbf{y}_i)$. By minimizing the cost on some training data, we obtain an optimal mapping function $f(\mathbf{z}, \mathbf{W}_\star)$ that produces new vector representations in the target space, e.g. $(\mathbf{a}_i, \mathbf{b}_i)$. According to the defined objective, similarity values between similar face images in the target space should be larger than that in the original space (i.e. $cos(\mathbf{a}_i, \mathbf{b}_i) > cos(\mathbf{x}_i, \mathbf{y}_i)$ when $s_i = 1$), while similarity values between dissimilar face images in the target space should be smaller than that in the original space (i.e. $cos(\mathbf{a}_i, \mathbf{b}_i) < cos(\mathbf{x}_i, \mathbf{y}_i)$ when $s_i = -1$). Thus it is easier to distinguish a similar pair from a dissimilar pair in the target space than in the original space, which leads to better verification performance. For pairwise speaker verification, we simply substitute the FV face vectors with the i-vectors representing speech utterances.

We use two *dotplot* figures to illustrate the pairwise similarity values between vectors before and after mapping for face images from the LFW dataset (Fig. 4.4). The mapping

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) Before mapping (b) After mapping

Fig. 4.4 Dotplots illustrating pairwise similarity matrices between vectors (a) before mapping and (b) after mapping for some face images from the LFW dataset. Dark points mean high similarity and light points mean low similarity. The mapping is linear and trained on data of LFW under the restricted setting.

function here is the linear one: $f(\mathbf{z}) = \mathbf{W}\mathbf{z}$ and the pairwise similarity values are calculated by the Cosine Similarity: $cos(\mathbf{a},\mathbf{b}) = \frac{\mathbf{a}^T\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$. We train this linear model on the LFW data under restricted setting. Figure 4.4 (a) shows the pairwise similarity matrix before mapping, i.e. in the original space; Fig. 4.4 (b) shows the pairwise similarity matrix after mapping, i.e. in the target space. Note that high similarity values are represented by dark pixels.

We select individuals with 30-100 images from the LFW dataset and order these images in a sequence such that images from the same individuals are grouped together. There are 1230 images in total for all the 29 individuals. We can see that each dotplot figure contains dark square regions along the diagonal, showing the high within-individual similarity values. In contrast, regions away from the diagonal represent the between-individual similarity values which are explicitly brighter. This implies that in both of the two dotplots, most of the dissimilar pairs have smaller similarity values than the similar pairs.

Comparing the dark blocks in the two dotplot figures (e.g. region 1), there are clear holes within the dark blocks in the original space (Fig. 4.4 (a)). But after mapping into the target space, the dark blocks are more salient along the diagonal (Fig. 4.4 (b)), which implies that the within-individual similarities have been strengthened by the mapping. Besides, comparing the non-diagonal regions (e.g. region 2) in the two dotplot figures, it is strange

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

that some of the between-individual similarities have also been strengthened despite our objective to weaken them.

However, it is indeed easier to distinguish a similar pair from a dissimilar pair in the target space than in the original space. To verify this, we define a triplet $\{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}$ where $\mathbf{x}_i$ and $\mathbf{y}_i$ represent two face images of the same individual, $\mathbf{z}_i$ represents a face image from a different individual. Ideally, we suppose that the within-individual similarity $cos(\mathbf{x}_i, \mathbf{y}_i)$ is always larger than the between-individual similarity $cos(\mathbf{x}_i, \mathbf{z}_i)$. We randomly select 1230 triplets from the similarity matrices in Fig. 4.4. Table 4.11 shows that the proportion of the triplets $\{\mathbf{x}_i, \mathbf{y}_i, \mathbf{z}_i\}$ satisfying $cos(\mathbf{x}_i, \mathbf{y}_i) > cos(\mathbf{x}_i, \mathbf{z}_i)$. We can see that the proportion has been significantly increased by the mapping (from 92.85% to 96.34%), which implies the effectiveness of the proposed TSML mapping.

Figure 4.4 and Table 4.11 have shown that we can learn a better metric for the training data. By using early stopping, we guarantee the learned metric to be also effective on the validation and test sets. Figure 4.5 (a) shows the learning curve of the linear model used in Fig. 4.4. The blue line shows the learning curve on the validation set, and the red line denotes the learning curve on the test set. According to *early-stopping*, only the results at the peak of the validation learning curve are retained as performance reporting. We can see that all the recorded accuracies of the training, validation and test sets are higher than that at the beginning, in other words, a better metric is indeed obtained for all the data.

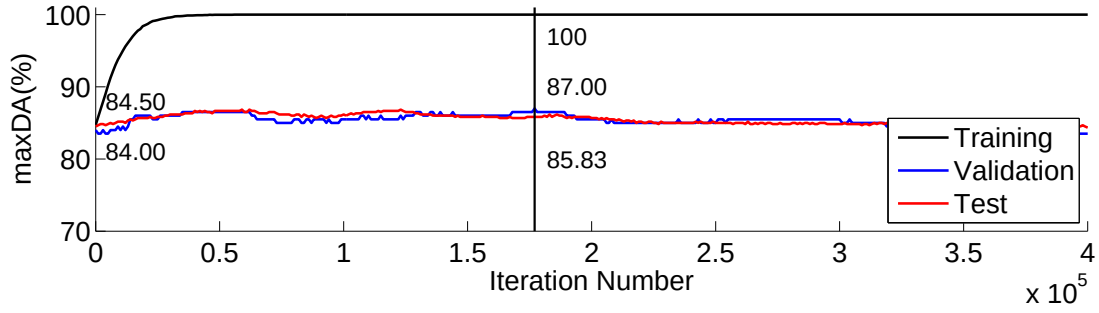
However, when the learning curve of the training set rises quickly to the top (i.e. 100%), the learning curves of the validation and test sets only climb up a little. The large gap between the accuracies of the training and test sets is so called the *over-fitting gap*. In the following sections, we will present several techniques to reduce the over-fitting gap and improve the verification performance on the test set.

C. Experimental results

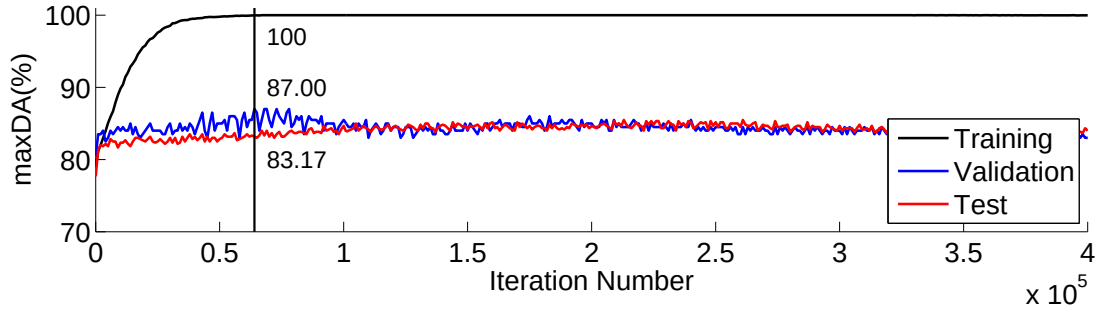
At the beginning, we directly calculated maxDA scores on the whitened feature vectors, i.e. the 500-dimensional FV vectors for the LFW dataset and 600-dimensional i-vectors for the NIST i-vector dataset. We consider this evaluation as the baseline. According to the different neuron models defined in Section 4.4.1, we evaluated three metric learning approaches in the experiments:

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

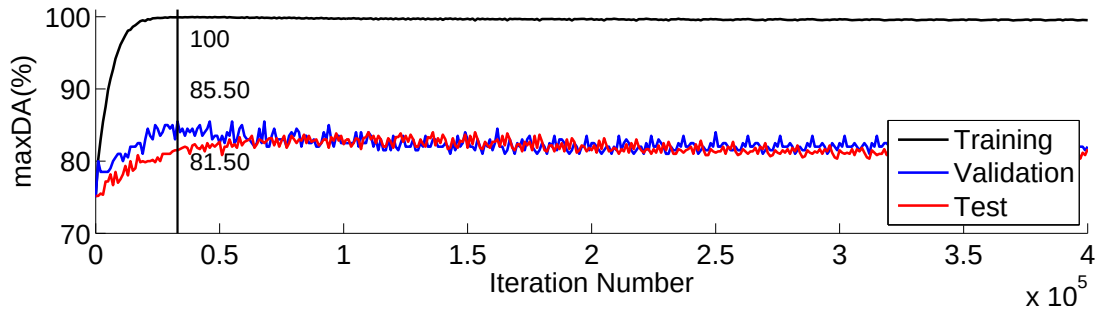
Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) Learning curve of TSML-Linear



(b) Learning curve of TSML-Nonlinear



(c) Learning curve of TSML-MLP

Fig. 4.5 Learning curves of different TSML models. Curves on the training, validation and test sets are represented by black, blue and red lines, respectively. All the models are trained on the LFW data under the restricted setting. According to *early stopping*, the vertical line indicates the model having the best performance on the validation set. Without any additional regularization techniques, the more complex the learning model is, i.e. having more parameters, the larger the over-fitting gap is.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

All these models are trained on both similar and dissimilar pairs. Results on the LFW-funneled dataset and the NIST i-vector dataset are summarized in Tables 4.12 and 4.13, respectively. We also re-implement the state-of-the-art WCCN method as a comparison.

The first phenomenon we can observe is that unrestricted training produces better results than restricted training. More training data bring up an accuracy improvement of about 5%, e.g. from 86.23% to 91.43% by TSML-Nonlinear on the LFW dataset. We have known since mid-seventies [154, 93, 99] that many methods increase in accuracy with increasing training data until they reach optimal performance. Generally, more training data better capture the underlying distribution of the whole dataset and thus reduce the over-fitting gap between training and test.

The second observation is that the linear model, TSML-Linear, performs better than the nonlinear models, TSML-Nonlinear and TSML-MLP. For example, TSML-Linear obtains an accuracy of 89.78% but TSML-MLP only gets 84.88% on the Nist i-vector dataset. Specifically, more parameters (i.e. additional bias terms or/and more layers of neurons) and the nonlinearity make the two nonlinear models more powerful to adapt themselves to the training data. However, without any additional anti-over-fitting techniques, generalization to the test data is not guaranteed. Figure 4.5 shows the learning curves of the three models 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 [100] and Dropout [153], can be introduced to reduce the risk of over-fitting for such a slightly deeper nonlinear model, but their analysis would go beyond the scope of this thesis. 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.

D. Learning on similar pairs only for restricted training

In previous sections [184], we have shown that learning on similar pairs only improves the verification performance under the restricted training. Hence we train the proposed three models on similar pairs only, namely, TSML-Linear-Sim, TSML-Nonlinear-Sim and TSML-MLP-Sim. The results are also shown in Tables 4.12 and 4.13. Figure 4.6 compares the performance of TSML and TSML-Sim on the LFW-funneled dataset and the NIST i-vector datset, respectively, where 'TSML' denotes the three standard models that trains on both similar and dissimilar pairs, 'TSML-Sim' means learning on similar pairs only.

From the right bars in the two figures, under the unrestricted configuration, we can see that the TSML-Sim models perform comparable with the TSML models. This implies that

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.12 Mean maxDA scores (±standard error of the mean) of pairwise face verification on the LFW-funneled dataset. '-Sim' means learning on similar pairs only.

Approaches	Restricted Training	Unrestricted Training
Baseline	84.83±0.38	
WCCN [8, 26]	91.10±0.45	91.17±0.36
TSML-Linear	87.95±0.40	92.03±0.38
TSML-Nonlinear	86.23±0.39	91.43±0.52
TSML-MLP	84.10±0.45	89.30±0.73
TSML-Linear-Sim	**91.90±0.52**	**92.40±0.48**
TSML-Nonlinear-Sim	90.58±0.52	91.47±0.37
TSML-MLP-Sim	88.98±0.64	89.03±0.58

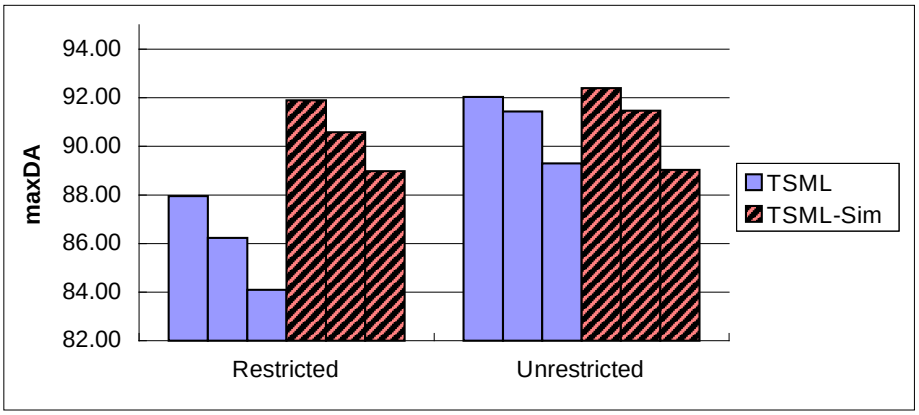
Table 4.13 Mean maxDA scores (±standard error of the mean) of pairwise speaker verification on the NIST i-vector dataset. '-Sim' means learning on similar pairs only.

Approaches	Restricted Training	Unrestricted Training
Baseline	87.78±0.39	
WCCN [8, 26]	91.69±0.29	91.97±0.33
TSML-Linear	89.78±0.25	93.97±0.20
TSML-Nonlinear	87.43±0.31	93.11±0.20
TSML-MLP	84.88±0.24	90.21±0.36
TSML-Linear-Sim	**92.94±0.15**	**93.99±0.24**
TSML-Nonlinear-Sim	91.29±0.25	93.43±0.23
TSML-MLP-Sim	89.59±0.45	90.83±0.30

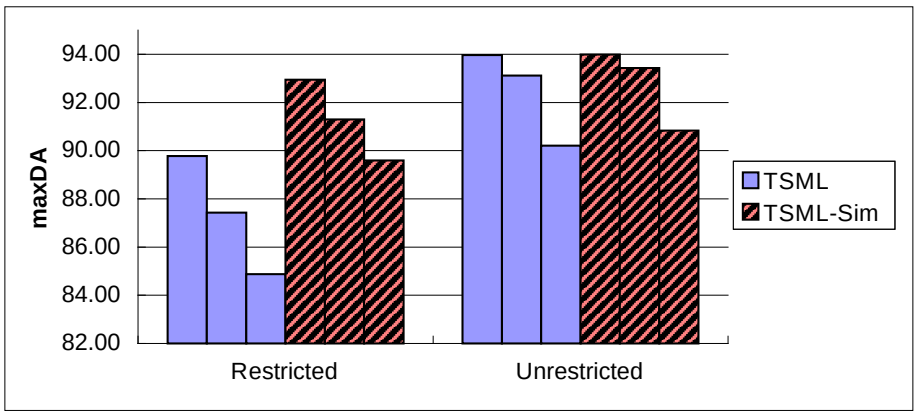
during training, only the similar pairs are adequate for learning a better metric. Moreover, under the restricted configuration, i.e. when the training data are limited, the TSML-sim models largely outperform the TSML models (see the left bars in Figure 4.6), which strongly evidences the substantial contribution of the similar pairs. In other words, it is the lack of enough dissimilar pairs that causes over-fitting problems for the TSML models.

Concretely, the setting of equal quantity of similar and dissimilar pairs is problematic for restricted training. Assuming 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 exponentially 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, we select only 300 similar pairs and 300 dissimilar pairs in each subset of the LFW dataset. As a consequence, learning on such limited dissimilar pairs

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) Results on LFW-funneled



(b) Results on NIST i-vector

Fig. 4.6 Performance comparison between TSML and TSML-Sim on the LFW-funneled dataset and the NIST i-vector dataset. 'TSML' denotes the four models that trains on both similar and dissimilar pairs, 'TSML-sim' means learning on similar pairs only.

causes serious over-fitting problems to the TSML models, that is why they perform worse than the TSML-Sim models. In contrast, when the training is unrestricted, enough dissimilar pairs can be covered during training. All similar pairs will be duplicated for many times but it does not matter.

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 to 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 [183], using hinge loss functions to filter invalid gradient descent from dissimilar pairs [69] or weighting the gradient contributions from similar and dissimilar pairs [125, 70].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 4.14 Comparison of TSML-Linear-Sim with other state-of-the-art results under the restricted configuration with no outside data on LFW-funneled.

Method	Accuracy
V1-like/MKL [133]	79.35±0.55
APEM (fusion) [102]	79.06±1.51
MRF-MLBP [3] (no ROC)	79.08±0.14
SVM-Fisher vector faces [150]	87.47±1.49
Eigen-PEP (fusion) [103]	88.97±1.32
Hierarchical-PEP (fusion) [101]	91.10±1.47
MRF-Fusion-CSKDA [4] (no ROC)	**95.89±1.94**
TSML-Linear-Sim (this work)	***91.90±0.52***

Table 4.15 Comparison of TSML-Linear-Sim with other methods using single face descriptor under the restricted configuration with no outside data on LFW-funneled.

Method	Feature	Accuracy
MRF-MLBP [3]	multi-scale LBP	79.08±0.14
APEM [102]	SIFT	81.88±0.94
APEM [102]	LBP	81.97±1.90
Eigen-PEP [103]	PEP	88.47±0.91
Hierarchical-PEP [101]	PEP	90.40±1.35
SVM [150]	Fisher Vector faces	87.47±1.49
WCCN [8]	Fisher Vector faces	91.10±0.45
TSML-Linear-Sim	Fisher Vector faces	**91.90±0.52**

E. 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 [74]. The comparison is summarized in Table 4.14, and the corresponding ROC curves are shown in Fig. 4.7. The curves of MRF-MLBP [3] and MRF-Fusion-CSKDA [4] are missing because the curve data are not provided on the public result page[8]. We can see that MRF-Fusion-CSKDA occupies the first place and the proposed TSML-Linear-Sim takes the second one with a 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 [4]. However, the proposed TSML-Linear-Sim method is linear and simple as it has only utilized a single feature, the FV vectors.

[8]http://vis-www.cs.umass.edu/lfw/results.html#ImageRestrictedNo

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

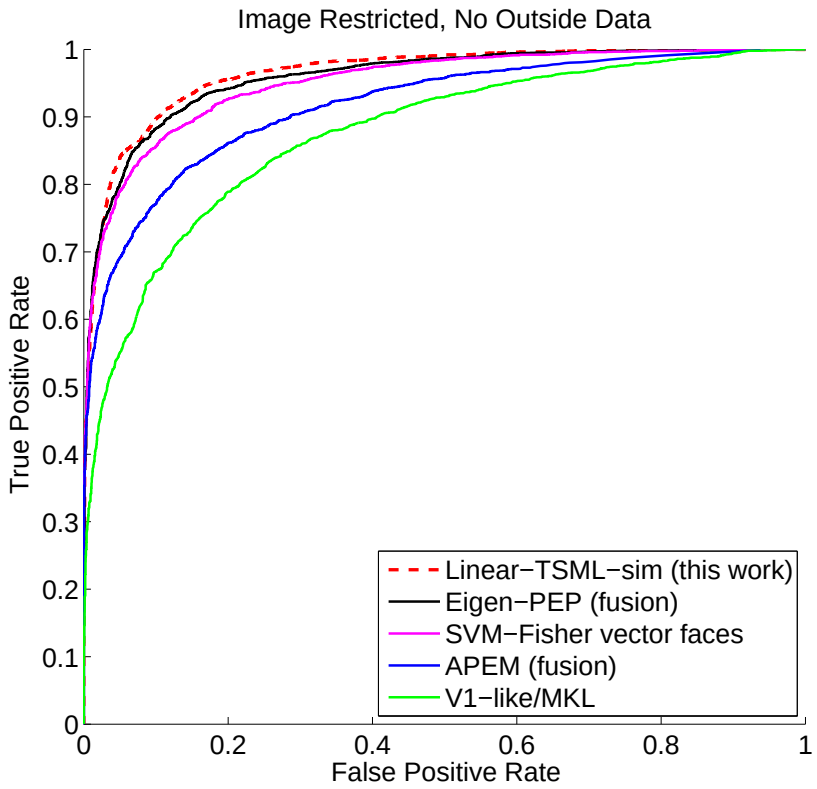


Fig. 4.7 ROC curves of Linear-TSML-Sim (red dashed line) and other state-of-the-art methods on the LFW dataset under the restricted configuration with no outside data.

Thus we collected the results of methods using a single feature in Table 4.15. Especially, we also applied another state-of-the-art approach WCCN [8] 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 Machine (SVM) method [150] on the FV vectors by 4.43% points (from 87.47% to 91.90%).

4.5 Conclusion

In this chapter, we have applied the proposed linear TSML method to a series of pairwise verification problems such as pairwise face verification, pairwise kinship verification and pairwise speaker verification. We presented a simple but effective solution of learning a linear model on similar pairs only.

We first pointed out that the setting of limited training data for some classes and the setting of mutually exclusive training and test sets make such a pairwise verification task

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

suffering from over-fitting problems. By extensive experimental validation, we presented several strategies and confirmed their effectiveness on reducing the risk of over-fitting:

- **More training pairs**: compared with restricted training only allows to use the specified training pairs in a dataset, unrestricted training covers enough dissimilar pairs and thus protect the models from over-fitting to a small portion of training data.

- **Linearity**: deep structure of more parameters and the nonlinearity make nonlinear models more powerful to adapt themselves to the training data. However, without any additional anti-over-fitting techniques, generalization to the test data is not guaranteed. In contrast, linearity acts like a simple generalization strategy that makes the linear model better fit to unseen data in the validation and test sets.

- **Learning on similar pairs only for restricted training**: under the data-restricted setting, training on equal quantity of similar and dissimilar pairs does not accord with the ratio of similar and dissimilar pairs in practice. Instead of getting more training data, we have shown that discarding the dissimilar pairs is a good way to avoid over-fitting to the few dissimilar pairs and to learn a better metric.

- **Early stopping**: when we use stochastic gradient descent for optimization, we employ early stopping to guarantee the learned metric to be also effective on the validation and test sets. Since both the validation and test sets are unseen data in the training set, evaluation results on the two sets are usually correlated and the model achieving the best performance on the validation set usually performs well on the test set.

