

การปรับเฉพาะสำหรับการจำแนกประเภท



นายภาสกร ตั้งชนะชัยอนันต์

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2553

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

CUSTOMIZATION FOR CLASSIFICATION



Mr. Pasakorn Tangchanachaianan

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

A Dissertation Submitted in Partial Fulfillment of the Requirements  
for the Degree of Doctor of Philosophy Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

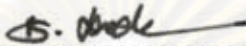
Academic Year 2010

Copyright of Chulalongkorn University

Thesis Title        CUSTOMIZATION FOR CLASSIFICATION  
By                     Mr. Pasakorn Tangchanachaianan  
Field of Study       Computer Engineering  
Thesis Advisor      Professor Boonserm Kijirikul, Ph.D.

---

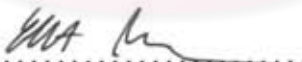
Accepted by the Faculty of Engineering, Chulalongkorn University in  
Partial Fulfillment of the Requirements for the Doctoral Degree

  
..... Dean of the Faculty of Engineering  
(Associate Professor Boonsom Lerthirunwong, Dr.Ing.)

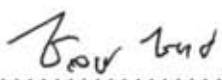
THESIS COMMITTEE

  
..... Chairman  
(Professor Prabhas Chongstitvattana), Ph.D.)

  
..... Thesis Advisor  
(Professor Boonserm Kijirikul, Ph.D.)

  
..... Examiner  
(Assistant Professor Chotirat Ratanamahatana, Ph.D.)

  
..... Examiner  
(Assistant Professor Sukree Sinthupinyo, Ph.D.)

  
..... Examiner  
(Assistant Professor Cholwich Nattee, Ph.D.)

ภาสกร ตั้งชนะชัยอนันต์ : การปรับเฉพาะสำหรับการจำแนกประเภท (CUSTOMIZATION FOR CLASSIFICATION) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ศ.ดร. บุญเสริม กิจศิริกุล, 76 หน้า.

การปรับเฉพาะเป็นกระบวนการทางการเรียนรู้ของเครื่อง ในการแก้ไข โมเดลที่เรียนรู้มาแล้ว เพื่อให้สามารถนำมาใช้กับข้อมูลเพิ่มเติมได้ดีขึ้น สิ่งที่แตกต่างกันจากการสร้าง โมเดลใหม่ขึ้นมาจากข้อมูลเพิ่มเติมเลยก็คือการปรับเฉพาะนั้นจะอาศัยสมมติฐานที่ว่า การรู้ โมเดลที่เรียนรู้มาแล้วนั้นจะเป็นข้อมูลที่มีประโยชน์ซึ่งจะช่วยให้การปรับปรุงผลในการเรียนรู้ของเครื่องขึ้นได้ อย่างไรก็ตาม ข้อจำกัดทั่วไปของอัลกอริทึมในการเรียนรู้เพิ่มเติมก็คืออัลกอริทึมเหล่านั้นมักจะขึ้นอยู่กับ โมเดลที่ใช้ นั่นก็ส่งผลให้จะต้องมีอัลกอริทึมใหม่สำหรับ โมเดลแต่ละประเภทและเกิดมีปัญหาคือในกรณีที่ไม่สามารถหาอัลกอริทึมที่เหมาะสมมาใช้ได้เนื่องจากไม่มีอัลกอริทึมสำหรับ โมเดลนั้นหรือไม่ทราบ โมเดลที่กำลังใช้

ดังนั้นเราจึงเสนอทางเลือกที่จะทำการฝึกเพิ่มเติม โดยใช้อัลกอริทึมที่ขึ้นอยู่กับประเภทของงาน เนื่องจากข้อจำกัดขั้นต่ำสุดของการที่ โมเดลนั้นจะถูกนำไปใช้ประโยชน์ได้ก็คือต้องรู้ว่า โมเดลนั้นมีไว้สำหรับงานอะไร จากงานในด้านการเรียนรู้ของเครื่องทั้งหมด เราได้นำเสนออัลกอริทึมสำหรับการจำแนกประเภท และการลดมิติ และได้ทำการทดสอบอัลกอริทึมเหล่านั้น โดยใช้ชุดข้อมูลที่สร้างจากข้อมูลที่มีอยู่จริง ผลการทดสอบแสดงให้เห็นถึงประสิทธิผลของวิธีการที่นำเสนอ

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

ภาควิชา ... วิศวกรรมคอมพิวเตอร์ ... ลายมือชื่อนิสิต ...  
สาขาวิชา ... วิศวกรรมคอมพิวเตอร์ ... ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก ...  
ปีการศึกษา ..... 2553 .....


##4971822121 : MAJOR COMPUTER ENGINEERING


KEYWORDS : CUSTOMIZATION / CLASSIFICATION / DIMENSIONALITY  
REDUCTION

PASAKORN TANGCHANACHAIANAN : CUSTOMIZATION FOR  
CLASSIFICATION. ADVISOR : PROF. BOONSERM KIJSIRIKUL, Ph.D.,  
76 pp.

Customization is a machine learning approach in modifying a learned model to be better adapted with additional data. In contrast to training a new model from the additional data, customization relies on the hypothesis that knowing the learned model will serve as useful information which could improve the result of machine learning. However, a common limitation for customization algorithms is that they are specific on the type of model. This results in requiring new algorithms for each type of models and problem could arise if the user is unable to determine appropriate algorithms due to the lack of algorithms for that model or the difficulty in determining the type of models.

Therefore, we propose an alternative of performing customization with the algorithm that is based on the task since the minimum requirement for a model to be useful is to know the task it is for. From among all possible tasks, we present customization frameworks and algorithms for classification and dimensionality reduction and perform experiments on them using real-world dataset. This experimental results show the effectiveness of our proposed methods.

Department : . Computer Engineering . Student's Signature  .....

Field of Study : . Computer Engineering . Advisor's Signature  .....

Academic Year : .....2010.....

## Acknowledgments

First of all, I would like to express my gratitude to my advisor, for being patient and understanding and having his faith on a student like me, and also trying to support me in everyway he can. I am grateful to Prof. Prabhas Chongstitvattana for his concern due to my abnormal situation and as well as other of my thesis committee members who have to give away their precious time and provide me with their comments.

I would like to thank Ratthachat Chatpatanasiri, who provides a lot of advice concerning my works, Teesid Korsrilabutr, the owner of L<sup>A</sup>T<sub>E</sub>X template which I modify into my dissertation, Tanasanee Phienthrakul, for keeping me company even when there is almost no other appearing member in our lab, and also for other MIND lab members for keeping me company and giving me precious experiences. Also, I would like to extend my thanks to other graduate students in our departments whom either keep me company and even provide me with supports that come in all forms.

And most importantly, I wish to thank the Thailand Research Fund for giving me scholarship, which greatly assists me in financial issue.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

# Contents

	Page
<b>Abstract (Thai)</b> . . . . .	iv
<b>Abstract (English)</b> . . . . .	v
<b>Acknowledgments</b> . . . . .	vi
<b>Contents</b> . . . . .	vii
<b>List of Tables</b> . . . . .	x
<b>List of Figures</b> . . . . .	xii
<b>Chapter</b>	
<b>1 Introduction</b> . . . . .	<b>1</b>
1.1 Objectives . . . . .	3
1.2 Scope . . . . .	3
1.3 Procedure . . . . .	4
1.4 Contributions . . . . .	4
1.5 Organization of the Thesis . . . . .	4
<b>2 Background</b> . . . . .	<b>5</b>
2.1 Conventions . . . . .	5
2.2 Customization . . . . .	5
2.3 Classification . . . . .	6
2.3.1 Weighted Nearest Neighbor . . . . .	6
2.3.2 Max Wins . . . . .	6
2.3.3 Adaptive Directed Acyclic Graph . . . . .	6
2.3.4 Optical Character Recognition . . . . .	7
2.3.5 Handwriting Recognition . . . . .	8
2.4 Dimensionality Reduction . . . . .	8
2.4.1 Linear Dimensionality Reduction . . . . .	10
2.4.1.1 Principal Component Analysis . . . . .	10
2.4.1.2 Linear Discriminant Analysis . . . . .	11
2.4.2 KPCA Methods of Mahalanobis Distance Learning for Nearest Neighbor . . . . .	12

	Page
2.5 Halton Sequence . . . . .	12
<b>3 Task-based Customization . . . . .</b>	<b>15</b>
3.1 Concept of Task-Based Customization . . . . .	15
3.2 Advantages and Drawbacks . . . . .	16
<b>4 Customization for Classification . . . . .</b>	<b>17</b>
4.1 Customization for Classification by Transforming Input Space . .	17
4.1.1 Generating sequence of transformation for probabilistic classifiers . . . . .	17
4.1.1.1 Algorithm . . . . .	17
4.1.1.2 Numerical Results . . . . .	20
4.1.2 Applying with other classifiers . . . . .	21
4.1.2.1 Algorithm . . . . .	21
4.1.2.2 Numerical Results . . . . .	23
4.1.3 Overall Framework . . . . .	26
4.2 Customization for Classification by Patching . . . . .	27
4.2.1 Algorithms . . . . .	27
4.2.2 Numerical Results . . . . .	32
4.3 Customization for Classification with Nonlinear Dimension- ality Reduction . . . . .	35
4.3.1 Numerical Results . . . . .	36
<b>5 Customization for Dimensionality Reduction . . . . .</b>	<b>44</b>
5.1 Framework of Customization for Dimensionality Reduction . . .	44
5.2 Combining Results from Linear Dimensionality Reduction . . . .	45
5.2.1 Algorithm . . . . .	45
5.2.2 Theoretical Results . . . . .	51
5.3 Numerical Results . . . . .	67
<b>6 Conclusion and Future Work . . . . .</b>	<b>71</b>
6.1 Conclusion . . . . .	71



	Page
6.2 Future Work . . . . .	71
<b>References . . . . .</b>	<b>73</b>
<b>Biography . . . . .</b>	<b>76</b>



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## List of Tables

	Page
2.1 First few terms in Halton sequence for two dimensional space. . . . .	13
4.1 Accuracy comparison between customization data, mapped customization data and mapped customization data using the proposed output function. . . . .	21
4.2 Accuracy from the generated classifier when using the algorithm that is applicable to non-probabilistic classifier. . . . .	23
4.3 Accuracy of each transformed subclassifier. . . . .	24
4.4 Accuracy of the combined classifier created from transformed subclassifiers. . . . .	25
4.5 Accuracy of each transformed subclassifier when using generated subclassifiers to determine the transformation. . . . .	25
4.6 Accuracy of the combined classifier created from transformed subclassifiers when using generated subclassifiers to determine the transformation. . . . .	26
4.7 Accuracy of patched subclassifiers from <i>pendigits</i> data. . . . .	33
4.8 Accuracy of patched subclassifiers from <i>optdigits</i> data. . . . .	33
4.9 Accuracy of combined classifier from patched subclassifier. . . . .	34
4.10 Accuracy of patched combined classifier. . . . .	35
4.11 Result from customization using linear dimensionality reduction algorithm. . . . .	37
4.12 Result from customization using the KMMLN algorithm. . . . .	37
5.1 Detail of all datasets used in the experiments of combining results from linear dimensionality reduction. The numbers of attributes shown are the numbers of attributes used in dimensionality reduction, omitting some useless attributes like index or specific name. . . .	67
5.2 Result from using the proposed frameworks with linear dimensionality reduction. Numbers whose values are minimal in their rows are typeset bold. The second column shows the reduced dimension. All of the equal values are different in higher precision. . . .	68