- **Regularization factor**: when we use batch gradient descent by an advanced optimization algorithm such as L-BFGS, we put in a regularization factor to prevent over-fitting. Concretely, for linear models, the regularization factor constrains the learned matrix to be close to a specified initialization matrix (see Section 4.2.2); for nonlinear models, Weight decay [100] is usually adopted to control the weights of neural networks, in order to reduce the large over-fitting gap between training and test.

With these strategies, the nature of learning a good metric makes the TSML method effective on all the different pairwise verification tasks. It presented competitive performance with the state-of-the-art methods in all the experiments. Especially, it has achieved the best result on the KinFaceW-II dataset in the FG 2015 Kinship Verification Evaluation [112].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Chapter 5

Applications on Classification and Dimensionality Reduction

5.1 Introduction and Related Work

In the previous chapter, we have applied linear Triangular Similarity Metric Learning (TSML) for a specific task of pairwise verification, which is different from traditional classification tasks. In machine learning and statistics, classification or identification is a classical problem of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known [2][1].

Compared with the setting of inadequate training data for some categories (i.e. classes) and the setting of mutually exclusive training and test sets in pairwise verification, a traditional classification task usually assumes enough training data for every class and most of test data belonging to a certain category in the training set. One of the earliest and simplest methods for classification is the k-Nearest Neighbors (kNN) classifier [34]. kNN classifies each unlabeled example by the majority label of its k nearest neighbors in the training set. Despite its simplicity, kNN often yields competitive results on many practical problems. Since performance of kNN classification heavily depends on the metric used to compute distances or similarities between examples, many metric learning methods have been proposed to improve classification performance of kNN [169, 55, 136, 52, 39, 60].

Generally, though the application changes from verification to identification, the objective of learning a metric from data pairs remains the same. In other words, we will still aim at

[1] https://en.wikipedia.org/wiki/Statistical_classification

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

specifying a mapping function to project data from the original space to the target space, and minimizing the triangular loss for all data pairs in the target space.

Especially, when we set the dimension of the target space lower than that of the original space, the process of mapping performs dimensionality reduction. Moreover, if the target space is a visualizable space, i.e. the dimension of the target space is lower than 3 so that one can see the objects in it, this particular kind of dimensionality reduction is also called data visualization. However, performing dimensionality reduction usually causes loss of information in the original data, thus we will face a challenging problem of keeping accurate classification while reducing the target dimension.

In Chapter 3, we have visualized data mapping of some toy data from specific Gaussian distributions. In this chapter, we will use our TSML models for classification and dimensionality reduction on practical data, including both small-scale and large-scale datasets. Concretely, we utilize the triangular loss function as the objective function for all different problems. In terms of the mapping function, we employ *neural networks* to perform nonlinear mappings: Multi-layer Perceptrons (MLP) for small-scale data; deep Convolutional Neural Networks (CNN) for large-scale data. Before introducing the details of our own methods, we first review the related literature of using neural networks for dimensionality reduction.

Neural Networks and Dimensionality Reduction

Using neural networks for dimensionality reduction is an old idea which has its origins in the late 1980's and early 1990's. The first work may be the Auto-Associate Neural Networks (AANN) [33, 56], a special type of MLP where the input and output layers have the same number of neurons, and the middle hidden layer has fewer neurons than the input and output layers. The objective of AANN is to reproduce the input pattern at its output. Thus it actually learns a mapping on the input patterns into a lower-dimensional space and then an inverse mapping to reconstruct the input patterns. Since it does not need the input data to be labeled, the middle hidden layer learns a compact representation of the input data in an unsupervised manner [43]. However, researchers have found that the dimensionality reduction by the AANN is quite similar with the well-known Principal Components Analysis (PCA) technique [44].

More recently, a more mature and powerful AANN, the deep autoencoder networks [68] have presented an effective way of initializing the network parameters that leads the low-dimensional coding much better than PCA. For all the layers in the deep networks, the authors proposed a restricted Boltzmann machine to pretrain the network parameters layer-by-layer, followed by a fine-tuning procedure for optimal reconstruction via the Backpropagation algorithm [142].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Different from the unsupervised dimensionality reduction by the above AANNs, we intend to employ the neural networks to perform dimensionality reduction in a supervised manner using siamese architectures. Siamese Neural Networks have first been presented by Bromley *et al.* [24] using Time Delay Neural Networks (TDNN) on the problem of signature verification. This idea was then adopted by Chopra *et al.* [32] who used Siamese Convolutional Neural Networks (CNN) for face verification, i.e. to decide if two given face images belong to the same person or not. Recently, Berlemont *et al.* [18] also successfully employed the Siamese Neural Networks for inertial gesture recognition and rejection.

Remainder of this chapter is organized as follows: Section 5.2 incorporates MLP with the triangular loss function for classification and dimensionality reduction on small-scale datasets. Section 5.3 illustrates how to use our CNN-based TSML model and a hybrid training algorithm to realize end-to-end data visualization on large-scale data. Finally, we draw our conclusions in Section 5.4.

5.2 Classification and Visualization on Small-scale Data

Since the 1980's [142], Multi-layer Perceptrons (MLP) have been a popular solution to object classification problems such as image recognition [181] and speech recognition [106, 21]. It has been demonstrated that MLPs are able to approximate arbitrary nonlinear mappings and thus can easily fit to a set of training data [100]. However, good generalization to a set of test data is not guaranteed so that MLPs are usually sensitive to over-fitting problems in practice.

A classical MLP consists of an input layer, one or more hidden layer(s) and an output layer of perceptrons. There is no theoretical rule but only empirical tricks for configuring the structure of an MLP, i.e. the values of hyperparameters, including the number of hidden layers, the size of each hidden layer and the type of activation functions. These hyperparameters determine the power of the MLP and should be carefully selected for a practical problem. Thus developing the right MLP is indeed an art of balance between under-fitting and over-fitting. Generally, in a multi-class classification problem, apart from the hyperparameters, the size of the input layer is determined by the dimension of input feature vectors; and the size of the output layer is fixed to the dimension of predefined target vectors. Target values are typically binary for classification problems. For example, for a 3-class classification problem, we usually set unit vectors $[1,0,0]^T, [0,1,0]^T, [0,0,1]^T$ as target vectors for the 3 classes. In other words, the output dimension of an MLP is usually constrained to the number of classes in the problem.

In this work, we present TSML-MLP as a semi-supervised learning method for classification. It performs learning on similar and dissimilar data pairs while the classical MLP

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

trains on fully labeled training samples. TSML-MLP incorporates the proposed triangular loss function with two MLPs in a siamese architecture, to relax the constraint on the output dimension, making flexible dimensionality reduction to the input data. The two MLPs in TSML-MLP actually share the same set of parameters. Compared with the classical MLP that constrains the outputs approaching some predefined target values, TSML-MLP defines a specific objective: (1) for an input pair from the same class, making the pairwise similarity between their outputs larger; (2) for an input pair from different classes, making the pairwise similarity between their outputs smaller. With such an objective, the dimension of the target space can be arbitrarily specified.

More interestingly, TSML-MLP has this advantages *without* losing its superior ability of accurate classification. In our experiments, we compare the TSML-MLP with the classical MLP for face identification on the Extended Yale B dataset [51]. In addition, we employ a statistical significance testing method called Bootstrap Resampling [84] to evaluate the comparison between TSML-MLP and the classical MLP. The testing results show that TSML-MLP achieves comparable performance with the classical MLP on the problem of face identification.

Overall, the main contributions of this work are summarized as below:

- we show the capability of TSML-MLP for dimensionality reduction and data visualization in 2-d and 3-d spaces. We find that TSML-MLP projects the original input data to the vertexes of a regular polyhedron.
- we demonstrate that TSML-MLP has the above advantages *without* losing its superior ability of accurate classification. It achieves comparable performance with the standard MLP on face classification.

5.2.1 Multi-layer Perceptrons

We use the same MLP either as an independent classifier or as the mapping function in the siamese architecture of TSML-MLP. Figure 5.1 shows the structure of the MLP to realize the mapping from the input to the output, size of the only hidden layer is set to 100. Size of the input layer is 262 to receive the whitened feature vectors. Adjacent layers are fully connected and the activation function is the tanh function. Size of the output layer is larger than 2, depending on the model and its application.

The MLP classifier

For a given input sample $\mathbf{x}_i$, assuming its output on the MLP is $\mathbf{a}_i$. Formally, the output is a function of the input and the weights $\mathbf{W}$ of the MLP: $\mathbf{a}_i = f(\mathbf{x}_i, \mathbf{W})$. The objective function

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

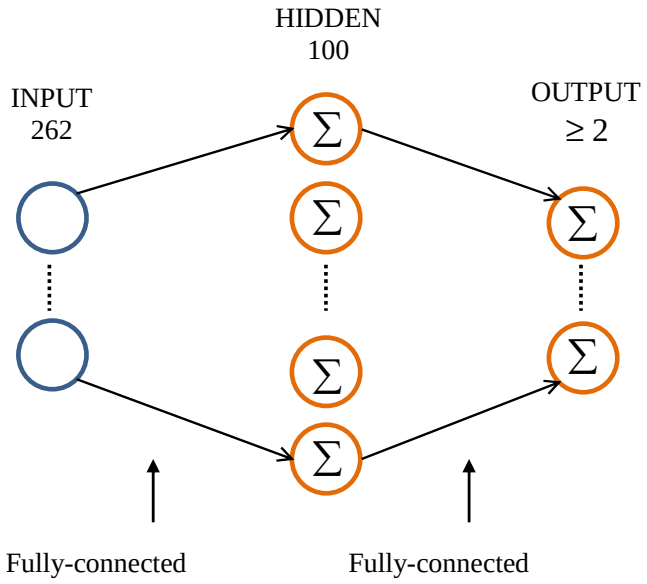


Fig. 5.1 Diagram of the 3-layer MLP used for face classification.

of an MLP classifier is simply the Mean Squared Error (MSE) between the computed outputs and their desired targets for all training samples:

$$J = \frac{1}{2N}\sum_{i=1}^{N}(\mathbf{a}_i - \mathbf{g}_i)^2, \tag{5.1}$$

where N is the number of all possible training samples, $\mathbf{g}_i$ is the target vector for the output sample $\mathbf{a}_i$. Remind that $\mathbf{g}_i$ is usually hand-crafted unit vectors. For example, for a 3-class classification problem, we usually set the unit vector $[1,0,0]^T$ as target vector for the first class. Thus, index of the largest element in an output $\mathbf{a}_i$ indicates the class that it belongs to.

TSML-MLP

As we have mentioned in previous chapters, the proposed TSML methods takes a pair of samples as the input and requires no hand-crafted targets. For the i_{th} pair $(\mathbf{x}_i, \mathbf{y}_i)$ from a training set, the mapping function $f(\cdot)$ produces two outputs $(\mathbf{a}_i, \mathbf{b}_i)$ where $\mathbf{a}_i = f(\mathbf{x}_i, \mathbf{W})$ and $\mathbf{b}_i = f(\mathbf{y}_i, \mathbf{W})$. The objective of TSML-MLP is to minimize the triangular loss function:

$$J_i = \frac{1}{2}\|\mathbf{a}_i\|^2 + \frac{1}{2}\|\mathbf{b}_i\|^2 - r\|\mathbf{c}_i\| + r^2, \tag{5.2}$$

where r is a constant constraint on the vector length; $\mathbf{c}_i = \mathbf{a}_i + s_i\mathbf{b}_i$, representing the simplified Triangular Similarity and $s_i = 1$ (respectively $s_i = -1$) means that the two vectors $\mathbf{a}_i$ and $\mathbf{b}_i$ are a within-class pair (respectively a between-class pair). Minimizing such a function makes

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

a within-class pair closer and separates a between-class pair as much as possible. After the mapping, a kNN classifier is used to perform classification on all the outputs.

Difference between MLP and TSML-MLP

From Chapter 3, we have known that the classical MLP and TSML-MLP have similar gradient formulations, so that we can employ the same Backpropagation algorithm [142] for training them. However, there are also apparent differences between them on both the input and output layers.

For each training vector $\mathbf{x}_i$, the classical MLP needs to know which class $\mathbf{x}_i$ belongs to. In contrast, TSML-MLP takes a more flexible constraint: it only needs the side information – whether two input vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ are of the same class or not. The relationship between the two constraints can be summarized as:

- when we know the class labels of $\mathbf{x}_i$ and $\mathbf{y}_i$, we know whether they are of the same class or not;
- however, even we know whether $\mathbf{x}_i$ and $\mathbf{y}_i$ are of the same class or not, we may have no idea about the class labels of $\mathbf{x}_i$ and $\mathbf{y}_i$.

As a result, TSML-MLP is applicable with the second constraint while the classical MLP is not, i.e. TSML-MLP can learn on side information only. More important, we will demonstrate that the relaxation of constraint would not cause classification accuracy loss to experiments.

On the output layer, the classical MLP fixes the output dimension equal to the number of classes. However, TSML-MLP has no constraint on the output dimension. Therefore, for a problem with more than 3 classes, TSML-MLP is applicable for data visualization, i.e. projecting the input data into 2-d or 3-d spaces; but the classical MLP can only make a projection into a space with dimension more than 3. In following sections, we will illustrate the effect of TSML-MLP on dimensionality reduction and data visualization.

Batch Gradient Descent or Stochastic Gradient Descent

Once we have defined an error function and its gradient, the Backpropagation algorithm [142] applies gradient descent to minimize the overall error for all training data iteratively. Typical gradient descent algorithms include stochastic gradient descent, batch gradient descent, and the trade-off between them, mini-batch gradient descent.

Particularly, stochastic gradient descent uses only one data sample for training in each iteration and mini-batch gradient descent takes several data samples as a small batch in each

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

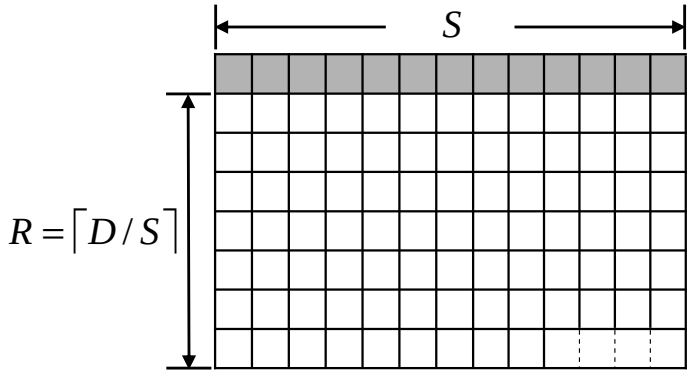


Fig. 5.2 Index matrix for mini-batch gradient descent of TSML-MLP. The first row stores the S within-class pairs, followed by all the between-class pairs. The empty positions in the end of the matrix can be optionally filled with some between-class pairs.

training iteration. These two algorithms are quite similar and useful for online learning applications but the latter is usually preferred because it yields smoother gradient change. Batch gradient descent uses all training samples in each iteration so it is only available for offline training. Some advanced optimization algorithms can help batch gradient descent to accelerate the learning speed, namely, the Conjugate Gradient Descent (CGD) algorithm [115] and the Limited-memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm [108]. Compared with stochastic and mini-batch gradient descent, the advantage of these advanced algorithms is that they have no need to manually pick a learning rate.

For the proposed TSML-MLP, the advanced batch gradient descent algorithms maybe only suitable for small-scale problems, because TSML-MLP takes *data pairs* for learning, and the number of all training sample pairs is exponentially larger than the number of all training samples. Specifically, for a problem of N training samples, the number of all possible sample pairs is $N(N-1)/2$. Therefore, for medium and large-scale problems, it is difficult to load all the training pairs as a whole batch, we have to use stochastic gradient descent or mini-batch gradient descent.

Commonly, a probable mini-batch contains equivalent number of within-class pairs and between-class pairs [32, 184]. However, the actual ratio of within-class pairs and between-class pairs is not equivalent. For example, for m classes each with n training samples, the number of within-class pairs is $mn(n-1)/2$ and the number of all between-class pairs is $mn(mn-n)/2$. Thus the ratio between within-class pairs and between-class pairs is $\frac{n-1}{n(m-1)}$, i.e. one within-class pair is accompanied by $\frac{n(m-1)}{n-1}$ between-class pairs. Consequently, instead of taking equivalent number of within-class pairs and between-class pairs in a mini-batch, we propose the following strategy to choose data pairs for a mini-batch:

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Algorithm 2: Optimization of the TSML-MLP

input : Training set; Number of training data N;
output: Parameters P

% initialization
Random initialization to the set of parameters P;
% optimization by back propagation
if *N is large* **then**
    % this is a large-scale problem ($N > 1000$)
    Set learning rate $\mu = 10^{-4}$;
    Generate mini batches that each contains 1 similar pair and R dissimilar pairs (Figure 5.2);
    Employ mini-batch gradient descent to update P;
else
    % this is a small-scale problem
    Generate a whole batch which contains all similar and dissimilar pairs;
    Employ batch gradient descent (the advanced L-BFGS algorithm) to update P;
% output the final set of parameters
return P.

- Count the training samples and denote the number as N, hence there are totally $N(N-1)/2$ sample pairs.
- Count the within-class pairs and denote the number as S, then the number of between-class pairs is $D = N(N-1)/2 - S$.
- Let $R = \lceil D/S \rceil$, i.e., the smallest integer not less than D/S.
- Make an index matrix with $R+1$ rows and S columns (Figure 5.2), put the indexes of the S within-class pairs in the first row and put the indexes of all the between-class pairs in the following rows.
- (Optional) Randomly pick some between-class pairs to fill the remain empty position in the end of the matrix.
- Take the indexes in each column as a mini-batch, which contains a single within-class pair and R between-class pairs.

In general, we summarize the optimization procedure for the proposed TSML-MLP in Algorithm 2. For a large-scale problem, mini-batch gradient descent is used in optimization; for a small-scale problem, batch gradient descent is adopted. Particularly, the scale of a problem is small or large depends on the machine capacity we used. In our case, we usually

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



Fig. 5.3 Example images of an individual in the Extended Yale B dataset. These frontal-face images were captured under various lighting conditions.

consider a problem with more than $1,000$ training samples as a large-scale problem, since the number of all possible similar and dissimilar pairs is at least $499,500$.

5.2.2 The Extended Yale B Dataset and Face Descriptors

We perform experiments on the Extended Yale B dataset [51]. It contains 2,414 frontal-face images of 38 individuals. These images were captured under various lighting conditions. All the images have been cropped and normalized, with the same size 192×168. Figure 5.3 provides some example images of an individual in the dataset. We can see that the lighting directions in different images are significantly varied. For instance, it is difficult to recognize the face in the middle of Figure 5.3 since it hides in deeply dark.

We divide the whole dataset into three non-overlapping subsets: training, validation and testing. We learn a model on the training set, choose the best set of parameters that achieves the highest performance on the validation set, and report the classification performance on the testing set using the best parameters. Especially, we take a small-scale training set in

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

the experiments: for each individual, only one out of ten images are used for training, i.e. there are 263 face images in the training set. And the size ratio of the training, validation and testing sets is 1:3:6. All the experiments are repeated 10 times with randomly shuffled data, and the mean accuracy ($\pm$ standard error of the mean) are reported.

Face Descriptors

Popular face descriptors for face detection and face recognition include eigenfaces [162], Gabor wavelets [38], haar-like features [105], SIFT [81], Local Binary Pattern(LBP) [1], etc. Recently, Barkan *et al.* [8] proposed Over-complete Local Binary Patterns (OCLBP), a new variant of LBP that significantly improved the face verification performance. Thus we adopt the OCLBP feature as the major face descriptor in our experiments. Besides, we also use Gabor wavelets and the standard LBP to represent the face images as a comparison. Following [8, 184], both the original face descriptors and their square roots are evaluated in the experiments.

Gabor wavelets: we extract Gabor wavelets with 5 scales and 8 orientations on each downsampled image. The downsampling rate is 10×10 for all the 192×168 images, thus the dimension of an extracted Gabor vector is 12,160 $(=5\times8\times19\times16)$.

Local Binary Patterns: we use the uniform LBP [128] to represent face images. The uniform LBP is denoted as $LBP^{u2}_{p,r}$, where $u2$ stands for 'uniform', (p,r) means to sample p points over a circle with a radius r. The dimension of an uniform pattern is 59. Concretely, each 192×168 image is divided into non-overlapping 16×16 blocks and uniform LBP patterns $LBP^{u2}_{8,1}$ are extracted from all the blocks. We concatenate all the LBP patterns into a feature vector, whose dimension is $7,788$ $(=12\times11\times59)$.

Over-complete Local Binary Patterns: unlike LBP adopts non-overlapping blocks, OCLBP adopts overlapping to adjacent blocks. Formally, the configuration of OCLBP is denoted as $S:(a,b,v,h,p,r)$. An image is divided into $a\times b$ blocks with vertical overlap of v and horizontal overlap of h, and then uniform pattern $LBP^{u2}_{p,r}$ are extracted from all the blocks. Moreover, the OCLBP is composed of several different configurations: $S_1:(16,16,\frac{1}{2},\frac{1}{2},8,1)$, $S_2:(24,24,\frac{1}{2},\frac{1}{2},8,2)$, $S_3:(32,32,\frac{1}{2},\frac{1}{2},8,3)$. The three configurations consider three block sizes: $16\times16, 24\times24, 32\times32$, and adopt half overlap rates along the vertical and horizontal directions.

We shift the block window to produce overlaps. Taking the 16×16 block window for example, with the shifting step $16\times\frac{1}{2}=8$ to the left and downwards, the total number of 16×16 blocks is $(\frac{192}{8}-1)\times(\frac{168}{8}-1)=460$. Similarly, shifting the 24×24 window produces 195 blocks and shifting the 32×32 window produces 110 blocks. The dimension of our OCLBP vectors is $45,135$ $((460+195+110)\times59)$.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

The input layer of an MLP equals to the dimension of input feature vectors, so high dimensional feature vectors make the MLP having an unnecessary large size of weights. To avoid this problem, we apply whitened PCA to reduce the vector dimension. Since the size of the training set is small (only 263 samples), we keep all the variance during dimensionality reduction. Thus the reduced dimension is 262, and these 262-d feature vectors are taken as inputs to the classical MLP or TSML-MLP.