## List of Tables (cont.)

	Page
5.3 Classification accuracy after performing customization upon dimensionality reduction. (G) and (S) are the results from original data and customization data respectively, and (N) is the one created from both models with our algorithm. . . . .	70



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## List of Figures

	Page
2.1 Visualization for Adaptive Directed Acyclic Graph. . . . .	8
2.2 300 sampled data in two dimensions ( $[0, 1]^2$ ) and dispersion value measured with box space. The box shows the area which results in dispersion value. . . . .	14
4.1 Visualization of the problem in determining objective for mapping. The class of data are shown by depicting each of them as square and circle. The solid line is a likely decision boundary resulted from summing the objective value calculated from each data and the dashed line is the most suitable decision boundary. . . . .	19
4.2 Plotting of output function $s(x) = 1 - ((1 - p(x))^2)$ . . . . .	20
4.3 Visualization of results from transforming generated probabilistic classifiers. . . . .	38
4.4 Framework of customization for classification by Transforming Input Space. . . . .	39
4.5 Framework of customization for classification with nonlinear dimensionality reduction. . . . .	40
4.6 New classifier from the framework for customization of classification with nonlinear dimensionality reduction. . . . .	41
4.7 Visualization of generated dataset. . . . .	42
4.8 Embedding the data into a higher dimension. . . . .	43
5.1 Framework for customization for dimensionality reduction. . . . .	46
5.2 The new dimension reducer as the result of using the framework. . . .	47

# CHAPTER I

## INTRODUCTION

Classification or supervised learning is the process in identifying the class or group to which a data belongs, based on the basis of a training set of data whose classes are given. For example, in the task where we want to identify the type of creatures from their DNA sequence, each class is a possible type of creatures while the information to be used for prediction is the DNA sequence. The most logical way is to use our knowledge about the DNA sequence to determine to which kind of animals this DNA sequence belongs. However, the task of machine learning is not only to apply the knowledge that we have learned, but also to extract these knowledge from the data as well. The algorithm for classification will base its decision upon the dataset of DNA sequences with given classes of the creatures, interpret these information in some way and provide an output as a process of decision making called classifier.

The problem we are interested in is called customization, or in some research work is called user-adaptation. A simple description of customization is that there will be additional information about the problem in the form that we will be given an existing machine learning model that has already been trained to be suitable to the dataset in question up to some degree. For general usage, this model will be suitable to a dataset generated from a distribution with high similarity to the distribution of the dataset we are interested in. The main problem of customization is how we should use this model to boost or improve the result in the task of machine learning, and generally, the task would be inversely viewed as using the customization data to improve the model instead of using the model to improve the result from just using the customization data on the task. An obvious and practical real-world example of the problem in customization is Handwriting Recognition.

The situation in handwriting recognition is that we want to predict characters a person has written, with the information about how the writing tool has been stroked. An obvious characteristic of this problem is that every per-

son has different handwriting and there might not be any classifier that can correctly classify handwriting of everyone due to different writing habit. An obvious example would be for the characters with high similarity such as letter "O" and number "0" or letter "S" and number "5"; different people may write them with the same kind of strokes and hence given a data about the strokes used in writing a character, the result from classification of the character will depend on the writer as well. As a result, we will have to use handwriting data of each person to make a perfect handwriting classifier for himself. As in other tasks of machine learning, the higher the number of data, the higher probability that the resulted model will be good. Since handwriting of each person differs from each other, each of the input data must be generated by that specific person. Suppose that we want to have 10 samples of each character from a set of numbers 0-9 and alphabets A-Z. The total number of characters that the person in question must write will be  $(10 + 26) * 10 = 360$ , and that just might not be enough to create a good classifier for characters with similar writing pattern. The reason that handwriting recognition is a perfect task for customization is also because even though handwriting of everyone may be different from each other, but the same character of each person must be based on the same standard that it should be able for a man to read and recognize the character using his prior knowledge in the language. By using the language standard, we can create a classifier that fully employs additional information to perform customization.

Another advantage of customization is time efficiency. As the model does not have to be learned from scratch, performing customization is more efficient in time and it does not require the data previously used in making the model. One may view customization as incremental training which is performed by a new dataset at each time, where the model is adapted to the dataset while retaining the previous information up to some degree. Hence, the approach of customization will be suitable for the dataset of which characteristics can change overtime.

There are many algorithms for customization, but most of them have a common limitation that they usually make use of inner parameters of the model,

and thus they are model-specific (Fu et al., 2000; Lyu et al., 1995; Cao and Balakrishnan, 2005). Knowing the parameters of the model serves as additional information which would likely lead to better performance of the algorithm, but it also acts as additional constraint as well. If every algorithm for customization is model-specific, it will be obvious that we will need to have a customization algorithm for each type of models. This surely will be problematic when the type of the model cannot be identified correctly or when there is no customization algorithm for the model, which might likely be the case for any new kind of model. Therefore, we see the benefits of studying on how to perform customization in the way that can be applied to many types of models.

Though the concept of customization can be used on many tasks of machine learning, but this work will focus mainly on classification, due to the obvious application mentioned above. Also, we will base our interest only on the algorithm that can be applied to many types of models.

### **1.1 Objectives**

1. To introduce the approach for performing customization which is not specific to the type of the model.
2. To give the guideline and some examples in how to perform customization that is not specific to the type of the model.

### **1.2 Scope**

1. We will propose a framework and algorithms with the constraints that can be applied to many types of models.
2. We will focus on the task of classification and dimensionality reduction as a preprocess of the classification.
3. We will perform numerical experiments to evaluate the performance of the proposed method.

### 1.3 Procedure

1. Propose the framework for the task or subtask.
2. Propose an algorithm for the framework.
3. Conduct experiments using the proposed algorithm.

### 1.4 Contributions

1. We introduce the approach in performing customization which is not specific to the type of the model and also propose some frameworks and algorithms that belong to this approach.
2. Our work could lead to more development on the algorithms of the same kind.

### 1.5 Organization of the Thesis

We will make the settings of the problem more precise by providing some background knowledges in Chapter 2. In Chapter 3, we will introduce the concept of task-based customization and compare its advantages and drawbacks to the result from model specific algorithms. In Chapter 4 and Chapter 5, we will introduce some frameworks that are based on the concept of task-based customization on classification and dimensionality reduction and also give examples of algorithms that follows the approach, along with numerical results. Conclusion and future work are given in Chapter 6.

จุฬาลงกรณ์มหาวิทยาลัย



# CHAPTER II

## BACKGROUND

### 2.1 Conventions

We state here the definition that will commonly be used in the following sections. We will use an uppercase character such as  $X$  to refer to a matrix, while any lowercase character such as  $x$  will refer to a constant. The boldface letter such as  $\mathbf{x}$  will refer to a vector, and we will denote it as a matrix with one column. Another commonly used convention will be  $I_m$  which will refer to an identity matrix with  $m$  rows. For a dataset  $X$ , each column of  $X$  represents each data where each row represents the value of each attribute.

### 2.2 Customization

Customization or user-adaptation is an approach in modifying/improving the model that is previously trained on a dataset to be better fit to a new dataset. The most useful thing about this approach is that it does not require having the previous data and should be more efficient in the time of operation. However, one obvious drawback could be in the performance of the model resulted from performing customization with the new dataset, comparing with the model that is retrained using both the new dataset and the previous dataset from scratch.

In customization, there will be a problem of how to weight the importance between the previously learned model and customization data appropriately in the case that both of these datasets come from the same distribution. However, it could be proved as a useful approach if the distribution migrates overtime. Each time we perform customization on a model, it will act as trying to make the model be better fit to the new dataset and at the same time slowly making the model forget what it has learned from the previous dataset by a degree. This is why it seems as a good approach to adapt a well learned model to another distribution with some similarity, and it also can be viewed as training an existing model so that it will yield better result with another dataset.

## 2.3 Classification

The task of classification is to predict the value of an attribute of a data, based on the value of other attributes of data as input. There are two tasks which fit this description, namely classification and regression, but the difference between them is in the possible value of the predicted attribute, i.e. the value from classification is discrete and the value from regression is continuous.

### 2.3.1 Weighted Nearest Neighbor

The original  $k$ -nearest neighbor algorithm ( $k$ -NN) (Mitchell, 1997; Hastie et al., 2001; Michie et al., 1994; Han and Kamber, 2000) is considered the simplest algorithm for classification in discussion. The algorithm is based on the idea of voting for the resulted class among  $k$  of the most similar training examples, measured in the feature space. In order to improve this algorithm for better accuracy and also for smoother regression in the probability of the prediction, each chosen candidate for voting will be given with its impact on the classification result, based on how close they are to the target of the prediction. One can also view  $k$ -nearest neighbor as a special case of weighted nearest neighbor with the weight function  $\lim_{k \rightarrow \infty} x^k$  when  $x$  is the Euclidean distance measured between the training data and input data, or to say the infinite norm ( $\|\cdot\|_\infty$ ) of the measured distance.

### 2.3.2 Max Wins

Max Wins (Friedman, 1996), so called voting scheme, is a well-known approach to combine many binary classifiers into one multiclass classifier. This method could be easily described as performing voting between all binary classifiers. The most voted class will be the result of the classification for the multiclass classifier.

### 2.3.3 Adaptive Directed Acyclic Graph

Adaptive Directed Acyclic Graph (ADAG) (Kijirikul et al., 2002) is another approach to combine many binary classifiers into one multiclass classifier.

The method would generally be portrayed as traversing an inverse binary decision tree with each of possible classes as its leaf, as shown in Figure 2.1. The prerequisite requirement of this algorithm is to have existing binary classifiers between each pair of possible classes. At each node, the binary classifier for the pair of respective classes will be used on the data, and the unlikely class will be eliminated from consideration while the more likely one will be passed on along the tree and be used again in the comparison at the next node, until the root is reached with the most likely class. Compared to Max Wins, the main advantage of ADAG is that it requires  $n - 1$  times of classification by binary classifiers to eliminate  $n - 1$  unlikely answers instead of  $n(n - 1)/2$  times of classification by using Max Wins. However, its prediction result would still be inferior to Max Wins. Another characteristic of ADAG is that, given that there is an improvement on each of subclassifiers, the expected improvement on the classifier created by ADAG should yield greater improvement than Max Wins, since the classification would improve on each of the classifiers along the path from the correct class comparing to linear improvement from Max Wins.