5.2.3 Dimensionality Reduction in Face Classification

We evaluate three different methods in our experiments: *k*NN, MLP and the proposed TSML-MLP. Different from the classical MLP, it is hard for TSML-MLP to directly make class predictions on its output. We apply *k*NN on its output to perform class identification. This is also the reason why we evaluate the *k*NN method as a comparison. Specifically, *k*NN in our experiments uses the Cosine Similarity function to measure pairwise distance between data and the number of nearest neighbors k is set to 1.

Output dimension of TSML-MLP

Empirically, size of the hidden layer is set to 100 for both the classical MLP and TSML-MLP. As the number of different classes in the Extended Yale B dataset is 38, the output dimension of the classical MLP is fixed to 38. In contrast, TSML-MLP allows flexible output dimension, thus we shift the output dimension from 2 to 250 and record the influence on the identification accuracy. Note that the input dimension is 262, so we keep the output dimension less than 262 in order to perform dimensionality reduction. Figure 5.4 shows the identification accuracy curve of TSML-MLP method on the square-rooted OCLBP feature. We can see that the curve rises rapidly when the output dimension increases from 2 to 10, but then climbs up much slower. The optimal solution is with an output dimension more than 80.

Comparison with the classical MLP

Table 5.1 summarizes the results of different methods on different face descriptors on the Extended Yale B dataset. The output dimension of TSML-MLP is set to 80. Compared with *k*NN, TSML-MLP has brought significant improvement on face identification. Compared with the classical MLP, TSML-MLP achieves comparable results. For example, on the square-rooted LBP features, TSML-MLP obtains an accuracy of 96.34%, seems slightly better than the result of the classical MLP, 96.28%. Besides, methods using square-rooted features always obtain better performance than those using the original features. This phenomenon is consistent with that on the problem of face verification [184].

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

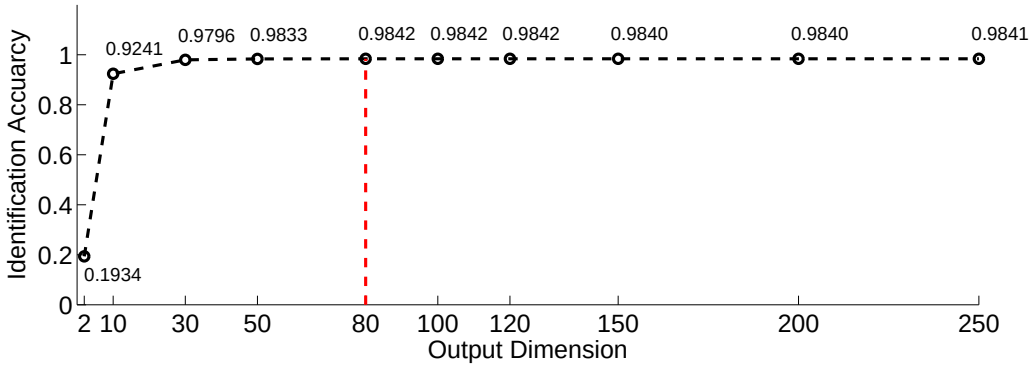


Fig. 5.4 Identification accuracy curve of TSML-MLP on the square-rooted OCLBP feature, with respect to the increasing output dimension.



Fig. 5.5 Face images that TSML-MLP using square-rooted OCLBP failed to recognize.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 5.1 Face identification accuracy (%) on the Extended Yale B dataset. Generally, TSML-MLP = MLP > kNN. The output dimension of TSML-MLP is set to 80.

Method		kNN	MLP	TSML-MLP
Gabor	original	69.37(±4.32)	**79.72(±3.49)**	79.70(±3.44)
	square-rooted	80.32(±0.43)	92.48(±0.27)	**92.62(±0.28)**
LBP	original	79.06(±0.42)	92.15(±0.41)	**92.27(±0.39)**
	square-rooted	84.78(±0.51)	96.28(±0.30)	**96.34(±0.31)**
OCLBP	original	82.50(±0.54)	96.41(±0.28)	**96.59(±0.31)**
	square-rooted	86.11(±0.55)	98.33(±0.17)	**98.42(±0.16)**

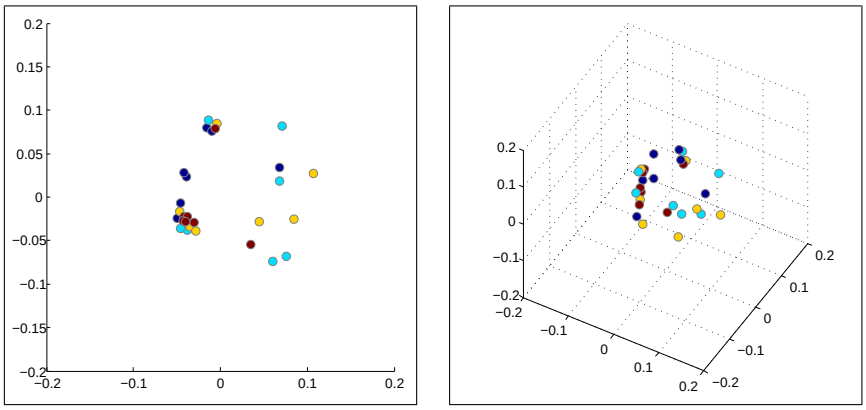
Table 5.2 Significance testing between MLP and TSML-MLP. A p-value smaller than 0.05 or 0.01 indicates a significant difference. Results confirm no significant difference between MLP and TSML-MLP.

Method		MLP	TSML-MLP	p-value
Gabor	original	**79.72(±3.49)**	79.70(±3.44)	0.4982
	square-rooted	92.48(±0.27)	**92.62(±0.28)**	0.3559
LBP	original	92.15(±0.41)	**92.27(±0.39)**	0.4150
	square-rooted	96.28(±0.30)	**96.34(±0.31)**	0.4486
OCLBP	original	96.41(±0.28)	**96.59(±0.31)**	0.3364
	square-rooted	98.33(±0.17)	**98.42(±0.16)**	0.3341

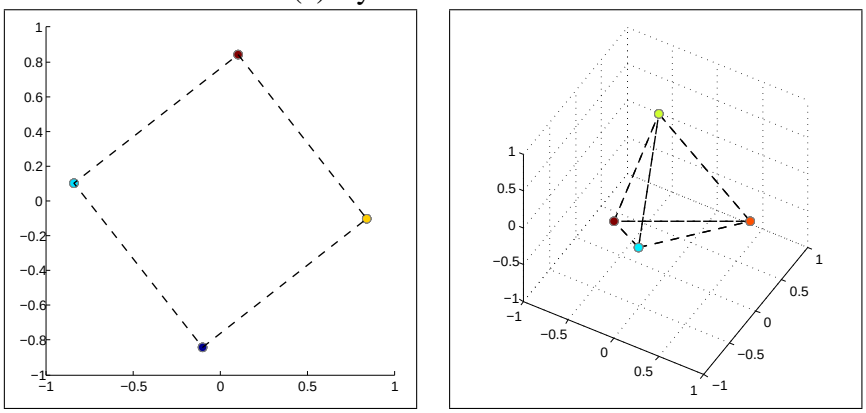
To confirm the comparison, we employ the Bootstrap Resampling approach [84] to evaluate the pairwise statistical significance between the two methods. Note that the smaller the p-value, the larger the significance. Usually, we consider a p-value smaller than 0.05 or 0.01 to indicate a significant difference. The significance testing results in Table 5.2 are all in the range [0.3, 0.5], showing that there is no significant performance difference between the classical MLP and TSML-MLP. We also test the significance between TSML-MLP and kNN, the p-value is always 0 on all the difference features, demonstrating that TSML-MLP has significantly improve the performance over the kNN method.

Comparing the three different face descriptors, the results on OCLBP are significantly better than those on Gabor wavelets and those on LBP. For example, TSML-MLP using square-rooted OCBLP achieves an average accuracy of 98.42% on the 10 repeated experiments. Figure 5.5 shows the face images that TSML-MLP failed to recognize. Most of the failure examples are rather dark so that it is difficult to extract effective facial texture features from them. However, there are also some failure examples in good lighting condition. This is probably because we apply kNN as the classifier and the final decision relies on the test sample's nearest neighbor in the training set. Since the training data are randomly selected, a good nearest neighbor for each test sample is not guaranteed.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) By Whitened PCA



(b) By TSML-MLP

Fig. 5.6 Visualization of dimensionality reduction into the 2-d or 3-d spaces using (a) Whitened PCA and (b) TSML-MLP.

Running time: however, in the completely supervised manner, MLP trains on all data samples but TSML-MLP takes all possible data pairs in training, so TSML-MLP runs much slower than MLP. For example, on the square-rooted OCLBP face descriptors, calculating the gradient once for all the data samples by the classical MLP costs only 0.0040 seconds on average, but calculating the gradient once for all the data pairs by TSML-MLP needs about 0.7340 seconds. The increase of time consumption exists for all metric learning methods in classification problems. Fortunately, this is affordable on the small-scale dataset used in the current work. The running time was recorded on a machine with a 4-core CPU, 8 GB RAM and 64-bit operating system.

5.2.4 Dimensionality Reduction in Data Visualization

In this subsection, we apply TSML-MLP to illustrate data visualization on a few data from the Extended Yale B dataset. We select the first 4 classes each with 7 face images, totally

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

28 face images. These images are represented by 262-d OCLBP feature vectors. For data visualization, all the input vectors are projected into the 2-d and 3-d spaces, respectively. In addition, we also visualize the projection of whitened PCA as a comparison in Figure 5.6.

Figure 5.6 (a) shows the data distribution in the 2-d and 3-d target spaces using whitened PCA, points with different colors are from 4 different classes. We can see that points of different classes are mixing in both the 2-d and 3-d spaces. In contrast, TSML-MLP successfully separates the points of different classes (Figure 5.6 (b)). More interestingly, points of the same class concentrate tightly at a certain position, standing as a vertex of a square in the 2-d space or a regular tetrahedron in the 3-d space. Note that both the square and the regular tetrahedron take the origin point as the center. Thus all the between-class pairs share exactly the same angle: (1) in the 2-d space, the angle between two points from different classes is 90°; (2) in the 3-d space, the between-class angle is about 109.47°. In summary, the objective of our TSML-MLP has been satisfied perfectly: separating the between-class pairs and concentrating the within-class pairs.

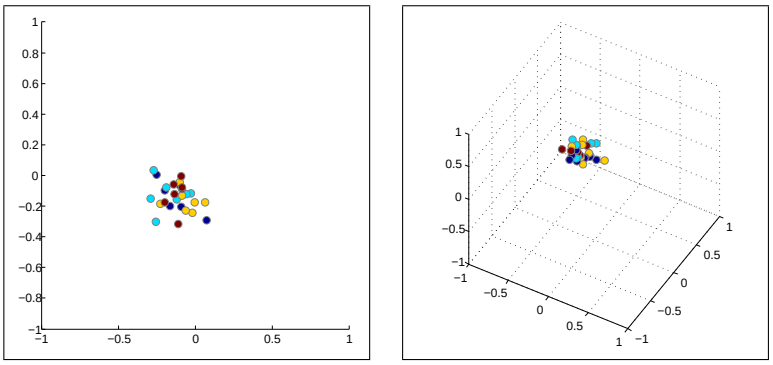
Figure 5.7 pictures a more detailed procedure of data projection by TSML-MLP using the mini-batch gradient descent algorithm (Section 5.2.1). At the beginning, TSML-MLP is initialized with random parameters, so we observe mixed data classes around the origin point in Figure 5.7 (a). Towards the objective of closing the within-class pairs and separating between-class pairs, the points scatter away after 1000 iterations. Successively, after 3000 iterations, data from different classes have found their own optimal positions, and we can see clear blank boundaries between different classes. Finally, after 20000 iterations, data of the same class concentrate at each optimal position in Figure 5.7 (d).

However, this data visualization is performed on a quite small amount of data and can not be directly applied on a large-scale dataset. In the following section, we will present a more powerful model than the MLP, i.e. the deep CNN model, to realize nonlinear mappings. We will also propose a hybrid training algorithm to train this model efficiently.

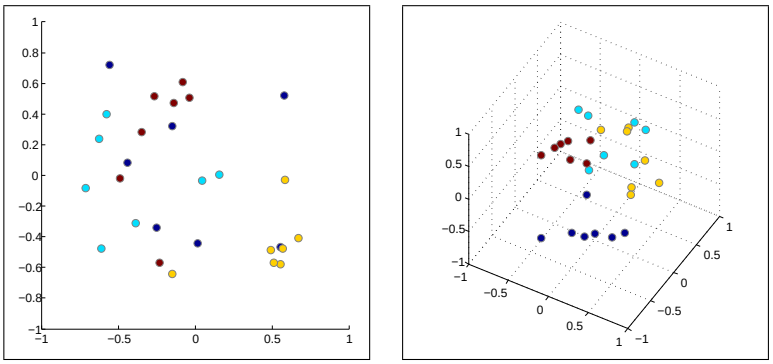
5.3 End-to-end Data Visualization on Large-scale Data

In all the previous works, hand-crafted features such as LBP, OCLBP are used to represent a face image. In contrast, end-to-end learning has received much attention recently because it requires no hand-crafted feature extraction algorithms but can be trained to predict outputs from low-level inputs without extracting features. For example, for speech recognition, Recurrent Neural Networks (RNN) learn to map directly from acoustic signals to phonetic sequences [57]; for image classification, Convolutional Neural Networks (CNN) are able to directly classify raw pixels into high-level concepts such as object categories [86].

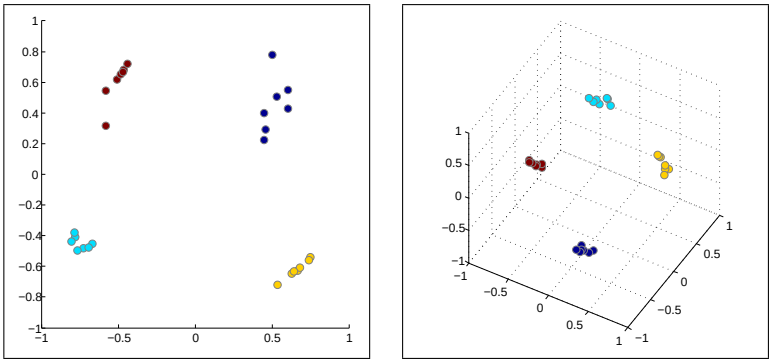
Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



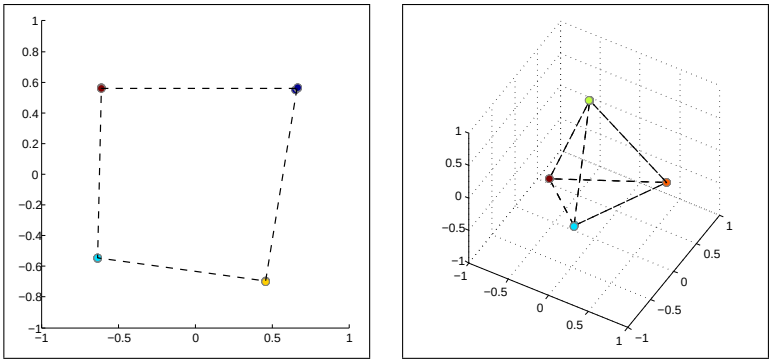
(a) Random Initialization



(b) After 1000 iterations



(c) After 3000 iterations



(d) After 20000 iterations

Fig. 5.7 Illustration of dimensionality reduction into the 2-d or 3-d spaces using TSML-MLP.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

CNN is the most popular choice for end-to-end learning on images since it is originally designed to receive an image as input. Like an MLP classifier, a classical CNN classifier is only available for constrained dimensionality reduction because the size of the output layer (i.e. the output dimension) is fixed to the number of classes in a classification problem. In this work, we integrates the proposed triangular loss function with two CNNs in a siamese architecture, to relax the constraint on the output dimension, making flexible dimensionality reduction to the input data. We name this method as TSML-CNN. Specially, we directly map raw handwritten digit images in the MNIST dataset into visualizable spaces, i.e. dimension of the target space is lower than 3 so that one can see the objects in it. This particular kind of dimensionality reduction is so called *end-to-end data visualization*.

Like the relationship between MLP and TSML-MLP, in a completely supervised manner, a CNN trains on all data samples and a TSML-CNN takes all possible data pairs in training, so a TSML-CNN runs much slower than a CNN classifier. Time consumption of training TSML-CNN on a large-scale dataset may be even unbearable. Therefore, we propose hybrid training to accelerate learning: (1) training a TSML-CNN on a small portion of training data to initialize the positions of different classes in the target space; (2) taking the initialized positions as class labels for all the data samples, and training a classical CNN on these re-labeled data. On the final outputs, we can apply kNN to perform class identification. We carry out experiments on the MNIST handwritten digits dataset [96], projecting all the original 28×28 images into the 1-dimensional, 2-dimensional or 3-dimensional spaces.

Overall, the main contributions of this work are the following:

- we present TSML-CNN to perform flexible dimensionality reduction and thus to realize end-to-end data visualization for a large-scale dataset. Moreover, we will introduce a hybrid training strategy to accelerate model training.
- we show that the proposed method realizes end-to-end data visualization *without* losing its superior ability of accurate classification. It achieves competitive performance on class identification in low dimensional and visualizable spaces.
- we demonstrate that the Triangular Similarity based objective provides us two different perspectives to visualize the mapping results: one is on the unit sphere (hypersphere) in a specific low dimensional target space, the other is on the unfolded plane (hyperplane) in an even lower space.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

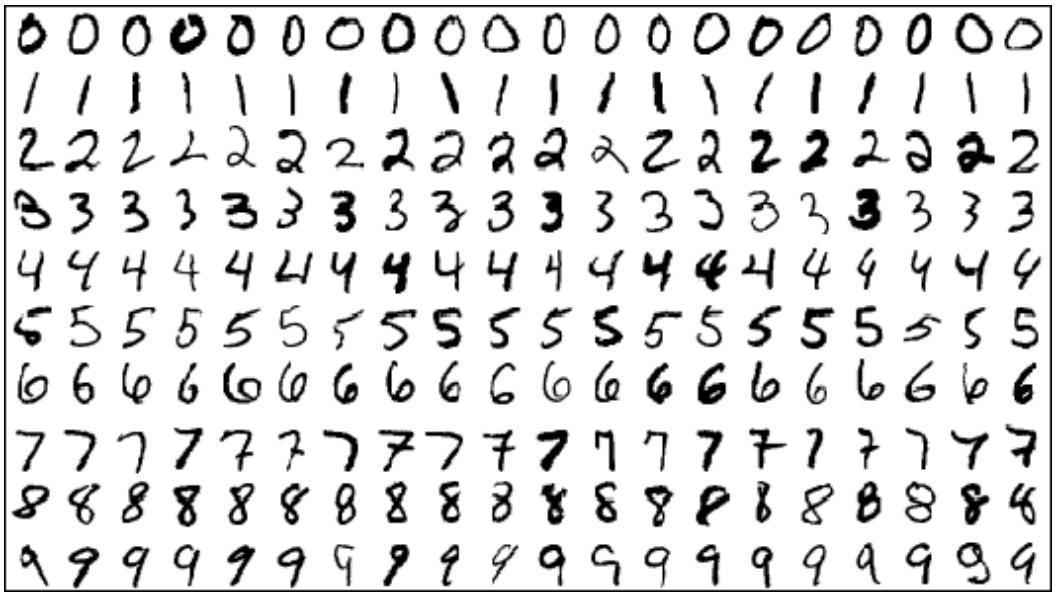


Fig. 5.8 Example images of handwritten digits in the MNIST dataset [96].

5.3.1 The MNIST Dataset and Convolutional Neural Networks

The MNIST handwritten digits dataset [96][2] is a popular benchmark for classification by neural network based algorithms. There are 70,000 8-bit grayscale images in total, where 60,000 images are separated as the training data and the remaining 10,000 images are used for testing. All the images are of the same size 28×28, capturing a digit from 0 to 9 with various writing styles. Figure 5.8 shows some example images of handwritten digits in the MNIST dataset.

The CNN architecture

CNNs are a specialized kind of neural networks for processing data that have a known *grid-like topology* [16], i.e. 2-dimensional data such as images and speech time-series [96, 127, 91, 98]. A CNN architecture comprises several kinds of neuron layers and has considerable capacity for data modeling. Designing a CNN is a complex task relying on rich empirical knowledge of configuring hyperparameters, i.e. determining the number of layers, the number of feature maps, the size of local receptive fields, the size of pools. In this work, we employ the CNN architecture recommended by [79][3] to process the images of the MNIST dataset. In particular, this CNN architecture is similar with that in LeNet-5 [96] but of different hyperparameters. Figure 5.9 illustrates the layers in this CNN architecture.

[2] http://yann.lecun.com/exdb/mnist/
[3] http://caffe.berkeleyvision.org/

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

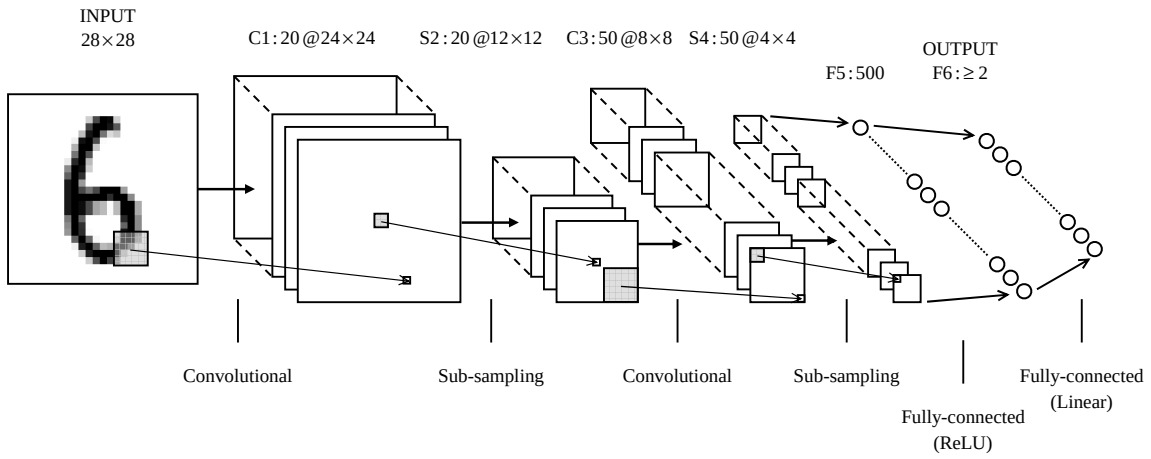


Fig. 5.9 Diagram of our proposed CNN architecture.

This CNN is composed of 6 layers, not counting the input layer. There are two convolutional layers at the 1st and 3rd layers, denoted by C1 and C3. Particularly, C1 and C3 are followed by two sub-sampling layers (i.e. pooling layers) S2 and S4.

For all the convolutional layers, the size of local receptive fields is always 5×5. And a feature map is learned on one or more pages of units in the previous layer. Taking a feature map learned on twenty previous pages for example, twenty 5×5 weight matrices would be used to perform convolution on each page in the previous layer, then the results are added to a trainable bias. Such a feature map contains 501 $(25 \times 20 + 1)$ parameters. If there is only one previous page, the number of parameters is simply 26 $(25 + 1)$.

For all the sub-sampling layers, the size of pools is always 2×2. Especially, we adopt a nonparametric sub-sampling operation, i.e. the max pooling: the four inputs in a pool are compared and the maximum is delivered to the following layer. And each feature map of sub-sampling has 4 connections between hidden units.

Layer C1 performs a convolution operation with 20 feature maps on the 28×28 input image, resulting in 20 pages of hidden units in the first hidden layer. C1 contains 520 (20×26) trainable parameters and $299,520$ $(24 \times 24 \times 520)$ connections.

On each of the 20 hidden pages, Layer S2 learns a feature map of sub-sampling. The 2×2 pool area is non-overlapping, thus we observe image down-sampling from each 24×24 page in C1 to a smaller page of size 12×12 in S2. S2 contains no parameters and $11,520$ $(12 \times 12 \times 20 \times 4)$ connections.

Layer C3 is a convolutional layer with 50 feature maps. Each feature map is learned on all the pages in the previous layer S2. Thus C3 has $25,050$ (50×501) trainable parameters and $1,603,200$ $(8 \times 8 \times 25,050)$ connections between hidden units.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Layer S4 is a sub-sampling layer with 50 feature maps. The sub-sampling operation from C3 to S4 is similar to that between C1 and S2. Thus S4 has no trainable parameters and $3,200$ ($4 \times 4 \times 50 \times 4$) connections.

The 5th layer F5 is a fully-connected layer having a ReLU activation function [54]. It contains 500 hidden units and thus $400,500$ ($500 \times (800+1)$) trainable parameters/connections, where 800 is the number of hidden units in S4.

The 6th layer F6 is also a fully-connected layer but performing a simple linear mapping into the target space. The size of this layer depends on the choice of the target space. For example, if we want to realize a mapping into the 3-dimensional space, we set the size of this layer to 3. We regard this layer as the output layer as we use the element values in this layer as the final vector representation for the input image.

TSML-CNN

Let a matrix $\mathbf{X}$ denote an input image and a vector $\mathbf{a}$ represent the output vector, the above CNN accomplishes a deep nonlinear mapping $f(\cdot)$ that $\mathbf{a} = f(\mathbf{X}, \mathbf{W})$ where $\mathbf{W}$ indicates all the parameters in this CNN. Involving the CNN in our proposed siamese architecture, TSML-CNN simply takes a pair of images $\mathbf{X}_i$ and $\mathbf{Y}_i$ from a training set and produces two outputs $\mathbf{a}_i = f(\mathbf{X}_i, \mathbf{W})$ and $\mathbf{b}_i = f(\mathbf{X}_i, \mathbf{W})$. Similar with TSML-MLP, TSML-CNN calculate the triangular loss on the two outputs by Equation (5.2).

We have known that training a siamese network converges much slower than training a classical single-track network of the same size (see Section 5.2). Especially, in the large training set of the MNIST dataset, the 60,000 training samples indicate 1.8 billion training pairs in total, thus time consumption of training TSML-CNN is unaffordable although the MNIST dataset is redundant.

TSML-Hybrid: Hybrid training

To maintain the advantage of flexible dimensionality reduction of TSML-CNN but train it more efficiently, we propose a hybrid training algorithm, namely TSML-Hybrid that includes the following four stages:

1. *Tiny-scale training*: select only a few training samples from each class and train TSML-CNN on corresponding similar and dissimilar training pairs. In practice, the similar pairs play an important role in controlling vector lengths, thus we usually select at least 2 training samples from each class in order to generate both similar and dissimilar pairs. With a few training data only, TSML-CNN gets convergence to an optimal solution very quickly, and outputs of different classes evenly scatter in the target space.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

2. *Transplanting*: set the centers of different classes in the target space as new labels for each class; take the learned parameters in TSML-CNN as initialization for a standard CNN. This operation can be considered as transplanting the CNN in TSML-CNN to a new single-track CNN. It is noteworthy that with the transplanting, the single-track CNN no longer requires hand-crafted target vectors and thus inherits the ability of flexible dimensional reduction from TSML-CNN.

3. *Large-scale training*: take the new labels as target vectors for each class and train the standard CNN on all training samples. The Mean Squared Error (MSE) function (Equation (5.1)) computes cost between output vectors and their desired targets. Compared with the number of possible data pairs, the number of data samples is much smaller thus efficient convergence can be reached.

4. *(Optional) Length normalization*: apply L2 normalization to make all the output vectors having unit length in the target space. The Cosine Similarity (as well as the Triangular Similarity) does not use the length information of vectors to distinguish two vectors, so performing length normalization removes length information of vectors and produces a more concise mapping result. This operation is especially interesting and useful for data visualization. Concretely, without the degree of freedom on length, the mapping outputs distribute on a sphere or hypersphere in the target space. According to the manifold learning theory that a hypersphere with a point removed is homeomorphic with a hyperplane [120], we can unfold the hypersphere into the lower dimensional space so that we have a new perspective to view the mapping result.

Compared to directly training a siamese network such as TSML-CNN, the proposed hybrid algorithm greatly improves the efficiency of training but does not suppress the classification performance. We will provide more results in the following experimental section.

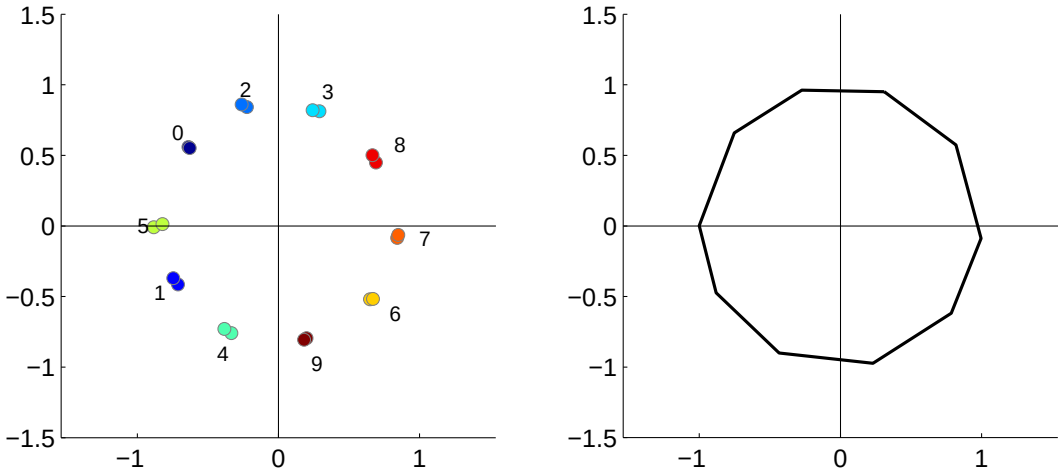
5.3.2 Dimensionality Reduction in Data Visualization

In this section, we use the above hybrid training algorithm, TSML-Hybrid, to realize data visualization of the MNIST data. We accomplish end-to-end data mapping into 2-dimensional, 3-dimensional and 4-dimensional spaces, respectively.

Case study 1: 2-dimension

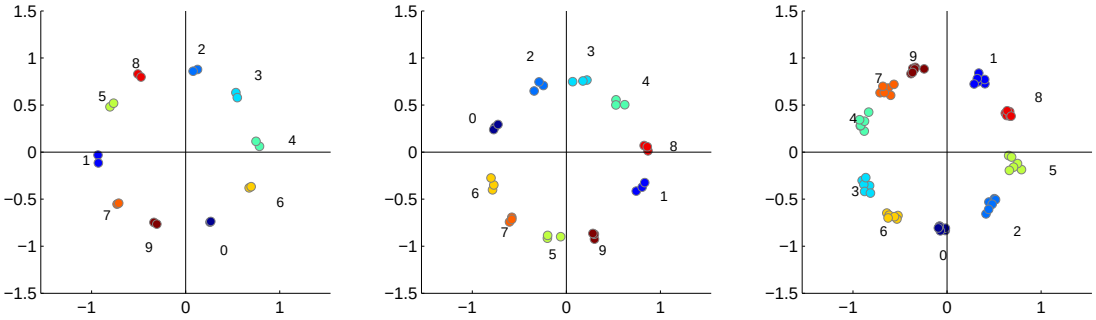
In *tiny-scale training*, we randomly select 2 training sample from each of the 10 classes in the MNIST dataset. We train TSML-CNN on these data pairs with the output dimension equal to 2. The learned model projects the 20 training samples into the target space as in

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) Data distribution (b) Polygon connecting the data

Fig. 5.10 After tiny-scale training, (a) the training points evenly scatter around the origin; (b) centers of different classes stand as vertexes of a polygon in the 2-dimensional space.



(a) 20 samples (b) 30 samples (c) 60 samples

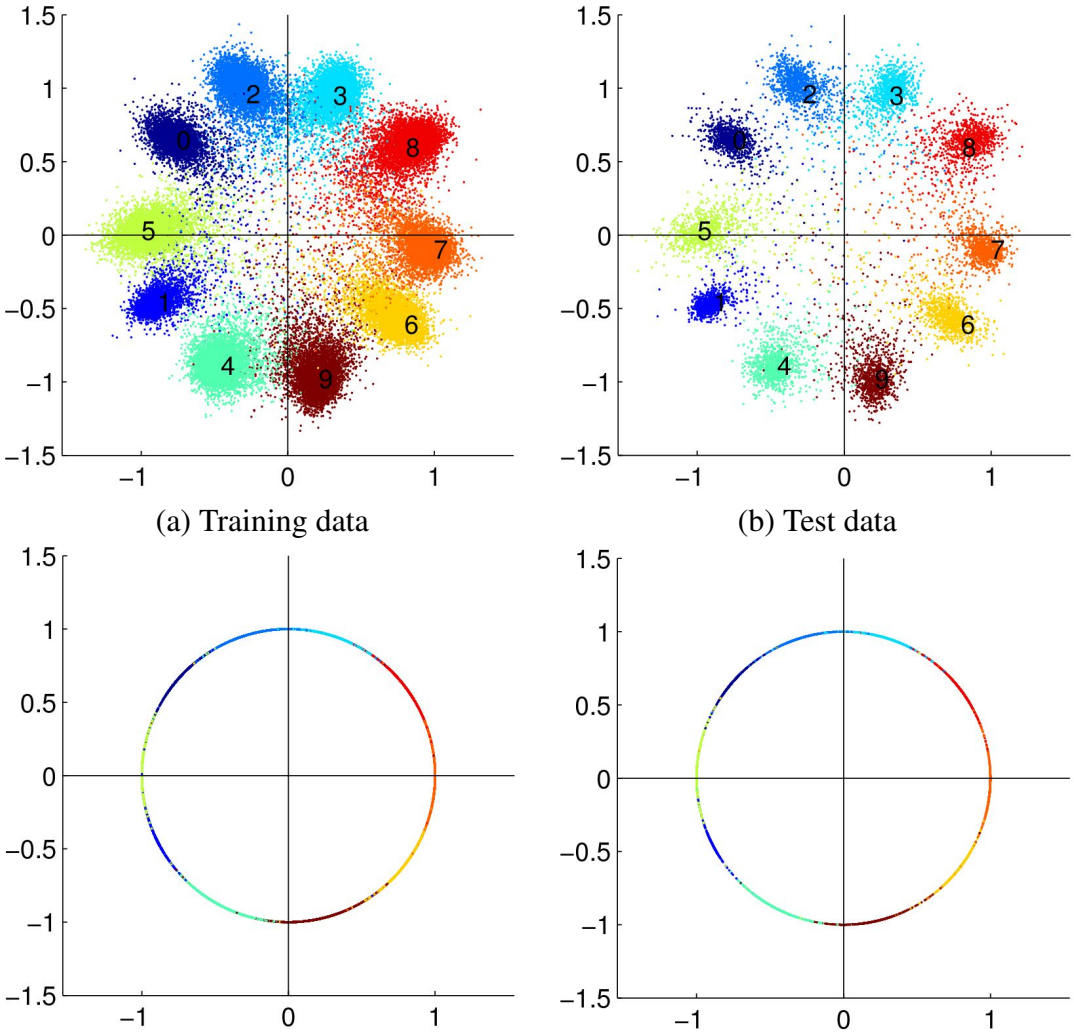
Fig. 5.11 Results after tiny-scale training with different data size and initialization.

Fig. 5.10 (a), where numbers beside the points denote the digits they represent. The positions of the digits shown in this figure depend on the random initialization of TSML-CNN. We observe no evident relation between adjacent digits in our experiments: Fig. 5.11 shows other mapping results with different data size and initialization, though the order of the 10 digits varies, the overall distribution is always approximating a circle.

After the tiny-scale training, an expected status of the training data has been reached, the 20 samples evenly distribute along a certain circle around the origin. Furthermore, connecting the centers of every class results in a 10-sided polygon (Fig. 5.10 (b)).

In *large-scale training*, the class centers in Fig. 5.10 are set as labels (i.e. target vectors) for their classes respectively. Minimizing the MSE cost function makes all the training

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) Training data (b) Test data

(c) Training data (d) Test data

Fig. 5.12 After large-scale training, either (a) the training data or (b) the test data assemble around different class centers. By length normalization, all the data are further projected onto the unit circle in (c) and (d).

samples close to these centers as much as possible. The final distribution after training is shown in Fig. 5.12 (a) and (b). We can see that data of each class scatter in a local small area in the 2-dimensional space, assembling around the labeled class center. Since we always use the Cosine Similarity to measure similarity between a pair of data, we can normalize the data vectors to have unit length without losing any discriminative information. After length normalization, all the data locate on the unit circle in the 2-dimensional space (see Fig. 5.12 (c) and (d)).

In manifold learning theory, a circle with a point removed is homeomorphic with a real line. In other words, as the length information is useless in distinguishing a pair of points on a circle, the circle can be represented by the degree of freedom on angle only. In Figure 5.13,

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

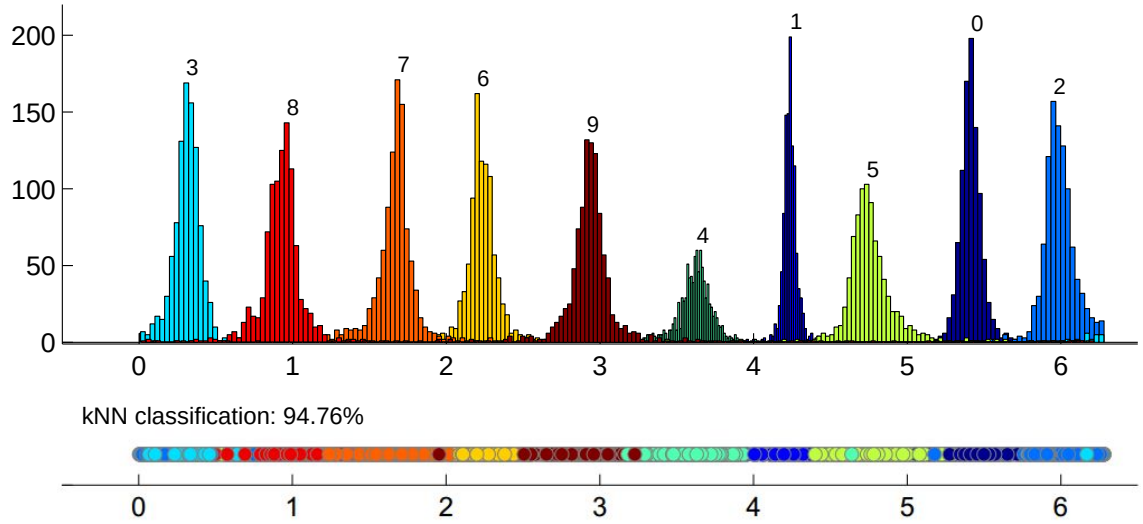


Fig. 5.13 The lower picture gives an unfolded view of the 2-dimensional MNIST test data from the 1-dimensional space; the upper picture provides the histogram of the data in 10 classes.

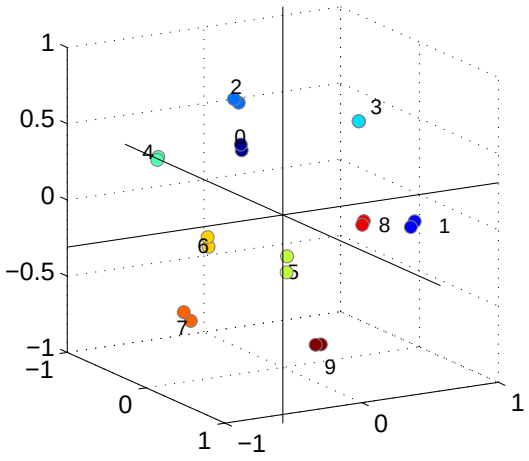
we unfold the circle to an interval $[0, 2\pi]$ in the lower picture and present the histogram of each class in the upper picture. We can see that each class has a sharp peak and adjacent classes are divided with clear valleys, which means that data in the same class have been well concentrated and data in different classes have been well separated. More surprisingly, *k*NN classification on the 1-dimensional data yields an accuracy of 94.76%. In conclusion, with only one mapping, we have obtained two views of data visualization on the test data, i.e. Fig. 5.12 (d) and Fig. 5.13.

Case study 2: 3-dimension

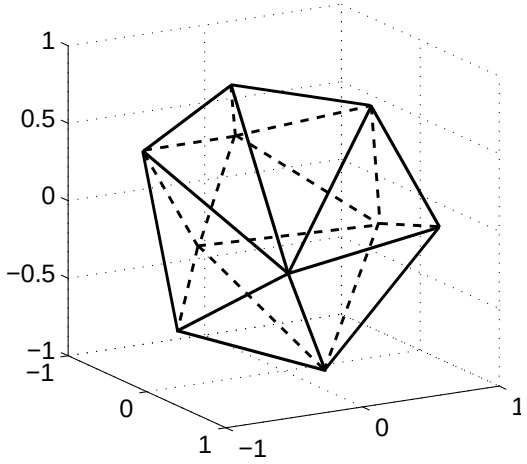
When we set the output dimension to 3, a different model is learned on the same 20 training samples. Firstly, the well-trained TSML-CNN projects the 20 training samples into the 3-dimensional space, comprising a convex polyhedron with 10 vertexes. Specifically, most planes of this polyhedron are triangular planes, planes with more than 3 sides may be possible but only in ideal situations. The mapping results and the convex polyhedron are shown in Fig. 5.14 (a) and (b).

Similar with the unit circle in the case of 2-dimension, the final CNN model produces a unit sphere carrying all the 3-dimensional data (see Fig. 5.14 (c) and (d)). More interestingly, a sphere with a point removed is homeomorphic with a plane. In other words, considering the sphere as the earth, the plane is like the world map. Figure 5.14 (e) illustrates the unfolded map for the test data. On this map, different classes of data locate as 10 isolated continents.