Note that though there is an improved version of ADAG called Reordering Adaptive Directed Acyclic Graph (RADAG) (Phetkaew et al., 2003) which could yield even better result than Max Wins, but its characteristic in relying on error estimation for each of the binary classifiers would make it be more complicated to use on any classifiers other than support vector machines (SVMs), requiring k-fold cross validation to estimate the error itself, and hence it will not be used in this dissertation.

#### 2.3.4 Optical Character Recognition

Optical Character Recognition (OCR) is a kind of classification task where we want to predict a character given an image of that character. The most usual form of OCR input is the value in each pixel of the image with fixed size of the character. The process of this task usually begins by scanning a paper for the text. After scanning is done, the next step is to separate the whole image into several individual images of characters before performing OCR on each of them. After that, the results are then combined into words and processed back

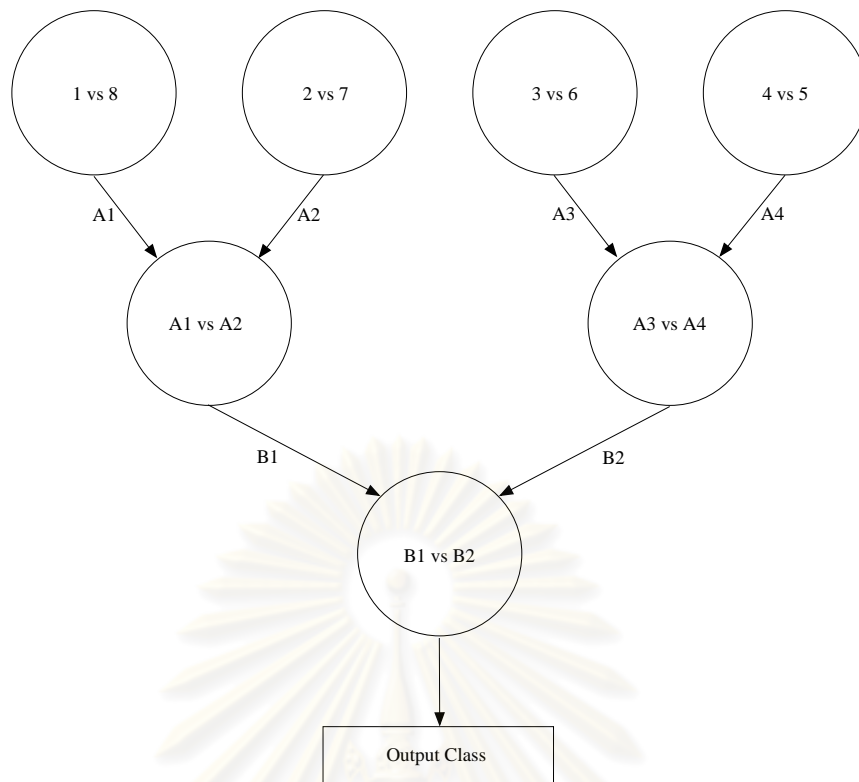


Figure 2.1: Visualization for Adaptive Directed Acyclic Graph.

into predicted text of the page.

### 2.3.5 Handwriting Recognition

Handwriting Recognition (HWR) is another kind of task in classification that we want to predict characters a person writes, with the information about how the writing tool has been stroked. The most general form of input of this task is the pen stroke that designates the position and traces how the pen is drawn and lift. Two-dimensional time series data is generated with the value designating the current position of the pen. However, in many cases, features from this time series will be extracted and be used as in normal classification.

## 2.4 Dimensionality Reduction

Dimensionality reduction refers to a process in reducing the number of attributes of data for further use. Namely, the algorithm for dimensionality reduction will yield the output as a function or process  $\mu : \mathbf{R}^n \rightarrow \mathbf{R}^m$  when  $n$  is the number of attributes as the input and  $m$  is the desired number of attributes.

Instead of using a dataset  $M$  for that task as usual, we will use the result from the transformation,  $\mu(M)$ , as dataset for the task instead.

There are many motivations to perform dimensionality reduction. The most obvious is for the efficiency. Suppose that the task to be performed requires a lot of resource so it cannot satisfy some constraint in time and memory. The choice will be either to change the algorithm or to reduce the load by reducing the size of data. Instead of reducing the number of examples, dimensionality reduction removes some information about each data. The matter to be considered in dimensionality reduction is how to do it in the fashion that the resulted data will be as useful as possible for the task.

Another purpose of dimensionality reduction is not because of limitation, but to improve the result. Usually, a dataset may contain many useless information that could be considered as noise. Performing dimensionality reduction will act as attempting to eliminate the most useless information from the data, namely eliminate the noise, and hence it will be useful for the algorithm that is noise sensitive. Another example is in altering the space of decision for the algorithm. Dimensionality reduction may act as transformation of input space in which the algorithm for the task does not effectively perform, and thus alter the possible decision which the algorithm can produce. As a result, the algorithm will be able to handle more complicated task by using dimensionality reduction as preprocessing. A good example is performing nonlinear dimensionality reduction before applying an algorithm whose decision is based on a limited number of linear planes. In this way we could perform better on the dataset which requires more complicated space of decision.

The main issues about dimensionality reduction can be described as following. The first is how to determine if the result from optimization should perform well on the upcoming task. That is choosing the value to be optimized with the expectation that the resulted dataset will be good for the algorithm which performs the desired machine learning task. The next issue is how to find the function or create the process  $\mu(M)$  that provides good score on that goal, namely the process of optimization. The last one is the constraint or limi-

tation in the space of dimensionality reduction.

## 2.4.1 Linear Dimensionality Reduction

Linear dimensionality reduction is dimensionality reduction with the constraint that  $\mu(M)$  must be linear transformation. This can be viewed as projecting the dataset on a linear subspace. Linear dimensionality reduction is important because of the optimization method. Limiting the space of transformation to be linear transformation results in better formulation of the problem and produces the form of semidefinite programming or quadratic programming which could be optimized with a well-known efficient procedure. In such a case, the method yields the optimal solution, not just a good one.

### 2.4.1.1 Principal Component Analysis

Principal component analysis (PCA) (Han and Kamber, 2000) is an unsupervised linear dimensionality reduction algorithm and is usually considered as the simplest form of dimensionality reduction which is widely used. The goal of PCA is to maximize the variance of a transformed dataset with a fixed number of attributes.

Principal component analysis consists of two main steps. The first is to calculate a covariance matrix of the dataset, i.e. the generalization of variances in many dimensions. Each element of covariance matrix  $C$  in row  $i$  and column  $j$  is according to the formula  $c_{ij} = \sum(x_i - \bar{x})(x_j - \bar{x}_j)$ , or if  $X$  is a matrix of the dataset with each row as each of the data that is translated to have zero mean then  $C = XX^T$ .

The second step of principal component analysis is to perform eigenvalue decomposition of the covariance matrix. Since the matrix  $XX^T$  is squared, symmetric and positive semidefinite, it follows that all the eigenvalues of the matrix are nonnegative real values. We can find the set of orthonormal eigenvectors of the matrix that span the entire space, and we can perform eigenvalue decomposition of the matrix. This means that we can find an orthogonal matrix  $U$  that  $C = U\delta U^T$  where  $\delta$  is the diagonal matrix of eigenvalues ordered by their sizes.

Transformation by principal component analysis to reduce the number of dimensions to  $m$  will be made by using the  $m$  eigenvectors with maximum eigenvalues as the new axis, in which the resulted new attributes are projection of the data on them. Let us consider this to have better understanding of principal component analysis. If we transform dataset  $X$  with orthogonal matrix  $U$  by projecting the data on its axis, then the new dataset will become  $U^T X$ . We can calculate the covariance of this new matrix as following:

$$\begin{aligned}
 U^T X (U^T X)^T &= U^T X X^T U, \\
 &= U^T C U, \\
 &= U^T U \delta U^T U, \\
 &= I_m \delta I_m, \\
 &= \delta.
 \end{aligned}$$

This means that the value of covariance matrix created from the transformed dataset will only exist in the diagonal elements. In another aspect, if we want a dimension that maximizes the variance along that direction, then it will be obvious that the first eigenvector will be the first column of  $U$ . By repeating this process of choosing the direction in the linear space that is orthogonal to all the already chosen directions, we will get the resulted axis of  $m$ -dimensional space as the  $m$  eigenvectors with the maximum eigenvalues of the covariance matrix.

#### 2.4.1.2 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is among one of the simplest supervised linear dimensionality reduction. The goal of linear discriminant analysis is to find a linear transformation that maximizes the discrimination between classes.

## 2.4.2 KPCA Methods of Mahalanobis Distance Learning for Nearest Neighbor

KPCA methods of Mahalanobis distance learning for nearest neighbor (KMMLN) (Chatpatanasiri et al., 2010, 2008) is a group of algorithms for dimensionality reduction. The idea of this type of algorithms is that given a linear dimensionality reduction, the algorithms make a nonlinear dimensionality reduction in the way that is similar to kernel trick of support vector machine by using Kernel Principal Component Analysis (KPCA) (Schölkopf et al., 1998). The algorithm consists of 3 steps, i.e. finding optimal kernel, performing KPCA, then using linear dimensionality reduction algorithms.

## 2.5 Halton Sequence

When there is the need to perform a well distributed sampling inside a close boundary of n-dimensional space, the easiest way will be to sample with uniform randomization. However, if the result from sampling is made in one or two dimensional box and the result of sampling is visualized by the scattering of the result when viewing the close boundary then, in most cases, one could easily feel that all the sampling result could be better distributed if hand-picked by man, as shown in Figure 2.2. In order to evaluate the sampling quality, there are two values called discrepancy and dispersion which serve as measurements, and can be formulated as following (Choset et al., 2005).

$$D(P, \mathbb{R}) = \sup_{R \in \mathbb{R}} \left| \frac{\mu(R)}{\mu(X)} - \frac{|P \cap R|}{N} \right|$$

$$\delta(P, \rho) = \sup_{x \in X} \min_{p \in P} \rho(x, p)$$

where  $D$  is discrepancy,  $\delta$  is dispersion,  $P$  is a set of point samples on the space  $X$ ,  $N$  is the number of points in  $P$ ,  $\mathbb{R}$  is the set of possible spaces used in measuring the value,  $\mu$  is a measure of space or size of the space and  $\rho$  is a metric measuring the distance. Both equations can be interpreted as following: discrepancy measures how the sampling is distributed, comparing the difference between ratio of sampled data and sampling space, while dispersion measures the size of the largest region in which none of the sampling could be found,



Table 2.1: First few terms in Halton sequence for two dimensional space.

$n$	$n_2$	$n_3$	$\Phi_2(n)_2$	$\Phi_3(n)_3$	$\Phi_2(n)$	$\Phi_3(n)$
1	1	1	.1	.1	1/2	1/3
2	10	2	.01	.2	1/4	2/3
3	11	10	.11	.01	3/4	1/9
4	100	11	.001	.11	1/8	4/9
5	101	12	.101	.21	5/8	7/9
6	110	20	.011	.02	3/8	2/9
7	111	21	.111	.12	7/8	5/9
8	1000	22	.0001	.22	1/16	8/9
9	1001	100	.1001	.001	9/16	1/27
10	1010	101	.0101	.101	5/16	10/27

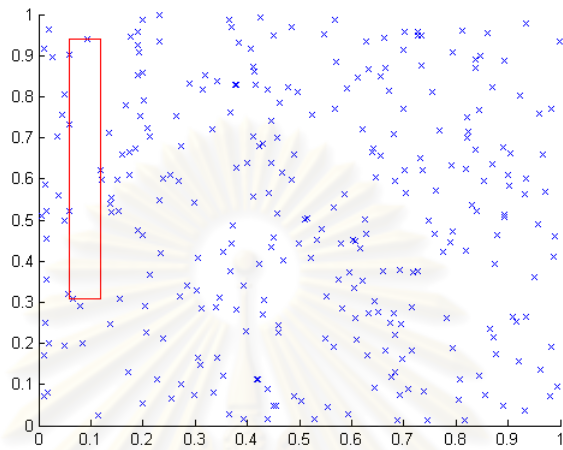
which is a ball in this equation.

In order to achieve better result than randomization and to guarantee the worst case result, one would have to resort to quasirandom sampling, that is using deterministic scheme to determine the sampling sequence, which could be further perturbed by adding some small randomization. One of well-known methods is Halton sequence (Halton, 1960), which could be formulated as following.

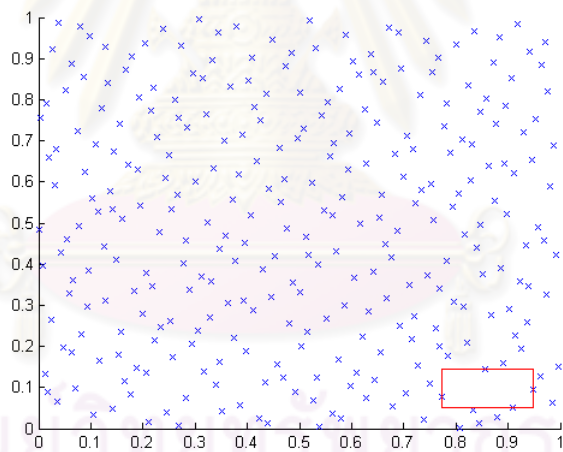
$$p_n = (\Phi_{b_1}(n), \Phi_{b_2}(n), \dots, \Phi_{b_d}(n)).$$

$$\Phi_{b_j}(n) = \sum a_{ij} b_j^{-(i+1)}.$$

Where  $d$  is the number of dimensions,  $b_j$  is the  $j^{\text{th}}$  prime, and  $a_{ij}$  is the  $i^{\text{th}}$  digit of number  $n$  when written as based  $j$  numeral and written backward from behind the decimal, as shown in Table 2.1.



(a) random number generator (dispersion=0.0383)



(b) Halton sequence (dispersion=0.0164)

Figure 2.2: 300 sampled data in two dimensions ( $[0,1]^2$ ) and dispersion value measured with box space. The box shows the area which results in dispersion value.

# CHAPTER III

## TASK-BASED CUSTOMIZATION

In this chapter, we will introduce the concept of task-based customization and also provide some backup arguments for the benefit of algorithms that follow this concept.

### 3.1 Concept of Task-Based Customization

As described in Chapter 2, customization is an approach in machine learning to modify/improve a model in order for it to yield better result. The most important requirement for customization is having the new dataset which we want the model to be adapted to. Also, in order to estimate the improvement from customization, it will require the minimum knowledges about the goal, which is the type of the task which the customization is made on. It is obvious that these knowledges are necessary for customization and by some more consideration ones can see that they are sufficient to perform a customization. However, with additional information, we should be able to perform even better.

Suppose that we do not know about any additional information other than the necessary ones for performing customization. It is obvious that the inner decision of the process will not be clarifiable and thus could only be estimated by observing outputs of some inputs we provide to the process. In simpler words, we must treat the process as a black box. Methods for using the information from this black box can be separated into two kinds. The first one is to keep the black box for future use by conducting some decision process on it and its results instead of using them as the entire process. The other one is to extract the available information from the black box, and then use the information to create a new process that can be clearly identify, and thus discard the black box.

The most common additional information that is used in customization will be the information about the model. Knowing the exact method for calculation of the process will surely yield better understanding than trying to

extract them by observing the results obtained from providing the model with the input. Moreover, once the model is known, the algorithm can alter the inner parameters in meaningful way which will sensibly yield better results.

The other bit of additional knowledge is the algorithm that is used to create the model. Many algorithms may produce similar kind of models which perform the same task. They usually have the same final goal, but the difference between them can be either how they strive to reach that goal or the definition of how they interpret the final goal into the measurement used by the algorithm. For the first case, there are two alternatives in trying to optimize a goal. One method would yield better result but require a lot of resources while the other method may be faster but not guaranteed to provide the optimum solution. For the second case, a good example is dimensionality reduction. The common goal for dimensionality reduction is that the resulted data yields good result for the upcoming task, but each algorithm may interpret the subgoal to achieve this result differently.

### **3.2 Advantages and Drawbacks**

An obvious statement is that, in general, model-specific algorithms should yield better result than task-based customization. However, there are many cases in which task-based customization is necessary due to the lack of information or available methods, e.g. models are in the forms that cannot be interpreted easily, the process is a black box by its nature, there is no algorithm for customization for the model.

# CHAPTER IV

## CUSTOMIZATION FOR CLASSIFICATION

In this chapter, we will propose our frameworks in performing customization for classification. We will also propose some algorithms that behave according to each framework and conduct numerical experiments to evaluate their performances.

### 4.1 Customization for Classification by Transforming Input Space

Here we will present the first approach. The intuition behind this approach is based on an expectation that the space described by the original classifier and the customization data could be very similar to each other. By being similar, we expect that the optimal decision boundary for customization data can be transformed into the decision space resulted from the original classifier via the operation that preserves its topology, namely both of these spaces are homeomorphic to each other.

#### 4.1.1 Generating sequence of transformation for probabilistic classifiers

##### 4.1.1.1 Algorithm

The main objective of this approach is to determine a mapping  $\mu : \mathbb{R}^n \rightarrow \mathbb{R}^n$  to be applied to input data, in order to make the data be better adapted to the classifier. The problem in performing task-based customization for classification is due to the nature of classification that yields discrete output. In the task of classification, there may be many classifiers which classify data correctly. Suppose that there are some data which are not classified correctly and we want to change or customize the classifier on them. Therefore, the issue is how much of the change should be made. If the change is as small as possible, which should be the ideal case when we want to preserve the result from the original classifier, then the decision boundary will be depended mostly on those data and will not have any effect on surrounding the data as much as they should be. Because of the difficulty in determining the proper decision boundary, we

decide to perform the task on probabilistic values of classification results, which can be thought of as performing regression and will be easier for the task-based constraint. Here, we present an algorithm for generating the mapping for a probabilistic binary classifier in Algorithm 1.

**input** : an original classifier, customization data  
**output**: transformation sequence

```

1 stepsize  $\leftarrow$  initial_stepsize;
2 transform_sequence  $\leftarrow$   $\emptyset$ ;
3 curdata  $\leftarrow$  customization data;
4 cur_total_prob  $\leftarrow$   $\sum p(\text{curdata}, \text{classifier})$ ;
5 while stepsize > stepsize_limit do
6   center  $\leftarrow$  getnext(halton_sequence);
7   direction  $\leftarrow$  estimate_gradient(center, classifier);
8   transformation  $\leftarrow$  (center, direction, stepsize);
9   mapdata  $\leftarrow$  transform(curdata, transformation);
10  map_total_prob  $\leftarrow$   $\sum p(\text{mapdata}, \text{classifier})$ ;
11  if map_total_prob > cur_total_prob then
12    curdata  $\leftarrow$  mapdata;
13    cur_total_prob  $\leftarrow$  map_total_prob;
14    transform_sequence  $\leftarrow$ 
      append(transform_sequence, transformation);
15  end
16  stepsize  $\leftarrow$   $\alpha \times$  stepsize;
17 end

```

**Algorithm 1:** Generating transformation sequence.

The mapping is separated into many transformation sequences, for its flexibility in order to achieve complicated nonlinear transformation. As shown in Algorithm 1, it will pick the center of each transformation from Halton sequence, to guarantee coverage of the input space. Then according to the current *stepsize* and estimated gradient, we will have a candidate for transformation described by three factors, i.e. the center of transformation, direction and magnitude. The transformation will be kept if it is likely to lead to better result. The change on space around the center of transformation is calculated as being

influenced by the attempt to move the space at center of transformation to a new position. For simplicity of calculation, we let the change be in the same direction to the change on the center, while the magnitude of the change is calculated by a distribution, proportional to the size of the change on the center. To achieve convergence, *stepsize* is gradually reduced in each loop.

In order to improve the algorithm, we further consider about the problem with imbalance on the number of training data, i.e. there are some classes with overwhelmingly more or less number of training instances. A problem may arise when using the proposed algorithm by summing up the probabilistic values of predictions resulted from all customization data, which could cause the algorithm to be over-biased on some classes with high number of training instances, as shown in Figure 4.1. In order to better cope with this problem and to lessen the difference from getting probabilistic outputs from different type of classifiers, we introduce the idea of using an output function. The purpose is to assign more importance to the data with higher probabilistic values, which gives the effect that all customization data will be more likely to be correctly classified, and lessens the effect which will mislead the decision boundary by other customization data. Therefore, instead of using probabilistic values directly, in this paper, we propose using the value from output function  $s(x) = 1 - ((1 - p(x))^2)$  which its shape is as shown in Figure 4.2.

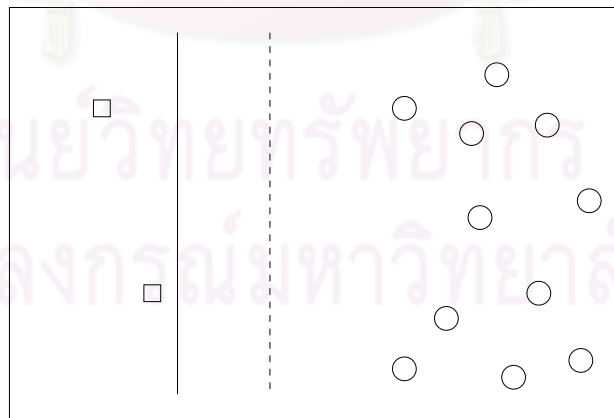


Figure 4.1: Visualization of the problem in determining objective for mapping. The class of data are shown by depicting each of them as square and circle. The solid line is a likely decision boundary resulted from summing the objective value calculated from each data and the dashed line is the most suitable decision boundary.