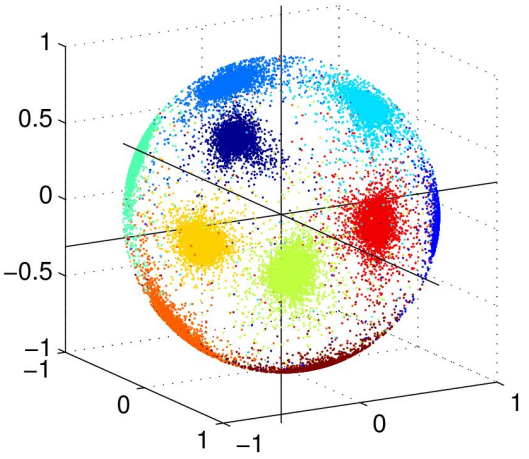
Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



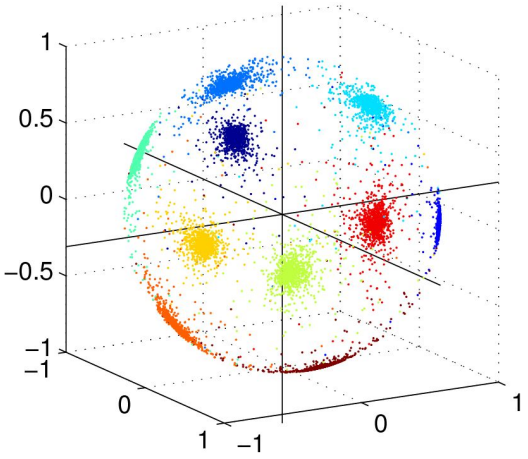
(a) Distribution after tiny-scale training



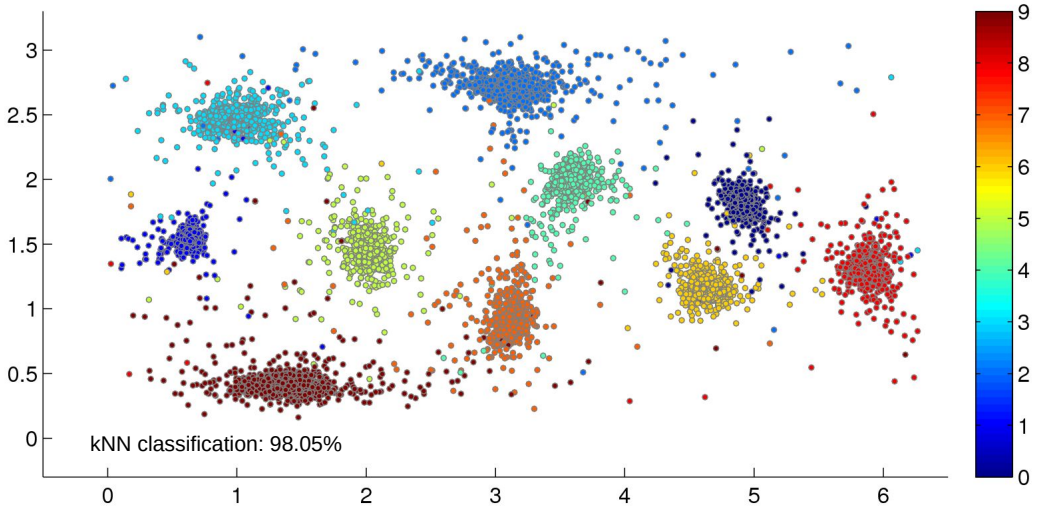
(b) Polyhedron connecting the tiny-scale data



(c) Training data after large-scale training



(d) Test data after large-scale training



(e) Unfolded view of the test data

Fig. 5.14 Mapping results of the 3-dimensional MNIST test data before and after unfolding.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

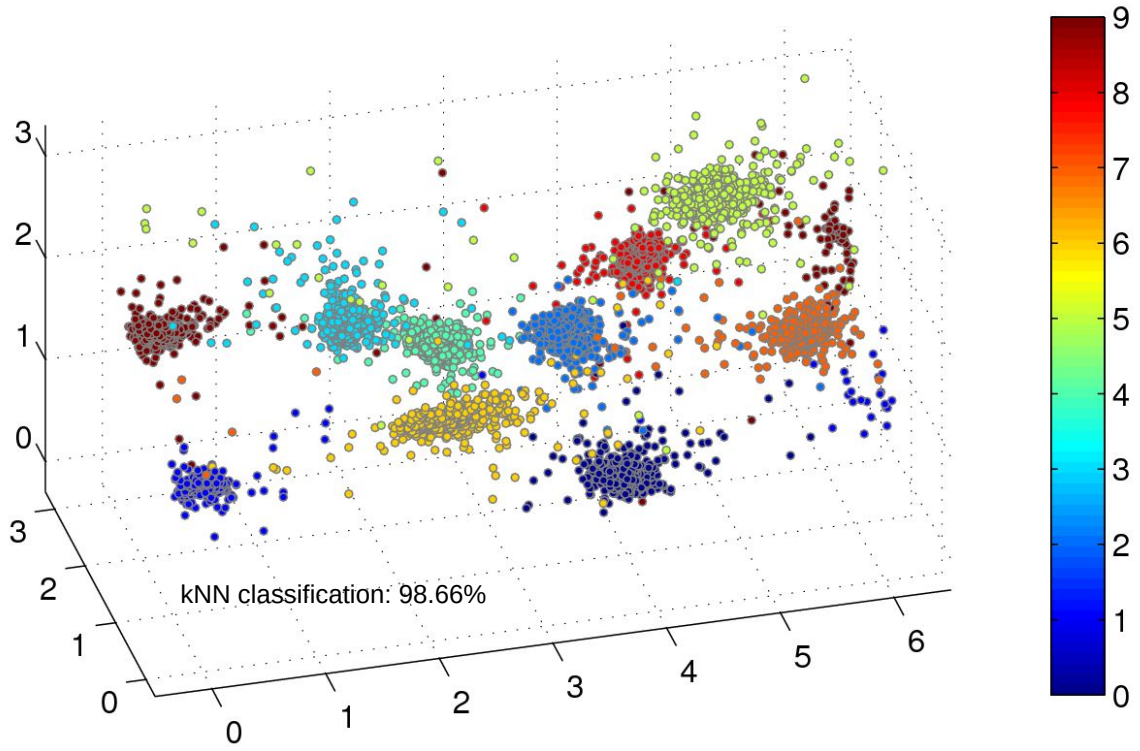


Fig. 5.15 An unfolded view of the 4-dimensional MNIST test data in the 3-dimensional space.

Note that the leftmost part of the map and the rightmost part are actually connected in the 3-dimensional space. The length and width of this map are 2π and π, respectively. In addition, kNN classification accuracy of the 2-dimensional data is 98.05%, much higher than that of the 1-dimensional data.

Case study 3: 4-dimension

In fact, we can not see the 4-dimensional space but only a subspace of it. Fortunately, the proposed method provides such a subspace of the target space, i.e. the unit hypersphere. Similarly, unfolding this 4-dimensional hypersphere results in a part of the 3-dimensional space (see Fig. 5.15). The length, width and height of this partial space are 2π, π and π, respectively. Like the other cases above, we observe clear boundaries between different classes. But one may notice that a few data of the two classes at the leftmost side are mapped to the rightmost side, although they are actually together in the 4-dimensional space. To make better data visualization, a probable solution can be duplicating the 3-dimensional mapping results to make the visualization locally complete. In terms of classification, since the kNN classifier makes prediction on the majority label of k-nearest neighbors in the training set,

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 5.3 Comparison on training time (in seconds) of different methods. '—' means no available result. View 1: before unfolding; View 2: after unfolding.

Methods	1-dim.	2-dim.	3-dim.	10-dim.	Mean
CNN classifier [79]	—	—	—	968.33	968.33
DrLIM [62]	—	9632.10	9579.83	9556.59	9589.51
TSML-Hybrid (view 1)	—	1111.64	1113.89	1119.19	1116.75
TSML-Hybrid (view 2)	1111.64	1113.89	1121.10	1117.92	

the above situation of data separation has little influence on the classification performance. The kNN classifier obtains an classification accuracy of 98.66% on the 3-dimensional data.

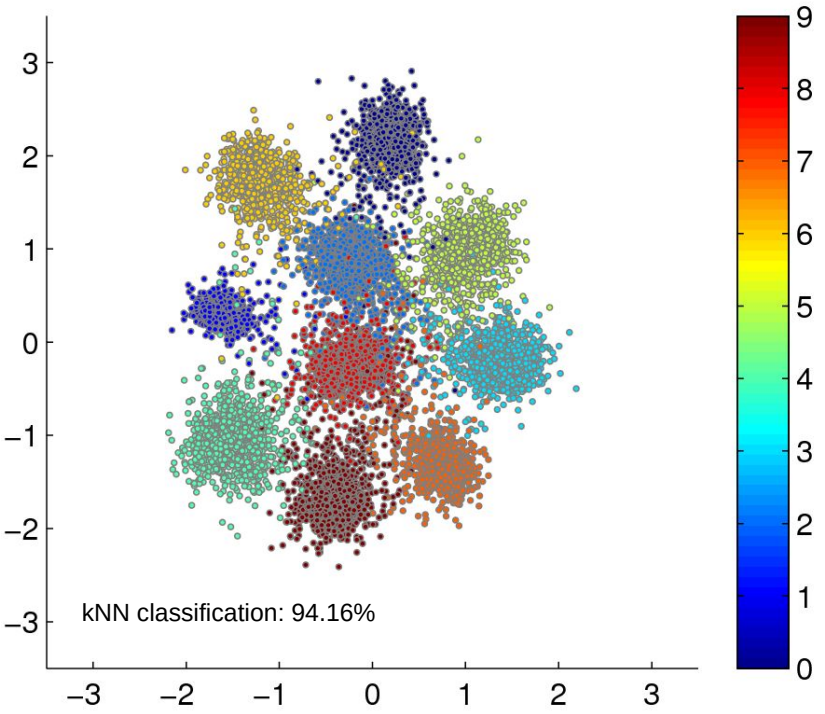
Comparison with the state-of-the-art

The most related work with ours is Dimensionality Reduction by Learning an Invariant Mapping (DrLIM) [62], which proposed a contrastive loss function to measure cost of data pairs. Different with our triangular loss function concerning the Cosine Similarity (or the Triangular Similarity), the contrastive loss function was based on the Euclidean distance. The objective of the contrastive loss is to minimize the distance between a similar pair and to separate any two dissimilar data with a distance margin. With the same CNN architecture used in TSML-Hybrid, DrLIM projects the MNIST test data into the 2-dimensional and 3-dimensional spaces as in Fig. 5.16. However, while DrLIM provides only one perspective to see the mapping results, TSML-Hybrid takes advantage of the triangular loss and offers two views of visualizing the output data.

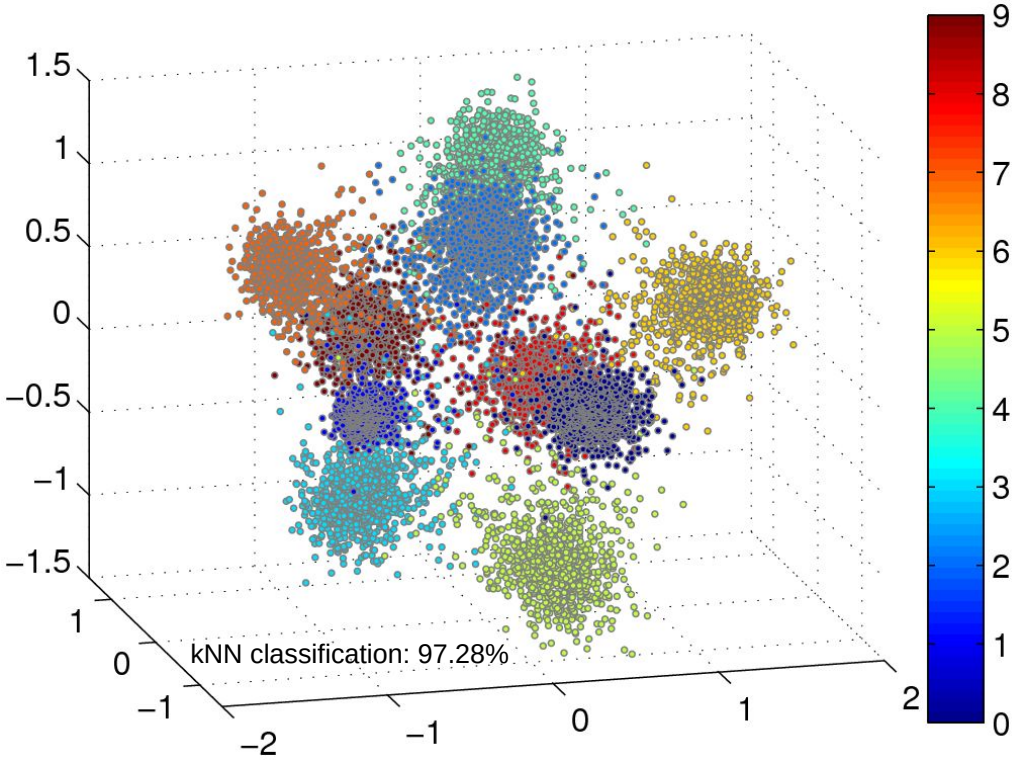
Running time: using the deep learning tool Caffe [79] in CPU mode only, we carried out all the experiments on a machine with a 2.3 GHz dual-core CPU, 4GB RAM and 64-bit operating system. In general, as DrLIM and TSML-Hybrid have the same deep CNN architecture, performing gradient descent once for either the triangular loss or the contrastive loss costs almost the same time. However, the hybrid training strategy helps TSML-Hybrid to train a model much faster. Concretely, the tiny-scale training costs about 150 seconds and the large-scale training costs about 960 seconds, hence tuning the above TSML-Hybrid models averagely costs 1110 seconds (see Table 5.3). For DrLIM that directly performs large-scale training on data pairs, the convergence speed slows down significantly than that of other methods, the average training time for each DrLIM model is about 9600 seconds.

Classification accuracy: a comparison on classification performance is summarized in Table 5.4. The classical CNN classifier – the present CNN followed by a Euclidean loss layer or softmax loss layer – is set as the baseline for classification. The proposed TSML-Hybrid method offers two views of the target space. Take a target space of dimension 10 for example,

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés



(a) In 2-dimensional space



(b) In 3-dimensional space

Fig. 5.16 Mapping results of the MNIST test data by DrLIM in (a) 2-dimensional and (b) 3-dimensional spaces.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 5.4 Comparison on classification accuracy of different methods. '—' means no available result. View 1: before unfolding; View 2: after unfolding.

Methods	1-dim.	2-dim.	3-dim.	10-dim.
CNN classifier [79]	—	—	—	98.97%
DrLIM [62]	—	94.16%	97.28%	98.35%
TSML-Hybrid (view 1)	—	94.76%	98.05%	**99.02%**
TSML-Hybrid (view 2)	**94.76%**	**98.05%**	**98.66%**	98.87%

View 1 is the result of TSML-Hybrid with the output layer size equal to 10; View 2 is the result of an output dimension 11 and then unfolded into the 10-dimensional space by coordinate transformation.

Firstly, we evaluate all the methods in the 10-dimensional space because the output of CNN is determined by the number of classes. Comparing these results we find that our TSML-Hybrid method obtains comparable classification results with the CNN classifier (98.97%) in the 10-dimensional space (99.02% and 98.87% in the two views respectively). However, DrLIM relatively performs worse than the other three (98.35%). This is because DrLIM trains on a limited number of data pairs and does not fully make use of all the labeled data samples.

Concerning the lower dimensional spaces, the CNN classifier failed for dimensionality reduction because the dimension of the handcrafted target vectors was fixed. DrLIM and TSML-Hybrid relaxes this constraint by metric learning and realizes classification in the visualizable spaces. Generally, TSML-Hybrid (View 2) gets better results than DrLIM and TSML-Hybrid (View 1) in these low dimensional spaces. For DrLIM and TSML-Hybrid (View 1), the classification accuracy decreases as the target dimension is reduced and a large decline occurs when the target dimension changes from 3 to 2. By using coordinate transformation, TSML-Hybrid (View 2) delays the decline till the dimension reduced from 2 to 1. Therefore, the View 2 is preferred if the classification performance is mainly concerned. Actually, better classification results have been published on the MNIST dataset[4] since the dataset was released in 1998. However, to the best of our knowledge, our results are the best in the visualizable spaces.

Comparison summary: a summary of comparison of the three methods is given in Table 5.5. Like the traditional CNN, the proposed TSML-Hybrid method employs a supervised training algorithm to perform efficient and effective optimization. Compared with the state-of-the-art method DrLIM, TSML-Hybrid has shown its superiority on both speed and classification result.

[4]http://yann.lecun.com/exdb/mnist/

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table 5.5 Comparison summary of different methods.

	CNN	DrLIM	TSML-Hybrid
Training mode	Supervised	Semi-supervised	Supervised
Training speed		Slower	
Data visualization	—	2/3-d	1/2/3-d
Classification accuracy		Lower	

5.4 Conclusion

In this chapter, we have studied the nonlinear variants of our TSML method by applications on both small-scale and large-scale datasets. Specifically, we had the triangular loss to define the pairwise data relationship in a target space and employed neural networks to realize nonlinear mappings.

Firstly, we integrated the commonly used neural networks, MLP, with the triangular loss, referred as TSML-MLP, for the problem of face identification on a small-scale dataset. We found that TSML-MLP behaved like the classical MLP in terms of classification performance. The major advantage of TSML-MLP is that unlike MLP requiring hand-crafted target vectors, TSML-MLP generates targets automatically and thus allows flexible dimensionality reduction into the target space. However, TSML-MLP costs more time for training because it learns on data pairs whose number are exponentially larger than that of data samples.

The increase of time cost may be unaffordable for experiments on large-scale datasets, thus we proposed a hybrid training algorithm to change the way of learning: (1) training a naive model on a few data pairs to produce target vectors; and then (2) training a mature model on all the data samples to achieve the goal of both dimensionality reduction and accurate classification. So that the training of the metric learning system can be as efficient as the traditional neural networks. We introduced CNN as the deep mapping function to project an image to a point in low dimensional spaces. More interestingly, the triangular loss enables us to benefit from classical manifold learning theories and visualize the output data in an even lower dimensional space. For example, we succeeded in mapping thousands of images in the MNIST dataset to the 1-dimensional space, i.e. a real interval, but having a competitive classification accuracy.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Chapter 6

Conclusion and Perspectives

6.1 Conclusion

In this thesis, we proposed a novel metric learning method based on the Triangular Similarity, referred as Triangular Similarity Metric Learning (TSML). We accomplished a thorough study of TSML by both theoretical analysis and experimental justification on different applications such as verification, classification and dimensionality reduction.

At the very beginning, we reviewed the literature of linear Metric Learning methods and also their non-linear variants, i.e. Siamese Neural Networks. We concluded that Siamese Neural Networks and Metric Learning can be regarded as *a unifying study* of designing a good architecture to learn a good metric from data pairs. In other words, a good Metric Learning system should be a collaborative product of formulating an effective metric-based cost function and designing a proper mapping function.

In terms of the cost function, we first proposed the *Triangular Similarity*, a novel similarity measurement which is equivalent to the Cosine Similarity. Based on a simplified version of the Triangular Similarity, we further developed the *triangular loss* function in order to perform metric learning, i.e. to increase the similarity between two vectors in the same class and to decrease the similarity between two vectors of different classes. After that, by examining the gradient function of the triangular loss, we found that its gradient has a similar formulation with the gradient function of the Mean Squared Error (MSE) that is widely used in neural networks for classification problems. Consequently, it allows us to employ the standard Backpropagation algorithm to perform optimization for the proposed TSML systems. Moreover, compared with other distance or similarity metrics, the triangular loss and its gradient naturally offer us an intuitive geometrical interpretation of the metric learning objective.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

In terms of the mapping function, we introduced three different options: a linear mapping realized by a simple transformation matrix, a nonlinear mapping realized by Multi-layer Perceptrons (MLP) and a deep nonlinear mapping realized by Convolutional Neural Networks (CNN). With these mapping functions, we presented three different TSML systems for various applications, namely, pairwise verification, object identification, dimensionality reduction and data visualization. For each application, we carried out extensive experiments on popular benchmarks and datasets to demonstrate the effectiveness of the proposed systems.

TSML-Linear for pairwise verification

We applied the proposed linear TSML system, referred as *TSML-Linear*, to a series of pairwise verification problems such as pairwise face verification, pairwise kinship verification and pairwise speaker verification. For all these tasks on pairwise verification, we first pointed out that the setting of limited training data for some classes and the setting of mutually exclusive training and test sets make such a kind of task suffering from over-fitting problems.

By extensive experimental validation, we presented several strategies and confirmed their effectiveness on reducing the risk of over-fitting. These strategies includes 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 additional regularization factors to strengthen generalization. With these strategies, the nature of learning a good metric of the TSML method makes itself effective on all the different pairwise verification tasks. Generally, one of the proposed approaches, TSML-Linear-Sim, has presented competitive performance with the state-of-the-art methods in all the experiments. Especially, it has achieved the best result on the KinFaceW-II dataset in the FG 2015 Kinship Verification Evaluation [112].

TSML-MLP for face identification

We applied the MLP-based nonlinear TSML system, referred as *TSML-MLP*, to a typical classification problem of face identification on a small-scale dataset. As the used triangular loss function automatically generates target vectors for input data pairs, the major advantage of TSML-MLP is that the target dimension can be arbitrarily specified and thus flexible dimensionality reduction is allowed. However, TSML-MLP learns on data pairs when a classical MLP trains on data samples, TSML-MLP costs more time for training because the number of data pairs is exponentially larger than that of data samples. In terms of classification performance, we found that TSML-MLP acted like the classical MLP that both

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

of them achieved competitive classification performance on state-of-the-art face descriptors such as Over-complete Local Binary Patterns (OCLBP).

TSML-Hybrid for end-to-end data visualization

We applied the CNN-based nonlinear TSML system, referred as *TSML-CNN*, to a large-scale problem of data visualization. TSML-CNN inherits the ability of flexible dimensionality reduction from TSML-MLP. However, on large-scale datasets, the traditional way of training on data pairs requires tremendous computational resources that TSML-CNN is hard to afford.

Hence we proposed a hybrid training algorithm, referred as *TSML-Hybrid* to change the way of learning: (1) training a TSML-CNN model on a few data pairs to produce target vectors; (2) transplanting the structure and weights of the TSML-CNN model into a new single-track CNN; (3) training the single-track CNN on all the data samples to achieve the goal of both dimensionality reduction and accurate classification. We demonstrated that the hybrid training is much faster than directly training a siamese model.

We applied this system for end-to-end data visualization on the MNIST dataset, i.e. projecting an image to a point in low dimensional spaces. Taking advantage of the fact that the triangular loss concerns the Cosine Similarity rather than the Euclidean distance, the length information of vectors can be ignored because the Triangular Similarity only uses the direction information to distinguish different vectors. Therefore, we employed manifold learning techniques to remove one more dimension from the CNN outputs and provided a new perspective to view the data distribution. The proposed TSML-Hybrid system achieved significantly better performance than the state-of-the art method for classification in low dimensional spaces.

6.2 Perspectives

TSML for applications on other kind of data

The proposed TSML systems mainly processed extracted image features or directly the raw images. For example, in both face verification and kinship verification tasks, TSML-Linear aimed at learning a better metric on face descriptors such as OCLBP, Fisher Vectors (FV); in face identification, TSML-MLP performed classification on the OCLBP features of face images as well; in end-to-end data visualization, TSML-Hybrid employed CNN to directly process images of hand-written digits. Only in pairwise speaker verification, we presented some work on identity vectors (i-vector). Hence one of the future directions is to develop TSML systems for more biometric data such as iris, voice, fingerprint, hand and

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

signature [50, 85], or for structured data such as strings [12], text documents [177] and 3D objects [61]. According to our experience, much efforts will be made in crafting feature representation or designing a deep architecture to automatically learn features from raw data, and developing generalization techniques to reduce the risk of over-fitting.

TSML in an unsupervised mode

All the proposed TSML systems performed learning in a supervised mode, but unsupervised learning is always an interesting and challenging problem in machine learning. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object [99]. It will be very interesting to see if unsupervised TSML systems can achieve comparable performance with the reported results by supervised models in this thesis.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

References

[1] Ahonen, T., Hadid, A., and Pietikäinen, M. (2004). Face recognition with local binary patterns. In *Computer Vision-ECCV 2004*, pages 469–481. Springer.

[2] Alpaydin, E. (2014). *Introduction to machine learning*. MIT press.

[3] Arashloo, S. R. and Kittler, J. (2013). Efficient processing of mrfs for unconstrained-pose face recognition. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8. IEEE.

[4] Arashloo, S. R. and Kittler, J. (2014). Class-specific kernel fusion of multiple descriptors for face verification using multiscale binarised statistical image features. *Information Forensics and Security, IEEE Transactions on*, 9(12):2100–2109.

[5] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern information retrieval*, volume 463. ACM press New York.