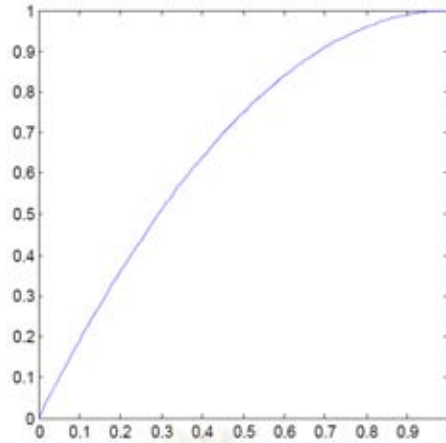


Figure 4.2: Plotting of output function  $s(x) = 1 - ((1 - p(x))^2)$ .

#### 4.1.1.2 Numerical Results

To demonstrate the usefulness of this algorithm, we generated experimental data in two dimensions. The result for the original probabilistic classifier was determined by equation  $p_1 = \frac{x+y}{\sqrt{2}}$ . The decision boundary was a straight diagonal line separating half of the input space, given that the input data was normalized into the range of  $[0, 1]^2$ . The probabilistic value was directly proportional to the distance measured from the decision boundary, with the minimum and maximum values of 0 and 1, respectively. On the other hand, customization data was generated from different distribution, determined by whether values of both attributes were less than 0.7. We applied Algorithm 1 with the parameter  $initial\_stepsize = 0.1$ ,  $stepsize\_limit = 0.001$ ,  $\alpha = 0.99$  and with changing impact as exponential function:  $e^{-x/\beta}$  where  $\beta = 1$ . We conducted an experiment using the mentioned probabilistic classifier, and a weighted nearest neighbor classifier generated from randomly sampled data according to the decision boundary from the equation, using  $1/x^k$  as weight function where  $k$  is the number of attributes, considering only 10 nearest neighbors for each class. These setting of parameters will be used as default value in the rest of this work. The result is shown in Table 4.1 and can be visualized as in Figure 4.3.

In Table 4.1, the result from classifying mapped customization data using the original classifier yielded some improvement compared to the result from classifying customization data. This means that the process transforms



Table 4.1: Accuracy comparison between customization data, mapped customization data and mapped customization data using the proposed output function.

	CUST DATA	MAPPED CUST DATA	OUTPUT FUNCTION
ORIGINAL CLASSIFIER	82.40	82.90	<b>87.70</b>
WNN	87.00	85.30	<b>89.00</b>

the space of customization data to be better fit with the space of the original classifier. However, when we tried using data sampled from the original classifier to create a probabilistic classifier, which was weighted nearest neighbor (WNN), the result from using the weighted nearest neighbor for mapped customization data became worse due to misleading of the probabilistic value from voting. By using the concept of output function, the problem was lessened and the result was shown with obvious improvement.

#### 4.1.2 Applying with other classifiers

##### 4.1.2.1 Algorithm

In order to apply the algorithm for probabilistic classifiers to all non-probabilistic classifiers, we propose an algorithm to create a probabilistic classifier from the former one, for the sole use of generation of transformation sequence. To treat the original classifier as a black box, we have to perform sampling and classify them using the classifier, whose result will be used as data to train the generated probabilistic classifier. Moreover to keep the influence from the nature of the classifier model at minimum, we decide to use an instance-based classifier. This results in the following algorithm to create a weighted nearest neighbor through sampling for a binary classifier in Algorithm 2.

**input** : original classifier  
**output**: weighted nearest neighbor classifier

```

1 NN_data  $\leftarrow \emptyset$ ;
2 NN_weight  $\leftarrow \emptyset$ ;
3 while size(NN_data) < datasize do
4   newdata  $\leftarrow$  getnext(halton_sequence);
5   predictweight  $\leftarrow$  classify(newdata,NN_data,NN_weight);
6   if predictweight < 0.5 then
7     NN_data  $\leftarrow$  append(NN_data,curdata);
8     NN_weight  $\leftarrow$  append(NN_weight,1-(2  $\times$  predictweight));
9   end
10 end

```

**Algorithm 2:** Generating a weighted nearest neighbor classifier from an existing classifier.

The main idea for this algorithm is that the newly generated data will be classified by the current classifier. If the classification result is wrong, then the current classifier will be improved by adding the newly generated data as a new instance. The weight of the instance is determined by the weight from the classification result.

For the generated classifier to be appropriately useful, there are two conditions which should be satisfied. Firstly, the classifier should be able to provide probabilistic prediction for each class, in a meaningful sense. Secondly, its decision boundary should be similar to that of the original classifier at least up to some degree. The sampling approach used in the algorithm implies curse of dimensionality, i.e. the amount of sampling required to satisfy the second condition exponentially increases with the number of attributes. In order to fix this problem, we decide to use the method of combining the result from the generated classifier with the original one.

Given that the weight for the original classifier is higher, the combined classifier will obviously predict the same result as the original, and the generated classifier will act as an alteration on probabilistic values, as in the following equation.

Table 4.2: Accuracy from the generated classifier when using the algorithm that is applicable to non-probabilistic classifier.

CUSTOMIZATION DATA	82.40
MAPPED CUSTOMIZATION DATA	82.40
MAPPED WITH OUTPUT FUNCTION	<b>83.20</b>

$$p_{new} = ((1 + \alpha)p_{original} + p_{generated}) / (2 + \alpha).$$

Where  $\alpha$  is a small positive real. The drawback of this method is that in order to achieve the same classification result as the original classifier, we have to trade that off with the continuity on the probabilistic value from the generated classifier.

We performed an experiment to evaluate the generated classifier in the previous subsection, using Algorithm 2 with datasize equal to 2,000. We replaced the original classifier with the generated one while all other parameters were set in the same way as in the previous experiment. The result is shown in Table 4.2.

Comparing results between Table 4.1 and Table 4.2, it is not surprising that the result from using the algorithm that is applicable to non-probabilistic classifier was inferior due to the additional error resulted from having the additional step in generating a probabilistic classifier from non-probabilistic one. However, the result in Table 4.2 still show slight improvement compared to the result before mapping.

#### 4.1.2.2 Numerical Results

We conducted an experiment to demonstrate the improvement made by the algorithm. In order to measure the performance we used the real-world datasets obtained from the UCI machine learning repository (Asuncion and Newman, 2007), *pendigits*, representing the tasks of classifying numeral digits for handwriting recognition, to demonstrate the task in adapting the classifier to another similar dataset. In the reality, the input data from handwriting recog-

Table 4.3: Accuracy of each transformed subclassifier.

	ORIGINAL	TRANSFORMED
WNN	99.39 $\pm$ 1.03	<b>99.69 <math>\pm</math> 0.57</b>
NEURAL NET	98.86 $\pm$ 1.42	<b>99.40 <math>\pm</math> 0.98</b>
SVMs	99.46 $\pm$ 0.68	<b>99.59 <math>\pm</math> 0.66</b>

nition is sequence of pen strokes, which is sequence of time series. However, all 16 attributes in the *pendigits* dataset were extracted features from this sequence of time series and we used these values for classification. The original training data was used to train the classifier, while the original test data, being generated from different group of users, was separated into two groups for the task of customization and testing. The original training data and the other set of data had 7,494 and 3,498 instances respectively which were almost equally distributed to 10 classes representing digits 0-9, and for each class, we assigned 50 instances as customization data. As a result, we had roughly 750 training data, 50 customization data and 300 test data for each class. The setting of parameters was the default value as stated in the previous subsection and all of data were normalized to be in the range of  $[0, 1]$ .

In this experiment, we used three types of classifiers: weighted nearest neighbor, neural networks (Mitchell, 1997; Hastie et al., 2001; Michie et al., 1994; Han and Kamber, 2000) and support vector machines (Cristianini and Shawe-Taylor, 2000), to create one-against-one classifiers between each pair of classes to which the customization was applied. Then these subclassifiers were combined by two methods of voting and decision trees. So, there were 45 subclassifiers for each type which were trained with roughly 1,500 training data. We performed customization on each subclassifier with 100 customization data and the experimental result was taken from about 300 test data for the experiment on subclassifiers and all 2,998 test data for the combined classifier. The results are as shown in Table 4.3 and Table 4.4.

An important note is that in Table 4.3, the value of standard deviation was not from cross validation but was the standard variation from the classifier of each class. Thus, high value of standard deviation means that there were high

Table 4.4: Accuracy of the combined classifier created from transformed sub-classifiers.

COMBINING METHOD	ALGORITHM	ORIGINAL	TRANSFORMED
MAX WINS	WNN	97.80	<b>98.13</b>
	NEURAL NET	95.43	<b>96.90</b>
	SVMs	97.67	<b>98.37</b>
ADAG	WNN	97.80	<b>98.10</b>
	NEURAL NET	95.76	<b>97.73</b>
	SVMs	97.67	<b>98.37</b>

Table 4.5: Accuracy of each transformed subclassifier when using generated subclassifiers to determine the transformation.

	ORIGINAL	TRANSFORMED
NEURAL NETWORK	<b>98.86 ± 1.42</b>	98.72 ± 1.39
SVMs	<b>99.46 ± 0.68</b>	98.71 ± 1.61

differences in difficulties in classifying each pair of digits. This fact is obvious since some digits like 1 and 7 are harder to classify apart with writing strokes, compared to classification between 0 and 4 which there are differences in the numbers of pen strokes, curve of the pen strokes and direction of pen strokes. For the experiment on the method that is applicable to non-probabilistic classifiers, we used Algorithm 2 to generate weighted nearest neighbor classifiers with 2,000 data. The results are shown in Table 4.5 and Table 4.6.

In Table 4.5 and Table 4.6, we did not conduct the experiment with weighted nearest neighbor on this matter since it is more realistic to use the data used in weighted nearest neighbor itself instead of generating new data. The results obviously imply that the generated probabilistic classifier obtained by transformation is worse than the original classifier. However, there still exists the problem about curse of dimensionality as previously stated. Since *pendigits* has sixteen attributes then it will require at least  $2^{16} = 65,536$  for the data to exist in every orthants; each sector determines the positiveness of the value in each axis. So it can be seen by how sparse the generated dataset with 2,000 instances is in the sixteen dimensional space.

Table 4.6: Accuracy of the combined classifier created from transformed sub-classifiers when using generated subclassifiers to determine the transformation.

COMBINING METHOD	ALGORITHM	ORIGINAL	TRANSFORMED
MAX WINS	NEURAL NETWORK	<b>95.43</b>	95.00
	SVMs	<b>97.67</b>	94.96
ADAG	NEURAL NETWORK	<b>95.76</b>	94.80
	SVMs	<b>97.67</b>	94.70

Though the result was not good for data with many attributes, in the case of a small number of attributes, this method was still able to yield satisfying result. The other issue is that though the classifiers are already probabilistic but each probabilistic classifier may return the value of probability in different sense so it may be a good idea to also use this method to generate a probabilistic classifier to guarantee the worst case in determining transformation sequence.

### 4.1.3 Overall Framework

Combining with the algorithm previously mentioned, we will get the framework of customization for classification as shown in Figure 4.4. All diagrams in our works follow this description. A block with rectangular shape represents a process and an oval shaped block represents data. A solid arrow represents the relation of being input to a process while a dashed arrow represents the relation of a process generating its output. A dashed box covers parts of the diagram that are kept as important result for continual usage. In this diagram, the box also shows the process within a customized classifier.

The framework starts with the original classifier which we want to perform customization on. Next, we generate a probabilistic classifier to be used in the process from the customization data and the original classifier. Then we determine the transformation sequence to make the customization data be better fit to the generated classifiers (since the generated classifiers have the same decision boundary to the original classifiers, the transformed customization data will also be better fit to the original classifiers as well). The customized classifier consists of the transformation sequence and the original classifier. Each time

classification is performed, the input data will be mapped with the transformation sequence then the mapped data will be classified by the original classifier, giving the prediction.

## 4.2 Customization for Classification by Patching

Here, we will present another alternative approach. The goal of this approach is to use customization data to classify if the result from the original classifier is trustable in a region. Otherwise, we will classify it with the other classifier created from the customization data instead. In our case, we will combine these processes into a classifier called patcher, which contains the same set of classes as those of the original classifier with one more class representing that the answer should be left to the original classifier. The reason that we name this approach patching is because this method provides the intuition that the result from the original classifier will be corrected in untrustable region by patching it with the result from another classifier, while the result in trustable region which needs no correction will be left as is. Thus, this gives the intuition as the original classifier is patched to fix the bad result.

### 4.2.1 Algorithms

We will present four algorithms which belong to this approach in this subsection.

```

input : original classifier, customization data
output: patching classifier
1 NN_data  $\leftarrow$  customization_data;
2 foreach data in NN_data do
3   | if  $\text{classify\_1nn}(\text{data}, \text{NN\_data}) = \text{data.class}$  then
4   | |  $\text{data.class} \leftarrow \text{original\_classifier}$ ;
5   | end
6 end

```

**Algorithm 3:** Patching using 1-nearest neighbor.

Algorithm 3 is the simplest among the four algorithms of this group which will be presented within this section. The idea is to use 1-nearest neighbor and

use the result from the original classifier if the data from 1-nearest neighbor is classified by the original classifier correctly.

**input** : original classifier, customization data

**output**: patching classifier

```

1 data_queue ← customization_data;
2 NN_data ← ∅;
3 correctset ← ∅;
4 while notempty(data_queue) do
5   | data ← dequeue(data_queue);
6   | if classify_1nnpatcher(data,NN_data,original_classifier)
7   | = data.class then
8   |   | correctset ← append(correctset,data);
9   | end
10  | else
11  |   | if classify(data,original_classifier) = data.class then
12  |   |   | data.class = original_classifier;
13  |   | end
14  |   | NN_data ← append(NN_data,data);
15  |   | foreach correctdata in correctset do
16  |   |   | if
17  |   |   |   | classify_1nnpatcher(correctdata,NN_data,original_classifier)
18  |   |   |   | ≠ data.class then
19  |   |   |   |   | remove(correctset,correctdata);
20  |   |   |   |   | enqueue(data_queue,correctdata);
21  |   |   | end
22  |   | end
23  | end
24 end

```

**Algorithm 4:** Patching using 1-nearest neighbor with the reduced number of instances.

Algorithm 4 is a slight modification version of Algorithm 3 in order to reduce the number of data used in the 1-nearest neighbor patcher. The idea is to add one data into the patching classifier at a time, and try not to add it if it



is classified correctly. Different from Algorithm 3, the classifier resulted from using Algorithm 4 will also depend on the order of data used in the algorithm. The procedure which we try to reduce the number of training data for 1-nearest neighbor will reduce the time used in classification and will also reduce the complexity of decision boundary, which might yield better result as well.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

**input** : original classifier, customization data

**output**: patching classifier

```

1 data_queue ← customization_data;
2 patch_data ← ∅, correctset ← ∅;
3 while notempty(data_queue) do
4   data ← dequeue(data_queue);
5   if classify_patcher(data, patch_data, ori_classifier) = data.class then
6     correctset ← append(correctset, data);
7   else
8     if classify_patch(data, patch_data) = ori_classifier then
9       data.radius ← find_radius(data, patch_data, correctset);
10      patch_data ← append(patch_data, data);
11     else if classify(data, ori_classifier) ≠ data.class then
12       data.radius ← find_radius(data, patch_data, correctset);
13       foreach ball in patch_data do
14         ball.radius ← reduce_radius(ball, data);
15       end
16       patch_data ← append(patch_data, data);
17     else
18       foreach ball in patch_data do
19         ball.radius ← reduce_radius(ball, data);
20       end
21     end
22     foreach correctdata in correctset do
23       if classify(correctdata, ori_classifier) ≠ correctdata.class then
24         remove(correctset, correctdata);
25         enqueue(data_queue, correctdata);
26       end
27     end
28   end
29 end

```

**Algorithm 5:** Patching with a reduced number of hyperspheres.

Algorithm 5 will create open balls or hyperspheres without boundary

with customization data as their centers. The result within the region of each ball is classified as belonging to the same class as the customization data which is the center of the ball. The result for undefined region is taken from the original classifier. The algorithm will perform similarly to Algorithm 4 except that the region of the hypersphere requires different handling. First of all, each hypersphere must not intersect with others, and that is why when there are some changes in *patch\_data*, the algorithm will require more complicated procedure. Note that there are two functions that require more description in Algorithm 5. For the first one, *reduce\_radius* will shrink down an existing hypersphere if the hypersphere is the cause of wrong classification or if the two arguments intersect with each other. If the wrong classification is caused by a new hypersphere then it will make the two hyperspheres become almost touched, and if it is a new data in different class then it will make the radius of the hypersphere half of the distance between the two instances. For the second one, *find\_radius* will be used when a new hypersphere or data is added. The function will try to make the new hypersphere as large as possible with some constraints that it must not intersect with other hyperspheres and that its radius is not more than half of the distance between its center and other instances not in any patch region. Since this may not be very obvious we will provide a proof that this algorithm will terminate as in Proposition 1.

**Proposition 1.** *Algorithm 5 will eventually terminate.*

*Proof.* In each loop, the algorithm will dequeue and perform at least one of the following actions; add data from the queue into *correctset* in the case that the data is correctly classified by the current classifier or add data from the queue to *patch\_data* then reduce radius of an existing hypersphere in order to fix the result of the current classifier. Each time enqueue is performed, the data must be taken from *correctset*, which causes the total numbers of data in the queue, *correctset* and *patch\_data* to always be the same. In each loop when enqueue is performed, it must also add another instance to *patch\_data* or reduce the size of a hypersphere. Adding an instance to *patch\_data* will cause the total number of instances in *correctset* and the queue to be lower as well and will eventually make the queue empty. The size of the hypersphere can be reduced for a limited number of times, bounded by the total number of instances. And if the radius

is at the lowest possible, then there cannot be an intersection with any other data, which causes the loop to perform other action instead. When both of these actions are not performed any longer then the number of instances in the queue cannot be increased. Hence, the loop will terminate. ■

**input** : original classifier, customization data  
**output**: patching classifier

```

1 patch_data ← customization_data;
2 foreach data in patch_data do
3   | s ← patch_data which (.class ≠ data.class);
4   | data.radius ← minx∈sd(data, x)/2;
5 end

```

**Algorithm 6:** Patching with intersectable hypersphere.

Algorithm 6 is different from the rest in that it allows intersection of regions, as long as they are both of the same class. The algorithm will make the radius of each hypersphere half of the distance between the center of the hypersphere and the closest instance in a different class.

#### 4.2.2 Numerical Results

In this section, we conducted experiments on the *pendigits* dataset and the classifiers in the previous section. Also, to better demonstrate the result, we decided to also use other datasets from the UCI machine learning repository called *optdigits*, representing the task of optical character recognition. The dataset *optdigits* represents images of digits 0-9 in 8x8 pixels, resulted in 64 attributes each of which is an integer in the range of 0-16 reflecting grayscale value of the images. The dataset was separated by the group of writers in the same way as *pendigits*, consisting of 3,823 instances of original training data and 1,797 instances that were used for customization and testing. For each class, 50 instances were assigned as customization data, and thus we had roughly 380 training data, 50 customization data and 130 test data for each class. We used principal component analysis to reduce the number of attributes to 16, the same as in *pendigits*, and then normalized the value of these 16 attributes to be in the range  $[0, 1]$  before preparing the data for the experiments in the same way. The

Table 4.7: Accuracy of patched subclassifiers from *pendigits* data.

	ORIGINAL	ALGO 3	ALGO 4	ALGO 5	ALGO 6
WNN	99.39 ± 1.03	99.62 ± 0.83	99.62 ± 0.82	99.61 ± 0.85	<b>99.72 ± 0.74</b>
NN	98.86 ± 1.42	99.41 ± 0.74	99.46 ± 0.71	99.26 ± 0.93	<b>99.51 ± 0.84</b>
SVMs	99.46 ± 0.68	99.63 ± 0.45	99.65 ± 0.47	99.63 ± 0.43	<b>99.75 ± 0.35</b>

Table 4.8: Accuracy of patched subclassifiers from *optdigits* data.

	ORIGINAL	ALGO 3	ALGO 4	ALGO 5	ALGO 6
WNN	99.58 ± 0.90	99.53 ± 0.92	99.62 ± 0.82	99.61 ± 0.85	<b>99.72 ± 0.74</b>
NN	98.22 ± 2.02	98.23 ± 2.03	99.46 ± 0.71	99.26 ± 0.93	<b>99.51 ± 0.84</b>
SVMs	<b>99.80 ± 0.37</b>	99.78 ± 0.39	99.65 ± 0.47	99.63 ± 0.43	99.75 ± 0.35

experiments on combined classifiers are shown in Table 4.7, Table 4.8 and Table 4.9. Since the algorithm can be applied to multiclass classifiers without any modification, we also conducted an experiment on combined classifiers, and the result is shown in Table 4.10.

The result from Table 4.7, Table 4.8, Table 4.9 and Table 4.10, show that the proposed algorithm improved the classification result in most cases, Algorithm 6 provided the best result among the four proposed algorithms which perform customization for classification by patching. An interesting issue is the result from applying the algorithms with multiclass classifiers. In Table 4.9 and Table 4.10, applying the algorithms with multiclass classifiers brought worse result in most cases, with the exception of Algorithm 3. For Algorithm 4 and Algorithm 5, their results depended on the order of the data used in the algorithms, and applying the algorithms with multiclass classifiers gave more bias resulted from the order of the data. For Algorithm 6, as the radius of the ball was calculated from the distance to the other data with different class, using the algorithm with multiclass classifiers provided smaller radius of the hyperspheres compared to using the algorithm on each subclassifier because the number of data in different classes had increased.