[6] Baldi, P. and Chauvin, Y. (1993). Neural networks for fingerprint recognition. *Neural Computation*, 5(3):402–418.

[7] Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. (2005). Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(6):937–965.

[8] Barkan, O., Weill, J., Wolf, L., and Aronowitz, H. (2013). Fast high dimensional vector multiplication face recognition. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1960–1967. IEEE.

[9] Battiti, R. and Brunato, M. (2014). *The LION Way: Machine Learning plus Intelligent Optimization*. LIONlab, University of Trento, Italy.

[10] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202.

[11] Bellet, A. and Habrard, A. (2015). Robustness and generalization for metric learning. *Neurocomputing*, 151(1):259–267.

[12] Bellet, A., Habrard, A., and Sebban, M. (2012). Good edit similarity learning by loss minimization. *Machine Learning*, 89(1-2):5–35.

[13] Bellet, A., Habrard, A., and Sebban, M. (2013). A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[14] Bellet, A., Habrard, A., and Sebban, M. (2015). *Metric Learning*, volume 9 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers (USA), Synthesis Lectures on Artificial Intelligence and Machine Learning, pp 1-151.

[15] Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[16] Bengio, Y., Goodfellow, I. J., and Courville, A. (2015). Deep learning. Book in preparation for MIT Press.

[17] Berg, T. and Belhumeur, P. N. (2012). Tom-vs-pete classifiers and identity-preserving alignment for face verification. In *BMVC*, volume 2, page 7. Citeseer.

[18] Berlemont, S., Lefebvre, G., Duffner, S., and Garcia, C. (2015). Siamese neural network based similarity metric for inertial gesture classification and rejection. In *11th IEEE International Conference on Automatic Face and Gesture Recognition*.

[19] Bertsekas, D. P. (1976). On the goldstein-levitin-polyak gradient projection method. *Automatic Control, IEEE Transactions on*, 21(2):174–184.

[20] Blake, C. and Merz, C. J. (1998). UCI repository of machine learning databases.

[21] Bourlard, H. and Wellekens, C. J. (1990). Links between markov models and multilayer perceptrons. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(12):1167–1178.

[22] Bouvrie, J. (2006). Notes on convolutional neural networks.

[23] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[24] Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., and Shah, R. (1993). Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.

[25] Bruno, E., Moenne-Loccoz, N., and Marchand-Maillet, S. (2006). Asymmetric learning and dissimilarity spaces for content-based retrieval. In *Image and Video Retrieval*, pages 330–339. Springer.

[26] Cao, Q., Ying, Y., and Li, P. (2013). Similarity metric learning for face recognition. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2408–2415. IEEE.

[27] Chang, C.-C. and Lin, C.-J. (2011). Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

[28] Chechik, G., Shalit, U., Sharma, V., and Bengio, S. (2009). An online algorithm for large scale image similarity learning. In *Advances in Neural Information Processing Systems*, pages 306–314.

[29] Chechik, G., Sharma, V., Shalit, U., and Bengio, S. (2010). Large scale online learning of image similarity through ranking. *The Journal of Machine Learning Research*, 11:1109–1135.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

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

[31] Chen, K. and Salman, A. (2011). Extracting speaker-specific information with a regularized siamese deep network. In *Advances in Neural Information Processing Systems*, pages 298–306.

[32] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, IEEE Conference on*, volume 1, pages 539–546. IEEE.

[33] Cottrell, G. W. and Metcalfe, J. (1990). Empath: face, emotion, and gender recognition using holons. In *Advances in Neural Information Processing Systems*, pages 564–571. Morgan Kaufmann Publishers Inc.

[34] Cover, T. M. and Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

[35] Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

[36] Cui, Z., Li, W., Xu, D., Shan, S., and Chen, X. (2013). Fusing robust face region descriptors via multiple metric learning for face recognition in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3554–3561. IEEE.

[37] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR), 2005 IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE.

[38] Daugman, J. G. (1988). Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(7):1169–1179.

[39] Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, pages 209–216. ACM.

[40] Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., and Ouellet, P. (2011). Front-end factor analysis for speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(4):788–798.

[41] Deng, J., Berg, A. C., and Fei-Fei, L. (2011). Hierarchical semantic indexing for large scale image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 785–792. IEEE.

[42] Deng, W., Hu, J., and Guo, J. (2005). Gabor-eigen-whiten-cosine: a robust scheme for face recognition. In *Analysis and Modelling of Faces and Gestures*, pages 336–349. Springer.

[43] Duffner, S. (2008). Face image analysis with convolutional neural networks.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[44] Dunteman, G. H. (1989). *Principal components analysis*. Number 69. Sage.

[45] Estrach, J. B., Szlam, A., and Lecun, Y. (2014). Signal recovery from pooling representations. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 307–315.

[46] Everingham, M., Sivic, J., and Zisserman, A. (2006). Hello! my name is... buffy"–automatic naming of characters in tv video. In *BMVC*, volume 2, page 6.

[47] Fan, H., Cao, Z., Jiang, Y., Yin, Q., and Doudou, C. (2014). Learning deep face representation. *arXiv preprint arXiv:1403.2802*.

[48] Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

[49] Garcia, C. and Delakis, M. (2004). Convolutional face finder: A neural architecture for fast and robust face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1408–1423.

[50] Garcia-Salicetti, S., Beumier, C., Chollet, G., Dorizzi, B., Jardins, J., Lunter, J., Ni, Y., and Petrovska-Delacrétaz, D. (2003). BIOMET: A multimodal person authentication database including face, voice, fingerprint, hand and signature modalities. In *Audio-and Video-Based Biometric Person Authentication*, pages 1056–1056. Springer.

[51] Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(6):643–660.

[52] Globerson, A. and Roweis, S. T. (2005). Metric learning by collapsing classes. In *Advances in neural information processing systems*, pages 451–458.

[53] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

[54] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323.

[55] Goldberger, J., Hinton, G. E., Roweis, S. T., and Salakhutdinov, R. (2004). Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520.

[56] Golomb, B. A., Lawrence, D. T., and Sejnowski, T. J. (1991). Sexnet: A neural network identifies sex from human faces. In *Advances in Neural Information Processing Systems*, pages 572–579.

[57] Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[58] Greenberg, C. S., Bansé, D., Doddington, G. R., Garcia-Romero, D., Godfrey, J. J., Kinnunen, T., Martin, A. F., McCree, A., Przybocki, M., and Reynolds, D. A. (2014). The NIST 2014 speaker recognition i-vector machine learning challenge. In *Odyssey: The Speaker and Language Recognition Workshop*.

[59] Guillaumin, M., Mensink, T., Verbeek, J., and Schmid, C. (2009a). Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 309–316. IEEE.

[60] Guillaumin, M., Verbeek, J., and Schmid, C. (2009b). Is that you? metric learning approaches for face identification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 498–505. IEEE.

[61] Hachani, M., Zaid, A. O., and Puech, W. (2014). 3D non-rigid pattern recognition based on structural analysis. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 3131–3135. IEEE.

[62] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE.

[63] Hassner, T., Harel, S., Paz, E., and Enbar, R. (2014). Effective face frontalization in unconstrained images. *arXiv preprint arXiv:1411.7964*.

[64] Hatch, A. O., Kajarekar, S. S., and Stolcke, A. (2006). Within-class covariance normalization for svm-based speaker recognition. In *Interspeech*.

[65] Hatch, A. O. and Stolcke, A. (2006). Generalized linear kernels for one-versus-all classification: application to speaker recognition. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, pages V–V. IEEE.

[66] Heikkilä, M., Pietikäinen, M., and Schmid, C. (2006). Description of interest regions with center-symmetric local binary patterns. In *Computer Vision, Graphics and Image Processing*, pages 58–69. Springer.

[67] Higham, N. J. (2002). *Accuracy and stability of numerical algorithms*. Siam.

[68] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[69] Hu, J., Lu, J., and Tan, Y.-P. (2014). Discriminative deep metric learning for face verification in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1875–1882. IEEE.

[70] Hu, J., Lu, J., and Tan, Y.-P. (2015a). Deep transfer metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 325–333.

[71] Hu, J., Lu, J., Yuan, J., and Tan, Y.-P. (2015b). Large margin multi-metric learning for face and kinship verification in the wild. In *Computer Vision–ACCV 2014*, pages 252–267. Springer.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[72] Huang, C., Zhu, S., and Yu, K. (2012). Large scale strongly supervised ensemble metric learning, with applications to face verification and retrieval. *arXiv preprint arXiv:1212.6094*.

[73] Huang, G. B., Jain, V., and Learned-Miller, E. (2007a). Unsupervised joint alignment of complex images. In *Computer Vision, 2007 IEEE 11th International Conference on*, pages 1–8. IEEE.

[74] Huang, G. B. and Learned-Miller, E. (2014). Labeled faces in the wild: Updates and new reporting procedures. *Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep*, pages 14–003.

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

[76] Hussain, S. U., Napoléon, T., and Jurie, F. (2012). Face recognition using local quantized patterns. In *British Machive Vision Conference*, pages 11–pages.

[77] Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430.

[78] Jain, P., Kulis, B., Dhillon, I. S., and Grauman, K. (2009). Online metric learning and fast similarity search. In *Advances in neural information processing systems*, pages 761–768.

[79] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM.

[80] Kan, M., Shan, S., Xu, D., and Chen, X. (2011). Side-information based linear discriminant analysis for face recognition. In *BMVC*, volume 11, pages 125–1.

[81] Ke, Y. and Sukthankar, R. (2004). Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE.

[82] Kedem, D., Tyree, S., Sha, F., Lanckriet, G. R., and Weinberger, K. Q. (2012). Non-linear metric learning. In *Advances in Neural Information Processing Systems*, pages 2573–2581.

[83] Kenny, P., Ouellet, P., Dehak, N., Gupta, V., and Dumouchel, P. (2008). A study of interspeaker variability in speaker verification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(5):980–988.

[84] Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *EMNLP*, pages 388–395. Citeseer.

[85] Krichen, E., Mellakh, M. A., Garcia-Salicetti, S., and Dorizzi, B. (2004). Iris identification using wavelet packets. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 4, pages 335–338. IEEE.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[86] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

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

[88] Larcher, A., Lee, K. A., Ma, B., and Li, H. (2014). Text-dependent speaker verification: Classifiers, databases and RSR2015. *Speech Communication*, 60:56–77.

[89] Law, M., Thome, N., and Cord, M. (2013). Quadruplet-wise image similarity learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 249–256.

[90] Law, M., Thome, N., and Cord, M. (2014). Fantope regularization in metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1051–1058.

[91] Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113.

[92] Le Barz, C., Thome, N., Cord, M., Herbin, S., and Sanfourche, M. (2015). Exemplar based metric learning for robust visual localization. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 4342–4346. IEEE.

[93] Learned-Miller, E., Huang, G., RoyChowdhury, A., Li, H., and Hua, G. (2015). Labeled faces in the wild: A survey.

[94] LeCun, B. B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer.

[95] LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10).

[96] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[97] LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. (2006). A tutorial on energy-based learning. *Predicting structured data*, 1.

[98] LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE.

[99] LeCun, Y. A., Bengio, Y., and Hinton, G. E. (2015). Deep learning. *Nature*, 521:436–444.

[100] LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[101] Li, H. and Hua, G. (2015). Hierarchical-PEP model for real-world face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4055–4064.

[102] Li, H., Hua, G., Lin, Z., Brandt, J., and Yang, J. (2013). Probabilistic elastic matching for pose variant face verification. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3499–3506. IEEE.

[103] Li, H., Hua, G., Shen, X., Lin, Z., and Brandt, J. (2015). Eigen-pep for video face recognition. In *Computer Vision–ACCV 2014*, pages 17–33. Springer.

[104] Li, X., Shen, C., Shi, Q., Dick, A., and Van den Hengel, A. (2012). Non-sparse linear representations for visual tracking with online reservoir metric learning. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1760–1767. IEEE.

[105] Lienhart, R. and Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In *Image Processing, 2002 International Conference on*, volume 1, pages I–900. IEEE.

[106] Lippmann, R. P. (1989). Review of neural networks for speech recognition. *Neural Computation*, 1(1):1–38.

[107] Lipton, Z. C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

[108] Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528.

[109] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[110] Lloyd, S. P. (1982). Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137.

[111] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[112] Lu, J., Hu, J., Liong, V. E., Zhou, X., Bottino, A., Islam, I. U., Vieira, T. F., Qin, X., Tan, X., Chen, S., et al. (2015). The FG 2015 kinship verification in the wild evaluation. In *11th IEEE International Conference on Automatic Face and Gesture Recognition*.

[113] Lu, J., Hu, J., Zhou, X., Zhou, J., Castrillón-Santana, M., Lorenzo-Navarro, J., Kou, L., Shang, Y., Bottino, A., and Figueiredo Vieira, T. (2014a). Kinship verification in the wild: The first kinship verification competition. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, pages 1–6. IEEE.

[114] Lu, J., Zhou, X., Tan, Y.-P., Shang, Y., and Zhou, J. (2014b). Neighborhood repulsed metric learning for kinship verification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(2):331–345.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[115] Luenberger, D. G. (1973). *Introduction to linear and nonlinear programming*, volume 28. Addison-Wesley Reading, MA.

[116] MacKay, D. J. (2003). *Information theory, inference and learning algorithms*. Cambridge university press.

[117] Martinez, A. M. (1998). The AR face database. *CVC Technical Report*, 24.

[118] Matas, J., Hamouz, M., Jonsson, K., Kittler, J., Li, Y., Kotropoulos, C., Tefas, A., Pitas, I., Tan, T., Yan, H., et al. (2000). Comparison of face verification results on the XM2VTFS database. In *Pattern Recognition, 15th International Conference on*, volume 4, pages 858–863. IEEE.

[119] McFee, B. and Lanckriet, G. R. (2010). Metric learning to rank. In *Proceedings of the 27th International Conference on Machine Learning*, pages 775–782.

[120] Mendelson, B. (1990). *Introduction to topology*. Courier Corporation.

[121] Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. (2012). Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *Proceedings of the 12th European Conference on Computer Vision*, pages 488–501. Springer.

[122] Mobahi, H., Collobert, R., and Weston, J. (2009). Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM.

[123] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

[124] Ng, A. Y., Jordan, M. I., Weiss, Y., et al. (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856.

[125] Nguyen, H. V. and Bai, L. (2011). Cosine similarity metric learning for face verification. In *Computer Vision–ACCV 2010*, pages 709–720. Springer.

[126] Nielsen, M. (2015). *Neural Networks and Deep Learning*.

[127] Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., and Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *Image Processing, IEEE Transactions on*, 14(9):1360–1371.

[128] Ojala, T., Pietikäinen, M., and Mäenpää, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):971–987.

[129] Orr, G. B. and Müller, K.-R. (2003). *Neural networks: tricks of the trade*. Springer.

[130] Ouamane, A., Messaoud, B., Guessoum, A., Hadid, A., and Cheriet, M. (2014). Multi scale multi descriptor local binary features and exponential discriminant analysis for robust face authentication. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 313–317. IEEE.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[131] Parameswaran, S. and Weinberger, K. Q. (2010). Large margin multi-task metric learning. In *Advances in neural information processing systems*, pages 1867–1875.

[132] Petersen, K. B., Pedersen, M. S., et al. (2008). The matrix cookbook. *Technical University of Denmark*, 7:15.

[133] Pinto, N., DiCarlo, J. J., and Cox, D. D. (2009). How far can you get with a modern face recognition test set using only simple features? In *Computer Vision and Pattern Recognition, 2009 IEEE Conference on*, pages 2591–2598. IEEE.

[134] Prechelt, L. (2012). Early stopping - but when? In *Neural Networks: Tricks of the Trade*, pages 53–67. Springer.

[135] Qamar, A. M. and Gaussier, E. (2009). Online and batch learning of generalized cosine similarities. In *Data Mining, 2009 IEEE International Conference on*, pages 926–931. IEEE.

[136] Qamar, A. M., Gaussier, E., Chevallet, J.-P., and Lim, J. H. (2008). Similarity learning for nearest neighbor classification. In *Data Mining, 2008 IEEE International Conference on*, pages 983–988. IEEE.

[137] Qi, G.-J., Tang, J., Zha, Z.-J., Chua, T.-S., and Zhang, H.-J. (2009). An efficient sparse metric learning in high-dimensional space via l1-penalized log-determinant regularization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 841–848. ACM.

[138] Reynolds, D. A. (1995). Speaker identification and verification using gaussian mixture speaker models. *Speech communication*, 17(1):91–108.

[139] Reynolds, D. A., Quatieri, T. F., and Dunn, R. B. (2000). Speaker verification using adapted gaussian mixture models. *Digital signal processing*, 10(1):19–41.

[140] Rizvi, S., Phillips, J., Moon, H., et al. (1998). The FERET verification testing protocol for face recognition algorithms. In *Automatic Face and Gesture Recognition, 1998 IEEE International Conference on*, pages 48–53. IEEE.

[141] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

[142] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

[143] Sanderson, C. and Lovell, B. C. (2009). Multi-region probabilistic histograms for robust and scalable identity inference. In *Advances in Biometrics*, pages 199–208. Springer.

[144] Schmidt, M. (2012). Minfunc: unconstrained differentiable multivariate optimization in matlab. *URL: http://www.di.ens.fr/mschmidt/Software/minFunc.html*.

[145] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *arXiv preprint arXiv:1503.03832*.

[146] Seo, H. J. and Milanfar, P. (2011). Face verification using the LARK representation. *Information Forensics and Security, IEEE Transactions on*, 6(4):1275–1286.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[147] Shalev-Shwartz, S., Singer, Y., and Ng, A. Y. (2004). Online and batch learning of pseudo-metrics. In *Proceedings of the twenty-first international conference on Machine learning*, page 94. ACM.

[148] Shental, N., Hertz, T., Weinshall, D., and Pavel, M. (2002). Adjustment learning and relevant component analysis. In *Computer Vision–ECCV 2002*, pages 776–790. Springer.

[149] Sim, T., Baker, S., and Bsat, M. (2002). The CMU pose, illumination, and expression (PIE) database. In *Automatic Face and Gesture Recognition, 2002 IEEE International Conference on*, pages 46–51. IEEE.

[150] Simonyan, K., Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2013). Fisher vector faces in the wild. In *British Machine Vision Conference*.

[151] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[152] Sivic, J. and Zisserman, A. (2009). Efficient visual search of videos cast as text retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):591–606.

[153] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

[154] Stone, C. J. (1977). Consistent nonparametric regression. *The annals of statistics*, pages 595–620.

[155] Sun, Y., Chen, Y., Wang, X., and Tang, X. (2014a). Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems*, pages 1988–1996.

[156] Sun, Y., Liang, D., Wang, X., and Tang, X. (2015). DeepID3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*.

[157] Sun, Y., Wang, X., and Tang, X. (2014b). Deep learning face representation from predicting 10,000 classes. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1891–1898. IEEE.

[158] Sun, Y., Wang, X., and Tang, X. (2014c). Deeply learned face representations are sparse, selective, and robust. *arXiv preprint arXiv:1412.1265*.

[159] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[160] Taigman, Y., Wolf, L., Hassner, T., et al. (2009). Multiple one-shots for utilizing class label information. In *BMVC*, pages 1–12.

[161] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1701–1708. IEEE.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[162] Turk, M., Pentland, A. P., et al. (1991). Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991 IEEE Computer Society Conference on*, pages 586–591. IEEE.

[163] Vedaldi, A. and Fulkerson, B. (2010). VLFeat: An open and portable library of computer vision algorithms. In *Proceedings of International Conference on Multimedia*, pages 1469–1472. ACM.

[164] Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

[165] Wang, J., Sun, K., Sha, F., Marchand-Maillet, S., and Kalousis, A. (2014). Two-stage metric learning. In *Proceedings of the 31th International Conference on Machine Learning (ICML), Beijing, China*, pages 370–378.

[166] Wang, X. and Tang, X. (2004). A unified framework for subspace face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1222–1228.

[167] Weinberger, K. Q., Blitzer, J., and Saul, L. K. (2005). Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480.

[168] Weinberger, K. Q. and Saul, L. K. (2008). Fast solvers and efficient implementations for distance metric learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1160–1167. ACM.

[169] Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244.

[170] Wolf, L., Hassner, T., and Maoz, I. (2011). Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE.

[171] Wolf, L., Hassner, T., and Taigman, Y. (2008). Descriptor based methods in the wild. In *Workshop on Faces in'Real-Life'Images: Detection, Alignment, and Recognition*.

[172] Xie, L., Zheng, L., Liu, Z., and Zhang, Y. (2012). Laplacian eigenmaps for automatic story segmentation of broadcast news. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):276–289.

[173] Xing, E. P., Jordan, M. I., Russell, S., and Ng, A. Y. (2002). Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512.

[174] Yan, H., Lu, J., Deng, W., and Zhou, X. (2014a). Discriminative multimetric learning for kinship verification. *Information Forensics and Security, IEEE Transactions on*, 9(7):1169–1178.

[175] Yan, H.-C., Lu, J., and Zhou, X. (2014b). Prototype-based discriminative feature learning for kinship verification.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

[176] Yi, D., Lei, Z., and Li, S. Z. (2013). Towards pose robust face recognition. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3539–3545. IEEE.

[177] Yih, W.-t., Toutanova, K., Platt, J. C., and Meek, C. (2011). Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 247–256. Association for Computational Linguistics.

[178] Ying, Y. and Li, P. (2012). Distance metric learning with eigenvalue optimization. *The Journal of Machine Learning Research*, 13(1):1–26.

[179] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer.

[180] Zhang, L., Chu, R., Xiang, S., Liao, S., and Li, S. Z. (2007). Face detection based on multi-block LBP representation. In *Advances in biometrics*, pages 11–18. Springer.

[181] Zhang, Z., Lyons, M., Schuster, M., and Akamatsu, S. (1998). Comparison between geometry-based and gabor-wavelets-based facial expression recognition using multi-layer perceptron. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 454–459. IEEE.

[182] Zheng, L., Duffner, S., Idrissi, K., Garcia, C., and Baskurt, A. (2015a). Siamese multi-layer perceptrons for dimensionality reduction and face identification. *Multimedia Tools and Applications*.

[183] Zheng, L., Idrissi, K., Garcia, C., Duffner, S., and Baskurt, A. (2015b). Logistic similarity metric learning for face verification. In *Acoustics, Speech and Signal Processing, 2015 IEEE International Conference on*. IEEE.