Now we compare both frameworks of transforming input space and patching. Transforming input space seems like a good approach in performing con-

Table 4.9: Accuracy of combined classifier from patched subclassifier.

	ALGORITHM	ORIGINAL	ALGO 3	ALGO 4	ALGO 5	ALGO 6
PENDIGITS MAX WINS	WNN	97.80	98.10	98.10	97.97	<b>98.50</b>
	NEURAL NET	95.43	96.83	97.36	96.46	<b>97.67</b>
	SVMs	97.67	98.37	98.50	98.17	<b>98.63</b>
PENDIGITS ADAG	WNN	97.80	98.20	98.10	98.03	<b>98.50</b>
	NEURAL NET	95.76	97.30	97.40	97.03	<b>97.73</b>
	SVMs	97.67	98.33	98.07	98.17	<b>98.60</b>
OPTDIGITS MAX WINS	WNN	97.22	97.07	96.38	97.22	<b>97.30</b>
	NEURAL NET	93.60	93.99	94.06	93.83	<b>94.37</b>
	SVMs	98.54	98.54	98.15	<b>98.61</b>	98.54
OPTDIGITS ADAG	WNN	97.22	97.07	96.68	97.22	<b>97.30</b>
	NEURAL NET	92.60	93.06	92.37	92.75	<b>93.29</b>
	SVMs	98.54	98.54	98.38	<b>98.61</b>	98.54

tinuous changes to create a more suitable classifier. However, an obvious drawback from transforming input space is the high number of parameters of the algorithm which is also due to applying the algorithm to the task of classification which yields discrete result. In contrast, patching is much simpler and has few parameters which cause it to have much less problem in overfitting. The idea of patching is to use the result from each of two classifiers, the original one and the one created from customization data. We can give some estimation like lower bound or expected performance of a classifier resulted from patching with both classifiers. Suppose that we determine whether we should believe the original classifier by random then the expected accuracy of the new classifier should be the average value of the accuracy from both classifiers. Moreover, by the fact that our decision is based on a plausible reasoning, the resulted classifier from patching should obviously yield better result than the average accuracy of both classifiers. Comparing the result from Table 4.4 and Table 4.9 which are the results from transforming input space using probabilistic value of *pendigits* and patching, patching by Algorithm 6 yields better result. However, this is due to difficulty in transforming input space. In summary, transforming input space is an approach with high potential but is hard to achieve and patching is a simple method but has limited potential in both upper bound and lower bound of the result.

Table 4.10: Accuracy of patched combined classifier.

	ALGORITHM	ORIGINAL	ALGO 3	ALGO 4	ALGO 5	ALGO 6
PENDIGITS MAX WINS	WNN	97.80	<b>98.40</b>	97.97	98.17	98.27
	NEURAL NET	95.43	<b>97.73</b>	97.40	96.83	96.83
	SVMs	97.67	98.27	97.30	98.10	<b>98.30</b>
PENDIGITS ADAG	WNN	97.80	<b>98.40</b>	97.97	98.17	98.27
	NEURAL NET	95.76	<b>98.03</b>	97.26	97.20	97.26
	SVMs	97.67	98.27	97.30	98.10	<b>98.30</b>
OPTDIGITS MAX WINS	WNN	97.22	97.07	95.07	97.22	<b>97.30</b>
	NEURAL NET	93.60	<b>93.99</b>	89.67	93.83	<b>93.99</b>
	SVMs	98.54	98.46	96.92	98.61	<b>98.69</b>
OPTDIGITS ADAG	WNN	97.22	97.07	95.07	97.22	<b>97.30</b>
	NEURAL NET	92.60	<b>93.06</b>	88.97	92.91	<b>93.06</b>
	SVMs	98.54	98.46	96.92	98.61	<b>98.69</b>

### 4.3 Customization for Classification with Nonlinear Dimensionality Reduction

This framework is based on the idea of transforming input space as well, but with a different method in finding the transformation. Instead of finding sequence of small transformations, this framework will use dimensionality reduction to find the transformation between both input spaces. This framework is shown in Figure 4.5 and the resulted classifier is shown in Figure 4.6.

The framework begins with generating data that represents the original classifier by randomization for its attribute values and uses the class from prediction of the original classifier. After that, we will embed the space of this data and the customization data into a higher-dimensional space. We then use supervised linear dimensionality reduction algorithms, such as KMMLN, with the expectation that it will align both data sets on each other without changing their topology. After that we will perform regression such that the data in that space should be mapped back into the space of the original classifier based on the value of the generated data. The new classifier will work by transforming the test data into the space that aligns it with the data belonging to the original classifier. After that, the values of attributes of the data in the original space corresponding to the test data will be calculated by regression, and that attribute values will be used by the original classifier for prediction.

### 4.3.1 Numerical Results

We show the experimental result of the proposed framework. For visualization, we used 2-dimensional data with no noise and with two possible classes, to test how the framework performed on each kind of basic linear transformation. We used the same customization data and test data while varying the original classifier which is 1-nearest neighbor generated from the space that can be mapped onto customization data perfectly by translation, scaling, and rotation. All the data in this experiment is shown in Figure 4.7.

The reason for us to choose 1-nearest neighbor as the original classifier is for easiness in the step of generating data that represents the original classifier. In this case, we used the data in 1-nearest neighbor directly. The next issue was how to embed both data into the same space. An easy implementation may be to use translation, or any kind of rigid transformation that makes both datasets not overlap with each other, and even easier, we may add another dimension whose value represents the dataset where each data comes from. The visualization of this is as in Figure 4.8.

Note that, this framework relies mainly on the expectation that the algorithm for nonlinear dimensionality reduction, which is the algorithm in the group of KMMLN in this case, will be able to align both datasets onto each other. To achieve this result, we considered both the method of embedding and the inner process of the KMMLN algorithm. In this experiment, we used the algorithm KDNE, KLMNN and KNCA which are the kernelized versions of linear dimensionality reduction algorithms DNE (Zhang et al., 2007), LMNN (Weinberger and Saul, 2009) and NCA (Goldberger et al., 2005) respectively, and the algorithms for them are already stated in the original paper of KMMLN (Chatpatanasiri et al., 2010, 2008). The results are shown in Table 4.11 and Table 4.12.

The results show the accuracy when using an original 1-nearest neighbor classifier and the accuracy of 1-nearest neighbor with customization data. There is clearly significant improvement compared to the original classifier, and some improvement compared to 1-nearest neighbor on customization data. How-



Table 4.11: Result from customization using linear dimensionality reduction algorithm.

	ORIGINAL	CUSTOM DATA	DNE	LMNN	NCA
TRANSLATION	83.6	<b>93.4</b>	83.6	83.5	83.7
SCALING	86.0	<b>93.4</b>	86.0	86.1	86.3
ROTATION	88.6	<b>93.4</b>	88.6	88.6	88.6

Table 4.12: Result from customization using the KMMLN algorithm.

	ORIGINAL	CUSTOM DATA	KDNE	KLMNN	KNCA
TRANSLATION	83.6	93.4	87.0	93.8	<b>97.9</b>
SCALING	86.0	93.4	87.7	<b>94.1</b>	87.9
ROTATION	88.6	93.4	91.0	<b>93.6</b>	91.1

ever, two issues need to be noted. First is that all the three datasets in this experiments are simple linear transformation from a dataset with really simple decision boundary, as shown in Figure 4.8. The second issue is that though the numerical result looks fine but from visualization, the two datasets did not align on each other when using KMMLN, with the exception of performing KNCA on the translation dataset which yielded much more significant difference compared to other cases in the experiment. Thus, the algorithm may not have much potential in higher dimensional datasets with more complicated decision boundary.

In conclusion, the framework seems to have the potential to yield satisfying result, but we could not bring experimental results as we expected yet. So we only present this framework here as it may be a good idea for further development in this line of work.

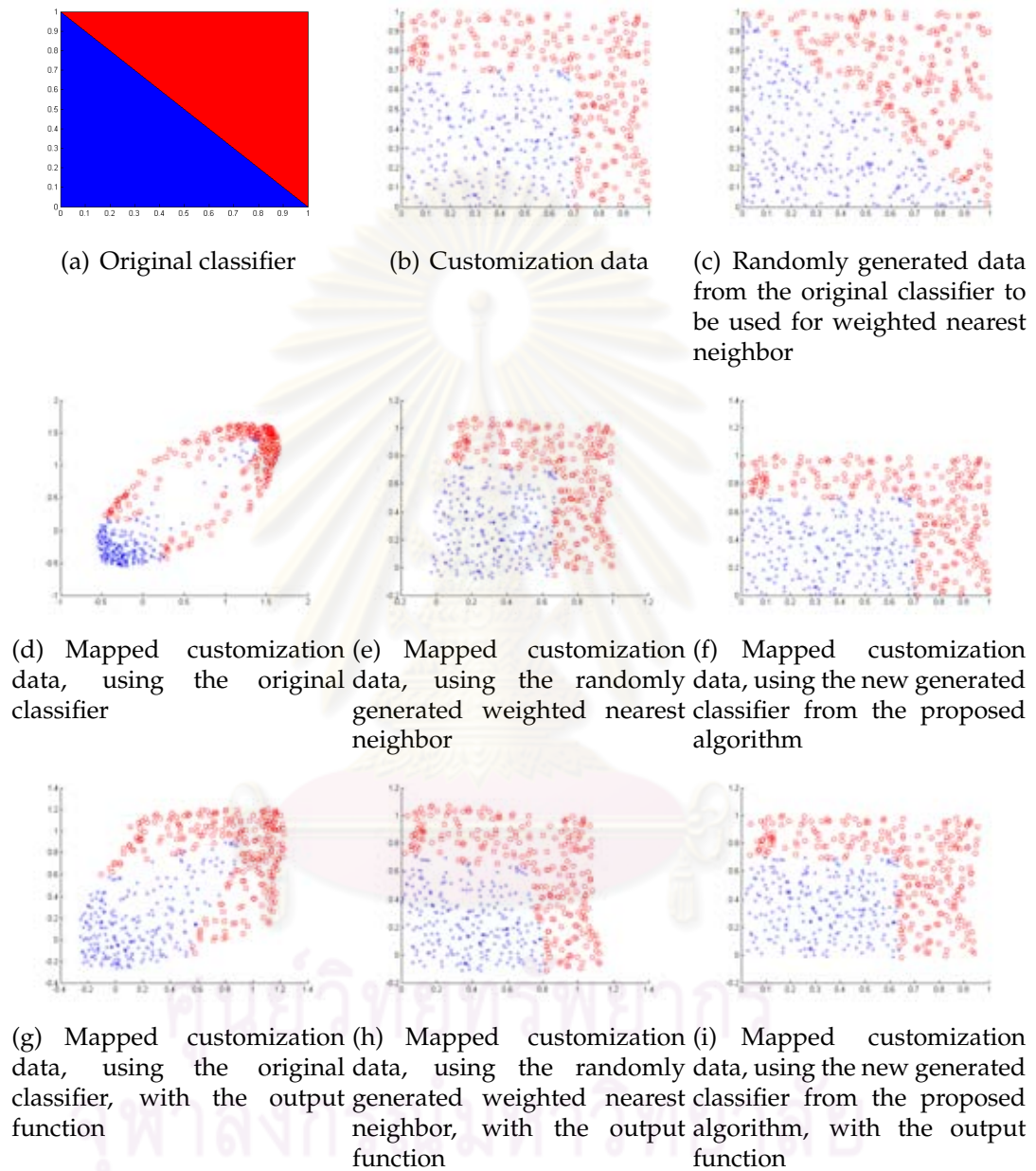


Figure 4.3: Visualization of results from transforming generated probabilistic classifiers.