[184] Zheng, L., Idrissi, K., Garcia, C., Duffner, S., and Baskurt, A. (2015c). Triangular similarity metric learning for face verification. In *11th IEEE International Conference on Automatic Face and Gesture Recognition*.

[185] Zheng, L., Leung, C.-C., Xie, L., Ma, B., and Li, H. (2012). Acoustic texttiling for story segmentation of spoken documents. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5121–5124. IEEE.

[186] Zhou, E., Cao, Z., and Yin, Q. (2015). Naive-deep face recognition: Touching the limit of LFW benchmark or not? *arXiv preprint arXiv:1501.04690*.

[187] Zhu, Z., Luo, P., Wang, X., and Tang, X. (2014). Recover canonical-view faces in the wild with deep neural networks. *arXiv preprint arXiv:1404.3543*.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Appendix A

Derivatives

A.1 Derivative of the vector norm

Given a column vector $\mathbf{x}$ has n elements $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, the L2 norm of the vector denotes the vector length, prove that

$$\frac{\partial \|\mathbf{x}\|}{\partial \mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}. \tag{A.1}$$

Proof: we start the proof from the right side,

$$\begin{aligned}
\frac{\partial \|\mathbf{x}\|}{\partial \mathbf{x}} &= \frac{\partial \sqrt{\mathbf{x}^T\mathbf{x}}}{\partial \mathbf{x}} = \frac{\partial (\mathbf{x}^T\mathbf{x})^{\frac{1}{2}}}{\partial \mathbf{x}} \\
&= \frac{1}{2}(\mathbf{x}^T\mathbf{x})^{-\frac{1}{2}}\frac{\partial (\mathbf{x}^T\mathbf{x})}{\partial \mathbf{x}} = \frac{1}{2}\frac{1}{(\mathbf{x}^T\mathbf{x})^{\frac{1}{2}}}\frac{\partial (\mathbf{x}^T\mathbf{x})}{\partial \mathbf{x}} \\
&= \frac{1}{2\|\mathbf{x}\|}\frac{\partial (\mathbf{x}^T\mathbf{x})}{\partial \mathbf{x}},
\end{aligned}$$

where $\mathbf{x}^T\mathbf{x} = \sum_{i=1}^{n} x_i^2$. The derivative of this inner product with respect to the vector $\mathbf{x}$ is simply a vector of derivatives on each elements:

$$\begin{aligned}
\frac{\partial (\mathbf{x}^T\mathbf{x})}{\partial \mathbf{x}} &= \frac{\partial \sum_{i=1}^{n} x_i^2}{\partial \mathbf{x}} = \frac{\partial \sum_{i=1}^{n} x_i^2}{\partial [x_1, x_2, \ldots, x_n]^T} \\
&= [\frac{\partial \sum_{i=1}^{n} x_i^2}{x_1}, \frac{\partial \sum_{i=1}^{n} x_i^2}{x_2}, \ldots, \frac{\partial \sum_{i=1}^{n} x_i^2}{x_n}]^T \\
&= [2x_1, 2x_2, \cdots, 2x_n]^T \\
&= 2\mathbf{x}.
\end{aligned}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Putting the above equations together:

$$\frac{\partial\|\mathbf{x}\|}{\partial\mathbf{x}} = \frac{1}{2\|\mathbf{x}\|}\frac{\partial(\mathbf{x}^T\mathbf{x})}{\partial\mathbf{x}} = \frac{1}{2\|\mathbf{x}\|}2\mathbf{x} = \frac{\mathbf{x}}{\|\mathbf{x}\|}.$$

QED.

A.2 Derivative of the bilinear similarity

The bilinear similarity is parameterized by a matrix $\mathbf{A}$ as $(\mathbf{Ax})^T\mathbf{Ay}$ [29], prove that

$$\frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial\mathbf{A}} = \mathbf{A}(\mathbf{xy}^T + \mathbf{yx}^T). \tag{A.2}$$

Proof: assuming $\mathbf{x}$ and $\mathbf{y}$ are n-dimensional column vectors that $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$, the transformation matrix $\mathbf{A}$ is of size $m \times n$, where $\mathbf{a}_i$ denotes the i_{th} row in the matrix and a_{ij} is the element at the intersection of the i_{th} row and the j_{th} column.

For the inner product between $\mathbf{Ax}$ and $\mathbf{Ay}$, its derivative with respect to the matrix $\mathbf{A}$ is simply a matrix of derivatives on each elements:

$$\frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial\mathbf{A}} = \begin{bmatrix} \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{11}} & \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{12}} & \cdots & \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{1n}} \\ \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{21}} & \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{22}} & \cdots & \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{m1}} & \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{m2}} & \cdots & \frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{mn}} \end{bmatrix}.$$

Specifically, as we have $\mathbf{Ax} = [\sum_{k=1}^n a_{1k}x_k, \sum_{k=1}^n a_{2k}x_k, \ldots, \sum_{k=1}^n a_{mk}x_k]^T$, the $(i,j)_{th}$ element in this derivative matrix can be unfolded as:

$$\begin{aligned}
&\frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{ij}} \\
&= \frac{\partial[\sum_{k=1}^n a_{1k}x_k, \sum_{k=1}^n a_{2k}x_k, \ldots, \sum_{k=1}^n a_{mk}x_k][\sum_{k=1}^n a_{1k}y_k, \sum_{k=1}^n a_{2k}y_k, \ldots, \sum_{k=1}^n a_{mk}y_k]^T}{\partial a_{ij}} \\
&= \frac{\partial \sum_{k=1}^n a_{1k}x_k \sum_{k=1}^n a_{1k}y_k + \sum_{k=1}^n a_{2k}x_k \sum_{k=1}^n a_{2k}y_k + \cdots + \sum_{k=1}^n a_{mk}x_k \sum_{k=1}^n a_{mk}y_k}{\partial a_{ij}} \\
&= \frac{\partial(\sum_{k=1}^n a_{ik}x_k \sum_{k=1}^n a_{ik}y_k)}{\partial a_{ij}} \\
&= \sum_{k=1}^n a_{ik}y_k \frac{\partial(\sum_{k=1}^n a_{ik}x_k)}{\partial a_{ij}} + \sum_{k=1}^n a_{ik}x_k \frac{\partial(\sum_{k=1}^n a_{ik}y_k)}{\partial a_{ij}} \\
&= x_j \sum_{k=1}^n a_{ik}y_k + y_j \sum_{k=1}^n a_{ik}x_k,
\end{aligned}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

this equation can be rewritten as a vectorization implementation:

$$\frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial a_{ij}} = x_j\mathbf{a}_i\mathbf{y} + y_j\mathbf{a}_i\mathbf{x},$$

where $\mathbf{a}_i$ is the i_{th} row of the matrix $\mathbf{A}$. Hence the derivative matrix is:

$$\begin{aligned}
\frac{\partial(\mathbf{Ax})^T\mathbf{Ay}}{\partial\mathbf{A}} &= \begin{bmatrix} x_1\mathbf{a}_1\mathbf{y}+y_1\mathbf{a}_1\mathbf{x} & x_2\mathbf{a}_1\mathbf{y}+y_2\mathbf{a}_1\mathbf{x} & \dots & x_n\mathbf{a}_1\mathbf{y}+y_n\mathbf{a}_1\mathbf{x} \\ x_1\mathbf{a}_2\mathbf{y}+y_1\mathbf{a}_2\mathbf{x} & x_2\mathbf{a}_2\mathbf{y}+y_2\mathbf{a}_2\mathbf{x} & \dots & x_n\mathbf{a}_2\mathbf{y}+y_n\mathbf{a}_2\mathbf{x} \\ \vdots & \vdots & \ddots & \vdots \\ x_1\mathbf{a}_m\mathbf{y}+y_1\mathbf{a}_m\mathbf{x} & x_2\mathbf{a}_m\mathbf{y}+y_2\mathbf{a}_m\mathbf{x} & \dots & x_n\mathbf{a}_m\mathbf{y}+y_n\mathbf{a}_m\mathbf{x} \end{bmatrix} \\
&= \begin{bmatrix} x_1\mathbf{a}_1\mathbf{y} & x_2\mathbf{a}_1\mathbf{y} & \dots & x_n\mathbf{a}_1\mathbf{y} \\ x_1\mathbf{a}_2\mathbf{y} & x_2\mathbf{a}_2\mathbf{y} & \dots & x_n\mathbf{a}_2\mathbf{y} \\ \vdots & \vdots & \ddots & \vdots \\ x_1\mathbf{a}_m\mathbf{y} & x_2\mathbf{a}_m\mathbf{y} & \dots & x_n\mathbf{a}_m\mathbf{y} \end{bmatrix} + \begin{bmatrix} y_1\mathbf{a}_1\mathbf{x} & y_2\mathbf{a}_1\mathbf{x} & \dots & y_n\mathbf{a}_1\mathbf{x} \\ y_1\mathbf{a}_2\mathbf{x} & y_2\mathbf{a}_2\mathbf{x} & \dots & y_n\mathbf{a}_2\mathbf{x} \\ \vdots & \vdots & \ddots & \vdots \\ y_1\mathbf{a}_m\mathbf{x} & y_2\mathbf{a}_m\mathbf{x} & \dots & y_n\mathbf{a}_m\mathbf{x} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{a}_1\mathbf{y} \\ \mathbf{a}_2\mathbf{y} \\ \vdots \\ \mathbf{a}_m\mathbf{y} \end{bmatrix}\mathbf{x}^T + \begin{bmatrix} \mathbf{a}_1\mathbf{x} \\ \mathbf{a}_2\mathbf{x} \\ \vdots \\ \mathbf{a}_m\mathbf{x} \end{bmatrix}\mathbf{y}^T \\
&= \mathbf{Ayx}^T + \mathbf{Axy}^T \\
&= \mathbf{A}(\mathbf{yx}^T + \mathbf{xy}^T).
\end{aligned}$$

QED.

A.3 Derivative of the parameterized vector norm

If the vector norm is parameterized by a matrix $\mathbf{W}$: $\|\mathbf{Wx}\|$, prove that

$$\frac{\partial\|\mathbf{Wx}\|}{\partial\mathbf{W}} = \frac{\mathbf{Wxx}^T}{\|\mathbf{Wx}\|}. \tag{A.3}$$

Proof: we start from the right side:

$$\frac{\partial\|\mathbf{Wx}\|}{\partial\mathbf{W}} = \frac{\partial((\mathbf{Wx})^T\mathbf{Wx})^{\frac{1}{2}}}{\partial\mathbf{W}} = \frac{1}{2}\frac{1}{((\mathbf{Wx})^T\mathbf{Wx})^{\frac{1}{2}}}\frac{\partial(\mathbf{Wx})^T\mathbf{Wx}}{\partial\mathbf{W}} = \frac{1}{2\|(\mathbf{Wx})\|}\frac{\partial(\mathbf{Wx})^T\mathbf{Wx}}{\partial\mathbf{W}},$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

according to Equation (A.2), we know that $\frac{\partial(\mathbf{Wx})^T\mathbf{Wx}}{\partial\mathbf{W}} = \mathbf{W}(\mathbf{x}\mathbf{x}^T + \mathbf{x}\mathbf{x}^T) = 2\mathbf{W}\mathbf{x}\mathbf{x}^T$, thus

$$\frac{\partial\|\mathbf{Wx}\|}{\partial\mathbf{W}} = \frac{1}{2\|(\mathbf{Wx})\|}\frac{\partial(\mathbf{Wx})^T\mathbf{Wx}}{\partial\mathbf{W}} = \frac{1}{2\|(\mathbf{Wx})\|}2\mathbf{W}\mathbf{x}\mathbf{x}^T = \frac{\mathbf{W}\mathbf{x}\mathbf{x}^T}{\|\mathbf{Wx}\|}.$$

QED.

A.4 Derivative of the Cosine Similarity

The Cosine Similarity is parameterized by a matrix $\mathbf{W}$ as:

$$s_\mathbf{W}(\mathbf{x},\mathbf{y}) = \frac{(\mathbf{Wx})^T\mathbf{Wy}}{\|\mathbf{Wx}\|\|\mathbf{Wy}\|},$$

prove that

$$\frac{\partial s_\mathbf{W}(\mathbf{x},\mathbf{y})}{\partial\mathbf{W}} = \frac{-1}{\|\mathbf{Wx}\|\|\mathbf{Wy}\|}[(\frac{(\mathbf{Wx})^T\mathbf{Wy}}{\|\mathbf{Wx}\|^2}\mathbf{Wx} - \mathbf{Wy})\mathbf{x}^T + (\frac{(\mathbf{Wx})^T\mathbf{Wy}}{\|\mathbf{Wy}\|^2}\mathbf{Wy} - \mathbf{Wx})\mathbf{y}^T]. \quad (A.4)$$

Proof: let $u(\mathbf{W})$ denote $(\mathbf{Wx})^T\mathbf{Wy}$ and $v(\mathbf{W})$ denote $\|\mathbf{Wx}\|\|\mathbf{Wy}\|$, from Equation (A.2), we know that $\frac{\partial u(\mathbf{W})}{\partial\mathbf{W}} = \mathbf{W}(\mathbf{x}\mathbf{y}^T + \mathbf{y}\mathbf{x}^T)$. According to Equation (A.3), the derivative of $v(\mathbf{W})$ is:

$$\begin{aligned}\frac{\partial v(\mathbf{W})}{\partial\mathbf{W}} &= \frac{\partial\|\mathbf{Wx}\|\|\mathbf{Wy}\|}{\partial\mathbf{W}} = \|\mathbf{Wy}\|\frac{\partial\|\mathbf{Wx}\|}{\partial\mathbf{W}} + \|\mathbf{Wx}\|\frac{\partial\|\mathbf{Wy}\|}{\partial\mathbf{W}} \\ &= \|\mathbf{Wy}\|\frac{\mathbf{W}\mathbf{x}\mathbf{x}^T}{\|\mathbf{Wx}\|} + \|\mathbf{Wx}\|\frac{\mathbf{W}\mathbf{y}\mathbf{y}^T}{\|\mathbf{Wy}\|}.\end{aligned}$$

Thus, the derivative of $s_\mathbf{W}(\mathbf{x},\mathbf{y})$ is:

$$\begin{aligned}\frac{\partial s_\mathbf{W}(\mathbf{x},\mathbf{y})}{\partial\mathbf{W}} &= \frac{\partial\frac{u(\mathbf{W})}{v(\mathbf{W})}}{\partial\mathbf{W}} = \frac{1}{v(\mathbf{W})}\frac{\partial u(\mathbf{W})}{\partial\mathbf{W}} - \frac{u(\mathbf{W})}{v(\mathbf{W})^2}\frac{\partial v(\mathbf{W})}{\partial\mathbf{W}} \\ &= \frac{1}{v(\mathbf{W})}[\frac{\partial u(\mathbf{W})}{\partial\mathbf{W}} - \frac{u(\mathbf{W})}{v(\mathbf{W})}\frac{\partial v(\mathbf{W})}{\partial\mathbf{W}}] \\ &= \frac{1}{\|\mathbf{Wx}\|\|\mathbf{Wy}\|}[\mathbf{W}(\mathbf{x}\mathbf{y}^T + \mathbf{y}\mathbf{x}^T) - \frac{(\mathbf{Wx})^T\mathbf{Wy}}{\|\mathbf{Wx}\|\|\mathbf{Wy}\|}(\|\mathbf{Wy}\|\frac{\mathbf{W}\mathbf{x}\mathbf{x}^T}{\|\mathbf{Wx}\|} + \|\mathbf{Wx}\|\frac{\mathbf{W}\mathbf{y}\mathbf{y}^T}{\|\mathbf{Wy}\|})] \\ &= \frac{1}{\|\mathbf{Wx}\|\|\mathbf{Wy}\|}[\mathbf{W}\mathbf{x}\mathbf{y}^T + \mathbf{W}\mathbf{y}\mathbf{x}^T - \frac{(\mathbf{Wx})^T\mathbf{Wy}}{\|\mathbf{Wx}\|^2}\mathbf{W}\mathbf{x}\mathbf{x}^T - \frac{(\mathbf{Wx})^T\mathbf{Wy}}{\|\mathbf{Wy}\|^2}\mathbf{W}\mathbf{y}\mathbf{y}^T] \\ &= \frac{-1}{\|\mathbf{Wx}\|\|\mathbf{Wy}\|}[(\frac{(\mathbf{Wx})^T\mathbf{Wy}}{\|\mathbf{Wx}\|^2}\mathbf{Wx} - \mathbf{Wy})\mathbf{x}^T + (\frac{(\mathbf{Wx})^T\mathbf{Wy}}{\|\mathbf{Wy}\|^2}\mathbf{Wy} - \mathbf{Wx})\mathbf{y}^T].\end{aligned}$$

QED.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

A.5 Derivative of the linear triangular loss

In Chapter 3, given a pair of vectors $(\mathbf{a}, \mathbf{b})$, the triangular loss is defined as:

$$J = \frac{1}{2}\|\mathbf{a}\|^2 + \frac{1}{2}\|\mathbf{b}\|^2 - r\|\mathbf{c}\| + r^2,$$

where r is a constant constraint on the vector length; $\mathbf{c} = \mathbf{a} + s\mathbf{b}$, representing the simplified Triangular Similarity in a positive triangle (s=1) or a negative triangle (s=-1). Especially, with a linear mapping function $f(\cdot)$: $\mathbf{a} = \mathbf{W}\mathbf{x}$ and $\mathbf{b} = \mathbf{W}\mathbf{y}$, prove that

$$\frac{\partial J}{\partial \mathbf{W}} = (\mathbf{a} - r\frac{\mathbf{c}}{\|\mathbf{c}\|})\mathbf{x}^T + (\mathbf{b} - sr\frac{\mathbf{c}}{\|\mathbf{c}\|})\mathbf{y}^T. \tag{A.5}$$

Proof: replacing **a**, **b** with **Wx**, **Wy**, respectively, the linear triangular loss is

$$\begin{aligned} J &= \frac{1}{2}\|\mathbf{W}\mathbf{x}\|^2 + \frac{1}{2}\|\mathbf{W}\mathbf{y}\|^2 - r\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\| + r^2 \\ &= \frac{1}{2}(\mathbf{W}\mathbf{x})^T\mathbf{W}\mathbf{x} + \frac{1}{2}(\mathbf{W}\mathbf{y})^T\mathbf{W}\mathbf{y} - r\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\| + r^2, \end{aligned}$$

According to Equation (A.2), we have

$$\begin{aligned} \frac{\partial(\mathbf{W}\mathbf{x})^T\mathbf{W}\mathbf{x}}{\partial \mathbf{W}} &= \mathbf{W}(\mathbf{x}\mathbf{x}^T + \mathbf{x}\mathbf{x}^T) = 2\mathbf{W}\mathbf{x}\mathbf{x}^T \\ \frac{\partial(\mathbf{W}\mathbf{y})^T\mathbf{W}\mathbf{y}}{\partial \mathbf{W}} &= \mathbf{W}(\mathbf{y}\mathbf{y}^T + \mathbf{y}\mathbf{y}^T) = 2\mathbf{W}\mathbf{y}\mathbf{y}^T. \end{aligned}$$

According to Equation (A.3), we have

$$\frac{\partial\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\|}{\partial \mathbf{W}} = \frac{\partial\|\mathbf{W}(\mathbf{x} + s\mathbf{y})\|}{\partial \mathbf{W}} = \frac{\mathbf{W}(\mathbf{x} + s\mathbf{y})(\mathbf{x} + s\mathbf{y})^T}{\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\|}.$$

Consequently, the derivative of the linear triangular loss is

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{W}} &= \frac{1}{2}\frac{\partial(\mathbf{W}\mathbf{x})^T\mathbf{W}\mathbf{x}}{\partial \mathbf{W}} + \frac{1}{2}\frac{\partial(\mathbf{W}\mathbf{y})^T\mathbf{W}\mathbf{y}}{\partial \mathbf{W}} - r\frac{\partial\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\|}{\partial \mathbf{W}} \\ &= \frac{1}{2}2\mathbf{W}\mathbf{x}\mathbf{x}^T + \frac{1}{2}2\mathbf{W}\mathbf{y}\mathbf{y}^T - r\frac{\mathbf{W}(\mathbf{x} + s\mathbf{y})(\mathbf{x} + s\mathbf{y})^T}{\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\|} \\ &= \mathbf{W}\mathbf{x}\mathbf{x}^T + \mathbf{W}\mathbf{y}\mathbf{y}^T - r\frac{\mathbf{W}(\mathbf{x} + s\mathbf{y})}{\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\|}\mathbf{x}^T - sr\frac{\mathbf{W}(\mathbf{x} + s\mathbf{y})}{\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\|}\mathbf{y}^T \\ &= (\mathbf{W}\mathbf{x} - r\frac{\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}}{\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\|})\mathbf{x}^T + (\mathbf{W}\mathbf{y} - sr\frac{\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}}{\|\mathbf{W}\mathbf{x} + s\mathbf{W}\mathbf{y}\|})\mathbf{y}^T. \end{aligned}$$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Moreover, substituting $\mathbf{Wx}$, $\mathbf{Wy}$ with $\mathbf{a}$, $\mathbf{b}$, respectively, the above equation can be rewritten as:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{W}} &= (\mathbf{Wx} - r\frac{\mathbf{Wx}+s\mathbf{Wy}}{\|\mathbf{Wx}+s\mathbf{Wy}\|})\mathbf{x}^T + (\mathbf{Wy} - sr\frac{\mathbf{Wx}+s\mathbf{Wy}}{\|\mathbf{Wx}+s\mathbf{Wy}\|})\mathbf{y}^T \\ &= (\mathbf{a} - r\frac{\mathbf{a}+s\mathbf{b}}{\|\mathbf{a}+s\mathbf{b}\|})\mathbf{x}^T + (\mathbf{b} - sr\frac{\mathbf{a}+s\mathbf{b}}{\|\mathbf{a}+s\mathbf{b}\|})\mathbf{y}^T \\ &= (\mathbf{a} - r\frac{\mathbf{c}}{\|\mathbf{c}\|})\mathbf{x}^T + (\mathbf{b} - sr\frac{\mathbf{c}}{\|\mathbf{c}\|})\mathbf{y}^T.\end{aligned}$$

QED.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Appendix B

Learning on Similar Pairs Only

B.1 Introduction

This appendix is a short version of our paper "Logistic Similarity Metric Learning for Face Verification" [183] published in the proceedings of 40th International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2015.

In this work, based on the Cosine Similarity Metric Learning (CSML), we develop a new method called Logistic Similarity Metric Learning (LSML) for pairwise face verification. Specifically, we introduce a parameter K to shift the similarity decision boundary, formulate the cost using the logistic loss function, and produce a probability estimation of a pair of faces being similar. We performed extensive experiments on the LFW-a dataset [75] under restricted configuration with label-free outside data. The proposed method achieved competitive performance over the state-of-the-art linear methods. Moreover, we propose a faster way to achieve the same goal: *learning on similar pairs only*. Learning on similar pairs has one thing in common with shifting the boundary that both of them make the similar training pairs contribute more to the gradient than the dissimilar training pairs. And the latter has fewer parameters to tune and requires less data for training. However, this should be under the linear constraint to prevent the probable large over-fitting problem in training.

B.2 Cosine Similarity Metric Learning

In the task of face verification, two face images of the same person are called a similar pair; otherwise, two face images of different persons are called a dissimilar pair or a different pair. By representing the face images as vectors, the verification of faces becomes a problem of measuring similarity between vectors.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Before introducing the CSML method, we present some important notations: a triplet $(\mathbf{x}_i, \mathbf{y}_i, s_i)$ represents a pair of instances, where $\mathbf{x}_i$ and $\mathbf{y}_i$ are two vectors, and $s_i = 1$ (respectively -1) means that the two vectors are similar (respectively dissimilar). A linear metric learning method defines a linear transformation $f(\mathbf{z}, \mathbf{A}) = \mathbf{A}\mathbf{z}$ on the raw feature vectors and produces another triplet $(\mathbf{a}_i, \mathbf{b}_i, s_i)$, where $\mathbf{a}_i = f(\mathbf{x}_i, \mathbf{A}) = \mathbf{A}\mathbf{x}_i$ and $\mathbf{b}_i = f(\mathbf{y}_i, \mathbf{A}) = \mathbf{A}\mathbf{y}_i$. The objective of CSML is employing this transformation to make similar vectors closer and separate dissimilar vectors: an optimal matrix $\mathbf{A}$ makes $cos(\mathbf{a}_i, \mathbf{b}_i) = 1$ for a pre-defined similar pair $(\mathbf{x}_i, \mathbf{y}_i)$ while making $cos(\mathbf{a}_i, \mathbf{b}_i) = -1$ for a dissimilar pair, where the cosine similarity $cos(\mathbf{a}_i, \mathbf{b}_i)$ is:

$$cos(\mathbf{a}_i, \mathbf{b}_i) = \frac{\mathbf{a}_i^T \mathbf{b}_i}{\|\mathbf{a}_i\| \|\mathbf{b}_i\|}. \tag{B.1}$$

The cost function of CSML is [125]:

$$J_{CSML} = \frac{1}{n} \sum_{i=1}^{n} -s_i cos(\mathbf{a}_i, \mathbf{b}_i) + \frac{\lambda}{2} \|\mathbf{A} - \mathbf{A}_0\|^2, \tag{B.2}$$

with gradient function:

$$\begin{aligned} \frac{\partial J_{CSML}}{\partial \mathbf{A}} &= \frac{1}{n} \sum_{i=1}^{n} \frac{\partial -s_i cos(\mathbf{a}_i, \mathbf{b}_i)}{\partial \mathbf{A}} + \lambda(\mathbf{A} - \mathbf{A}_0) \\ &= \frac{1}{n} \sum_{i=1}^{n} \frac{s_i}{\|\mathbf{a}_i\| \|\mathbf{b}_i\|} [(\frac{\mathbf{a}_i^T \mathbf{b}_i}{\|\mathbf{a}_i\|^2} \mathbf{a}_i - \mathbf{b}_i) \mathbf{x}_i^T + (\frac{\mathbf{a}_i^T \mathbf{b}_i}{\|\mathbf{b}_i\|^2} \mathbf{b}_i - \mathbf{a}_i) \mathbf{y}_i^T] \quad + \lambda(\mathbf{A} - \mathbf{A}_0), \end{aligned} \tag{B.3}$$

where n is the number of all similar and dissimilar pairs from the training data, λ is the regularization parameter, and $\mathbf{A}_0$ is any matrix that we want $\mathbf{A}$ to be regularized with: we set $\mathbf{A}$ to be $\mathbf{A}_0$ before optimizing the cost; hence during the optimization, the larger the parameter λ is, the closer $\mathbf{A}$ is to $\mathbf{A}_0$. Usually, we specify $\mathbf{A}_0$ as the identity matrix $\mathbf{I}$. More discussion on CSML can be found in Chapter 2 and the proof of the CSML gradient is in Appendix A.

B.3 Logistic Similarity Metric Learning

Minimizing the CSML cost function (Equation (B.2)) implies making $cos(\mathbf{a}_i, \mathbf{b}_i) > 0$ for a similar pair and making $cos(\mathbf{a}_i, \mathbf{b}_i) < 0$ for a dissimilar pair at the same time. In other words, CSML sets 0 as the decision boundary for this binary decision problem. However, in a limited space which contains a large quantity of classes, it's impossible that all the dissimilar pairs have negative cosine similarity values. For example, when there are more than 4 classes in the 2-dimensional space, we can find at least one pair of classes with the angle less than 90° (i.e. cosine similarity value larger than 0). Thus the assumption of setting $cos(\mathbf{a}_i, \mathbf{b}_i) < 0$

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

for all the dissimilar pairs is only feasible if the dimension of the output feature space is large enough. However, for a large number of classes, this high-dimensional output space may lead to many local minima and over-fitting.

Therefore, we introduce a positive constant K to shift the decision boundary. Moreover, following [60, 69] that employed the logistic loss function in distance metric learning to create a decision gap between the similar pairs and dissimilar pairs [97], we incorporate the logistic loss function with the cosine similarity cost function as:

$$J = \frac{1}{n}\sum_{i=1}^{n} ln(1+exp(-\frac{s_i(cos(\mathbf{a}_i,\mathbf{b}_i)-K)}{T})) + \frac{\lambda}{2}\|\mathbf{A}-\mathbf{A}_0\|^2, \tag{B.4}$$

where the constant T is the sharpness parameter which is set to 0.1 in our experiments. The corresponding gradient function is:

$$\frac{\partial J}{\partial \mathbf{A}} = \frac{1}{nT}\sum_{i=1}^{n}(1-\frac{1}{h_i})\frac{\partial -s_i cos(\mathbf{a}_i,\mathbf{b}_i)}{\partial \mathbf{A}} + \lambda(\mathbf{A}-\mathbf{A}_0), \tag{B.5}$$

where $h_i = 1+exp(-\frac{s_i(cos(\mathbf{a}_i,\mathbf{b}_i)-K)}{T})$ and the partial derivative $\frac{\partial -s_i cos(\mathbf{a}_i,\mathbf{b}_i)}{\partial \mathbf{A}}$ is the same as in Equation (B.3).

Now we relate the LSML method to the task of face verification. At first, we collect labeled similar/dissimilar pairs of vectors which represent pairs of face images, i.e. $(\mathbf{x}_i, \mathbf{y}_i, s_i)$, as training data. By initializing the linear transformation matrix $\mathbf{A}$ with the identity matrix, we can calculate the initial cost and gradient using Equations (B.4) and (B.5). After that, we employ the advanced L-BFGS [108] optimization algorithm to automatically update the transformation matrix $\mathbf{A}$ until the overall cost gets convergency. Specifically, we used a MATLAB implementation of L-BFGS provided by Mark Schmidt [144]. Finally, we will get an optimal solution $\mathbf{A}_\star$ which produces a local minimal cost on the current training data, and we use $\mathbf{A}_\star$ to transform all $(\mathbf{x}_i, \mathbf{y}_i)$ to the outputs $(\mathbf{a}_i, \mathbf{b}_i)$, remind that $\mathbf{a}_i = f(\mathbf{x}_i, \mathbf{A}_\star) = \mathbf{A}_\star \mathbf{x}_i$ and $\mathbf{b}_i = f(\mathbf{y}_i, \mathbf{A}_\star) = \mathbf{A}_\star \mathbf{y}_i$. Formally, we call $\mathbf{A}_\star$ the optimal metric that have been learned.

Naturally, we model the probability p_i that an output pair $(\mathbf{a}_i, \mathbf{b}_i)$ is similar by the standard logistic function, i.e. the sigmoid function:

$$p_i = \frac{1}{1+exp(-\frac{cos(\mathbf{a}_i,\mathbf{b}_i)-K}{T})}. \tag{B.6}$$

If p_i exceeds a pre-defined threshold γ, we label the pair $(\mathbf{a}_i, \mathbf{b}_i)$ as similar, otherwise we assign it as dissimilar. The parameter γ is tuned on a validation set, and then the best parameter is selected for test evaluation.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Table B.1 Face verification accuracy (±standard error of the mean) on LFW-a under restricted configuration with label-free outside data. Dimension of the whitened feature vectors is 300. Comparing the performance, LSML>WCCN>CSML>Baseline.

Method	LBP		OCLBP		SIFT		Gabor	
	original	square root	original	square root	original	square root	original	square root
Baseline	77.17±0.49	79.73±0.38	80.43±0.25	81.55±0.44	76.88±0.42	77.52±0.49	75.28±0.45	77.25±0.32
CSML	79.47±0.55	82.92±0.47	82.62±0.55	84.67±0.58	81.88±0.47	82.88±0.37	78.52±0.59	80.38±0.51
WCCN	80.40±0.39	84.23±0.33	83.75±0.51	86.83±0.37	82.72±0.39	84.17±0.25	78.68±0.62	81.52±0.65
LSML	**83.58±0.66**	**85.17±0.50**	**85.48±0.69**	***87.55±0.49***	**84.67±0.46**	**85.77±0.37**	**80.98±0.70**	**83.28±0.43**

Table B.2 Face verification accuracy (±standard error of the mean) on LFW-a under restricted configuration with label-free outside data. Dimension of the whitened feature vectors is 300. CSML-sim and LSML-sim learns on only the similar pairs from the training set. Comparing the performance, LSML=CSML-sim=LSML-sim.

Method	LBP		OCLBP		SIFT		Gabor	
	original	square root	original	square root	original	square root	original	square root
LSML	83.58±0.66	85.17±0.50	85.48±0.69	**87.55±0.49**	84.67±0.46	85.77±0.37	80.98±0.70	83.28±0.43
CSML-sim	83.27±0.73	85.32±0.56	85.43±0.70	**87.35±0.47**	85.07±0.47	85.98±0.44	81.28±0.59	83.55±0.50
LSML-sim	83.18±0.78	85.47±0.62	85.35±0.68	**87.35±0.48**	85.00±0.46	85.78±0.33	81.22±0.44	83.55±0.45

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

B.4 Experiment and Analysis

B.4.1 Experimental Setting

Actually, the experimental setting here is very similar with that in Chapter 4 for pairwise face verification on the LFW-a dataset. This dataset contains most kinds of facial variations in face pose, facial expression, illumination and partial occlusions, etc, and it has been the most popular benchmark for face verification. All of our experiments are performed under *the LFW restricted configuration with label-free outside data*: only the provided 6000 pairs of data are used for training and evaluation.

We only use View 2 subset of LFW for experimental performance evaluation. There are 5749 people in the dataset which are divided into mutually exclusive 10 folds: the person in any fold would not appear in the other fold. The total number of images in LFW is 13233, however, the number of images for each person varies from 1 to 530.

We perform a 10-fold cross-validation on the aligned LFW-a data [160]: in each experiment, we select 8 out of the 10 folds as the training set, the other 2 folds are used for validation and testing respectively. For example, the first experiment uses subsets (1,2,3,4,5,6,7,8) for training, subset 9 for validation and subset 10 for testing; the second experiment uses (2,3,4,5,6,7,8,9) for training, subset 10 for validation and subset 1 for testing. After 10 repetitions, we report the mean accuracy ($\pm$standard error of the mean).

Feature vectors

We use four face descriptors to represent the face images: Gabor wavelets [38], LBP [1], SIFT [81] and OCLBP [8]. For Gabor and LBP, we used exactly the same setting as in [125], dimension of Gabor and LBP is 4,800 and 7,080, respectively. For SIFT, we directly used the 3,456-d feature data provided by [60]. For OCLBP, the high dimensional variant of LBP, we used the same setting as in [184], dimension of the OCLBP descriptor is 46,846. Compared with LBP using non-overlapping shifting window, OCLBP allows overlapping to adjacent windows, therefore OCLBP is with much higher dimension than LBP and describes more detailed facial texture. Additionally, square-roots of all the descriptors are also evaluated. Moreover, following [26], we reduce the dimension of all the raw feature vectors to 300 by whitened PCA.

B.4.2 Results and Analysis

We perform experiments with CSML [125] and LSML for face verification on LFW-a under *the LFW restricted configuration with label-free outside data*. Especially, we implemented

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

the state-of-the-art method WCCN [8] as a comparison. In the experiments, we have three parameters to tune: the decision threshold γ, the regularization term λ and the shifting parameter K (only for LSML). The tuning range of γ was from 0 to 1 with a step size of 0.001 for all the experiments. The tuning range of λ was from 2×10^{-3} to 10×10^{-3} with a step size of 10^{-3} for CSML. For LSML, the tuning range of λ was from 15×10^{-3} to 20×10^{-3} with a step size of 10^{-3} and the tuning range of K was from 0 to 0.8 with a step size of 0.1.

Comparison with the state-of-the-art

To set a baseline, we first perform evaluation on the 300-d whitened feature vectors, i.e. setting the transformation matrix $\mathbf{A}$ as the identity matrix. Results on different features are listed in the first row of Table B.1.

Comparing CSML with the baseline, we can see significant performance gain for all the features. For instance, on the square-rooted OCLBP, CSML obtains a performance gain from 81.55% to 84.67% over the baseline. WCCN [8], further increases the accuracy to 86.83% on the same feature. And the proposed LSML method performs the best on all the features (the fourth row in Table B.1). For example, LSML achieves 87.55% on the square-rooted OCLBP. In summary, comparing the performance of the four methods, LSML>WCCN>CSML>Baseline.

Effectiveness of the shifting parameter K

Figure B.1 shows the accuracy-versus-K curve of the proposed LSML method using the square-rooted OCLBP. We tune the shifting parameter from 0 to 0.8, and record the mean accuracy and its standard error on the 10-fold experiments. The regularization parameter λ is kept as 17×10^{-3}. We can see that the curve rises rapidly when the decision boundary is shifted from 0, and arrives the peak 87.55% at $K=0.5$.

This curve illustrates that shifting the decision boundary towards the positive side can adjust the cost from the similar training pairs and the dissimilar training pairs, which leads to considerable improvement of verification performance.

B.4.3 Learning on Similar Pairs Only

From another perspective on the logistic loss function (Equations (B.4) and (B.5)), shifting the decision boundary also means making similar pairs contribute more to the gradient than the dissimilar pairs. To verify this, we sum up the gradient coefficient $(1-\frac{1}{h_i})$ in Equation (B.5) for similar pairs and dissimilar pairs, respectively. Generally, the coefficients are all positive numbers in the range $[0,1]$ and larger coefficients imply more contribution to the gradient. In

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

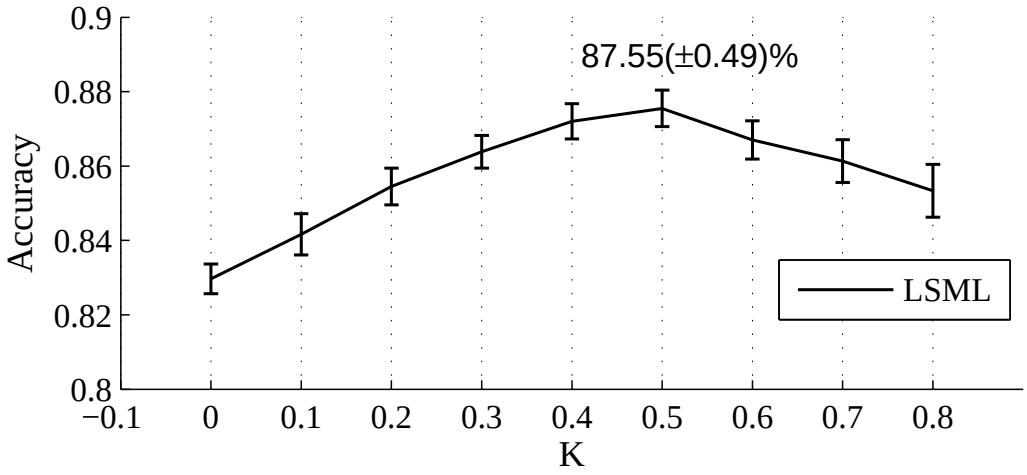


Fig. B.1 Accuracy-versus-K curve for the proposed LSML method using the square root of OCLBP. The regularization parameter $\lambda = 17 \times 10^{-3}$. The peak $87.55 \pm 0.49\%$ is at $K = 0.5$.

the example of Figure B.1, when $K = 0$, the sum of the gradient coefficients for the similar training pairs is 523.0 and that for the dissimilar pairs is 1200.1; when $K = 0.5$, we get 2186.0 and 19.7 correspondingly. This means that with the decision boundary shifted from 0 to 0.5, the contribution of the similar pairs to the gradient has been increased dramatically.

Thus we propose an argument that *under the linear constraint, learning on similar pairs only can find a proper decision boundary automatically*. Coincidentally, the WCCN computation is only based on pairs from the same class [8]. Concretely, we perform learning only on the similar pairs from the training set for CSML and LSML, namely CSML-sim and LSML-sim: the cost and gradient functions are kept the same but the dissimilar training pairs are abandoned. For LSML-sim, we keep the shifting parameter K to be 0 and the sharpness parameter T to be 1. The results are reported in the last two rows of Table B.2. We can see that the two methods achieve almost the same performance with the standard LSML method over all the features. For example, on the square-rooted SIFT descriptor, LSML, CSML-sim and LSML-sim obtain 85.77%, 85.98% and 85.78%, respectively.

Compared with the LSML that shifts the boundary by tuning a parameter K and trains on both similar and dissimilar pairs, fewer parameters and less training data lead to faster training for CSML-sim and LSML-sim. However, it is worth noting that it should be under the linear constraint, otherwise training on similar pairs only is prone to a large over-fitting problem.

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés

UNIVERSITÉ DE LYON

INSA

FOLIO ADMINISTRATIF

THESE DE L'UNIVERSITE DE LYON OPEREE AU SEIN DE L'INSA LYON

NOM : ZHENG
(avec précision du nom de jeune fille, le cas échéant)

DATE de SOUTENANCE : 10/05/2016

Prénoms : Lilei

TITRE : Triangular Similarity Metric Learning: a Siamese Architecture Approach

NATURE : Doctorat

Numéro d'ordre : 2016LYSEI045

Ecole doctorale : INFORMATIQUE ET MATHEMATIQUES (ED512)

Spécialité : Informatique et applications

RESUME :

Dans de nombreux problèmes d'apprentissage automatique et de reconnaissance des formes, il y a toujours un besoin de fonctions métriques appropriées pour mesurer la distance ou la similarité entre des données. La fonction métrique est une fonction qui définit une distance ou une similarité entre chaque paire d'éléments d'un ensemble de données. Dans cette thèse, nous proposons une nouvelle methode, Triangular Similarity Metric Learning (TSML), pour spécifier une fonction métrique de données automatiquement.

Le système TSML proposée repose une architecture Siamese qui se compose de deux sous-systèmes identiques partageant le même ensemble de paramètres. Chaque sous-système traite un seul échantillon de données et donc le système entier reçoit une paire de données en entrée. Le système TSML comprend une fonction de coût qui définit la relation entre chaque paire de données et une fonction de projection permettant l'apprentissage des formes de haut Niveau.

Pour la fonction de coût, nous proposons d'abord la similarité triangulaire (Triangular Similarity), une nouvelle similarité métrique qui équivaut à la similarité cosinus. Sur la base d'une version simplifiée de la similarité triangulaire, nous proposons la fonction triangulaire (the triangular loss) afin d'effectuer l'apprentissage de métrique, en augmentant la similarité entre deux vecteurs dans la même classe et en diminuant la similarité entre deux vecteurs de classes différentes. Par rapport aux autres distances ou similarités, la fonction triangulaire et sa fonction gradient nous offrent naturellement une interprétation géométrique intuitive et intéressante qui explicite l'objectif d'apprentissage de métrique.

En ce qui concerne la fonction de projection, nous présentons trois fonctions différentes: une projection linéaire qui est réalisée par une matrice simple, une projection non-linéaire qui est réalisée par Multi-layer Perceptrons (MLP) et une projection non-linéaire profonde qui est réalisée par Convolutional Neural Networks (CNN). Avec ces fonctions de projection, nous proposons trois systèmes de TSML pour plusieurs applications: la vérification par paires, l'identification d'objet, la réduction de la dimensionnalité et la visualisation de données. Pour chaque application, nous présentons des expérimentations détaillées sur des ensembles de données de référence afin de démontrer l'efficacité de notre systèmes de TSML.

MOTS-CLÉS : architecture Siamese; reconnaissance des formes; apprentissage métriques; Multi-layer Perceptrons; Convolutional Neural Networks

Laboratoire (s) de recherche : Laboratoire d'InfoRmatique en Image et Systèmes d'information (LIRIS)

Directeur de thèse: IDRISSI, Khalid

Président de jury : DORIZZI, Bernadette

Composition du jury :

DORIZZI, Bernadette	Prof.	Télécom SudParis	Présidente
MARCHAND-MAILLET, Stéphane	Prof.	University of Geneva	Rapporteur
THOME, Nicolas	MCF, HDR	Université Pierre et Marie Curie	Rapporteur
PUECH, William	Prof.	Université de Montpellier	Examinateur
BASKURT, Atilla	Prof.	INSA-LYON	Co-Directeur de thèse
IDRISSI, Khalid	MCF, HDR	INSA-LYON	Directeur de thèse
GARCIA, Christophe	Prof.	INSA-LYON	Examinateur

Cette thèse est accessible à l'adresse : http://theses.insa-lyon.fr/publication/2016LYSEI045/these.pdf
© [L. Zheng], [2016], INSA Lyon, tous droits réservés