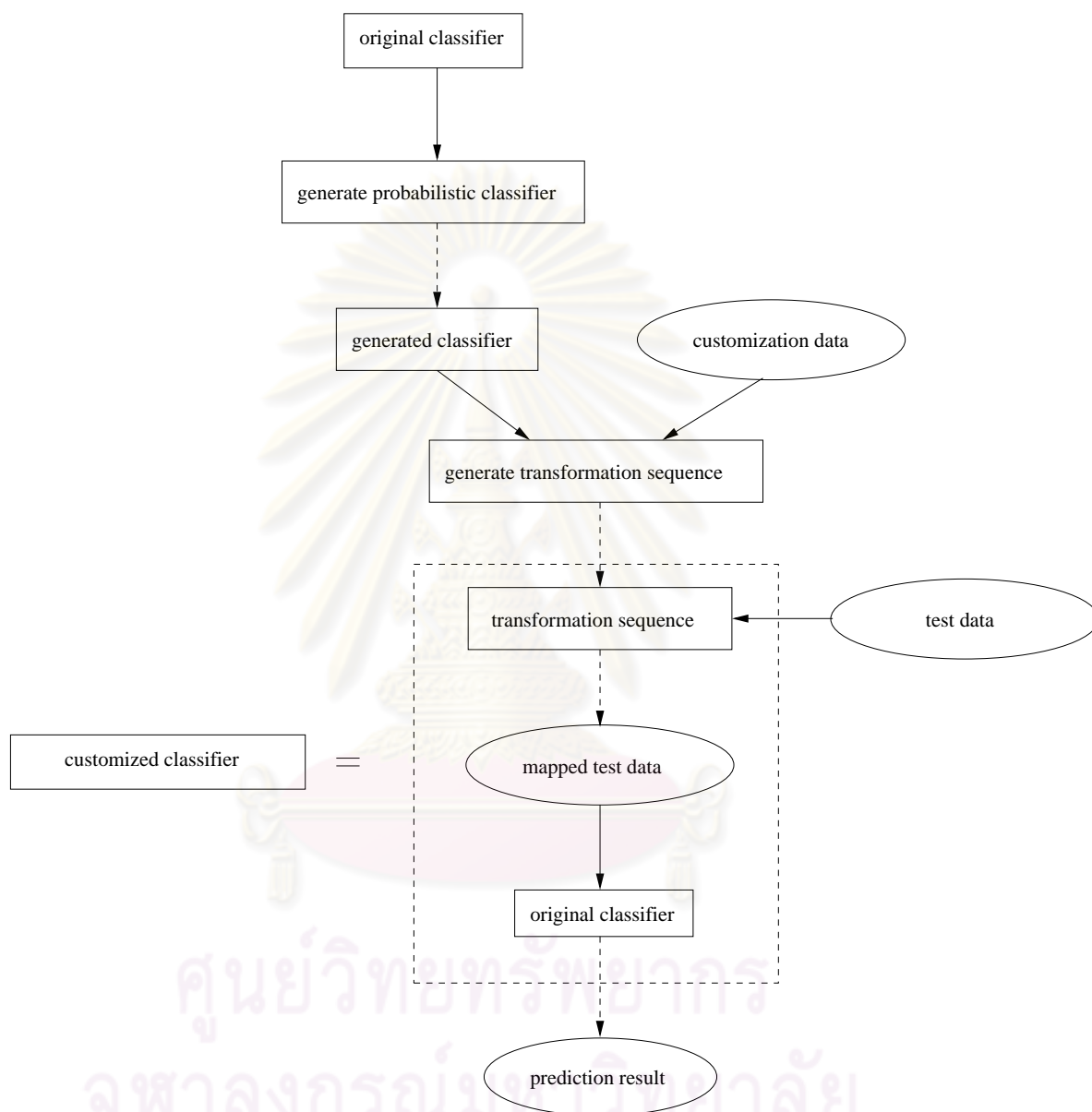


Figure 4.4: Framework of customization for classification by Transforming Input Space.

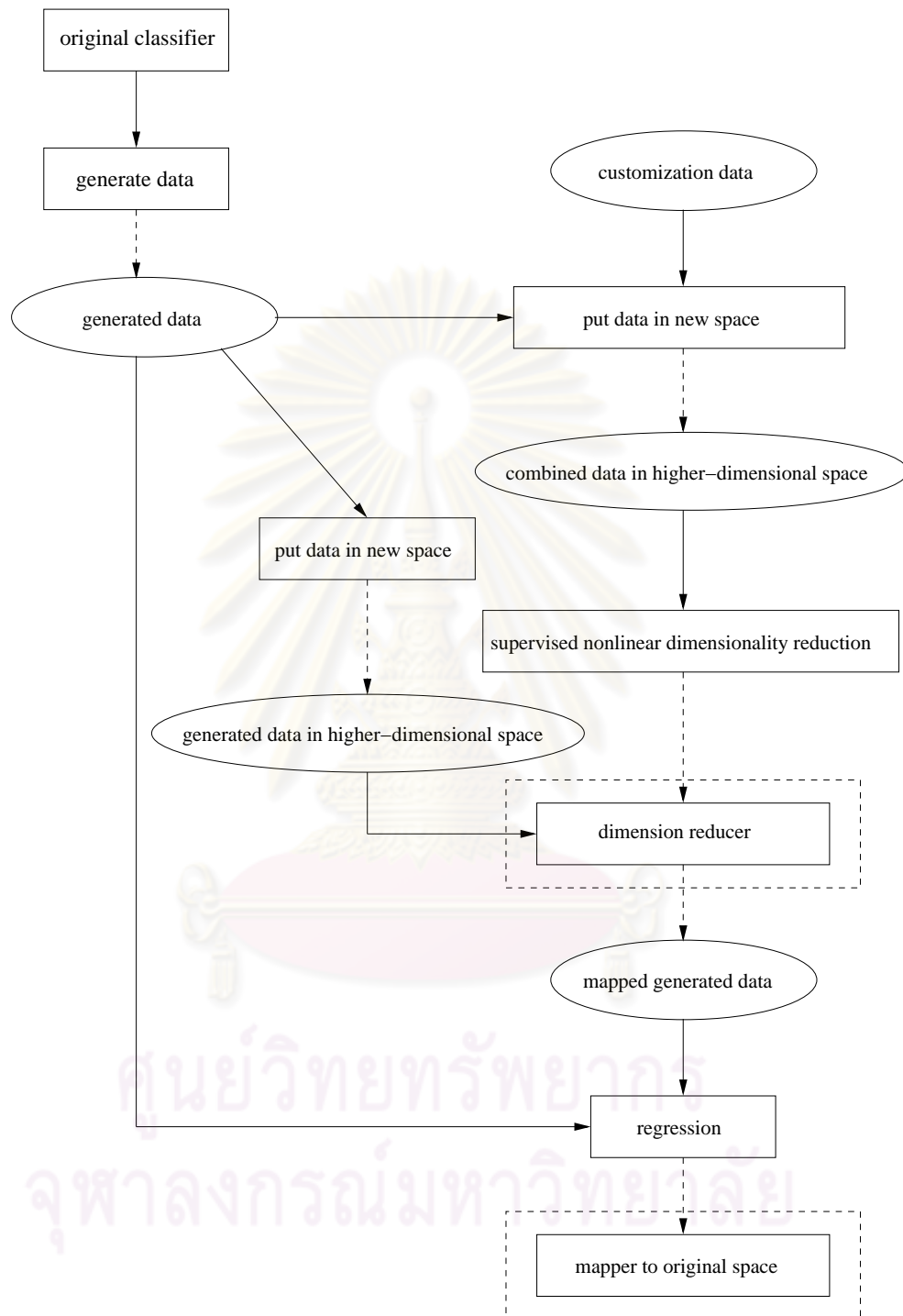


Figure 4.5: Framework of customization for classification with nonlinear dimensionality reduction.

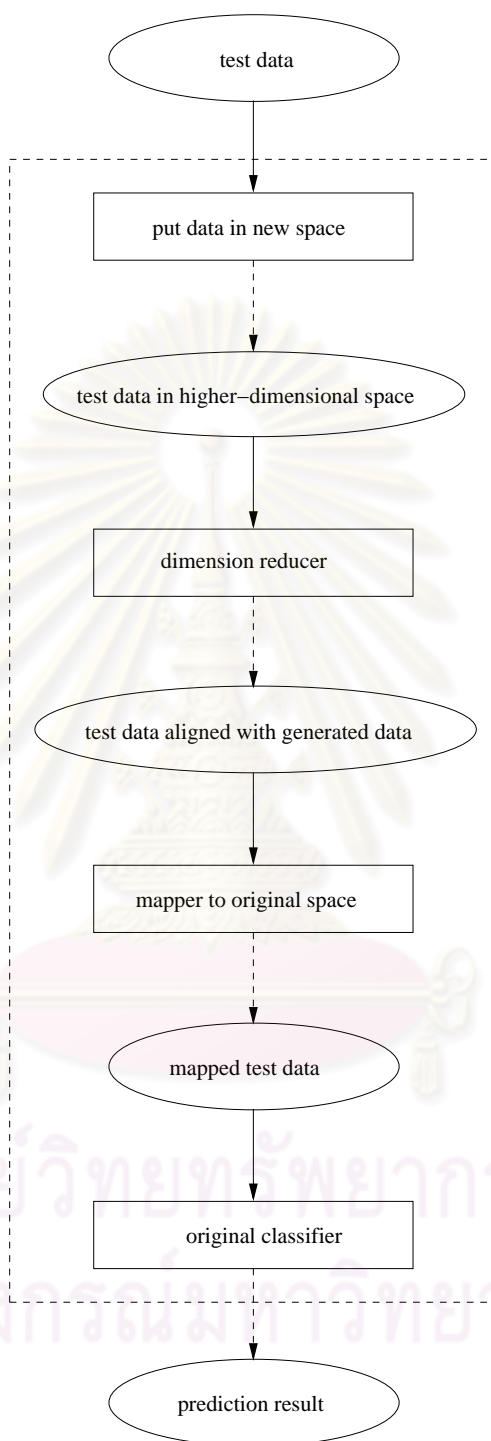


Figure 4.6: New classifier from the framework for customization of classification with nonlinear dimensionality reduction.

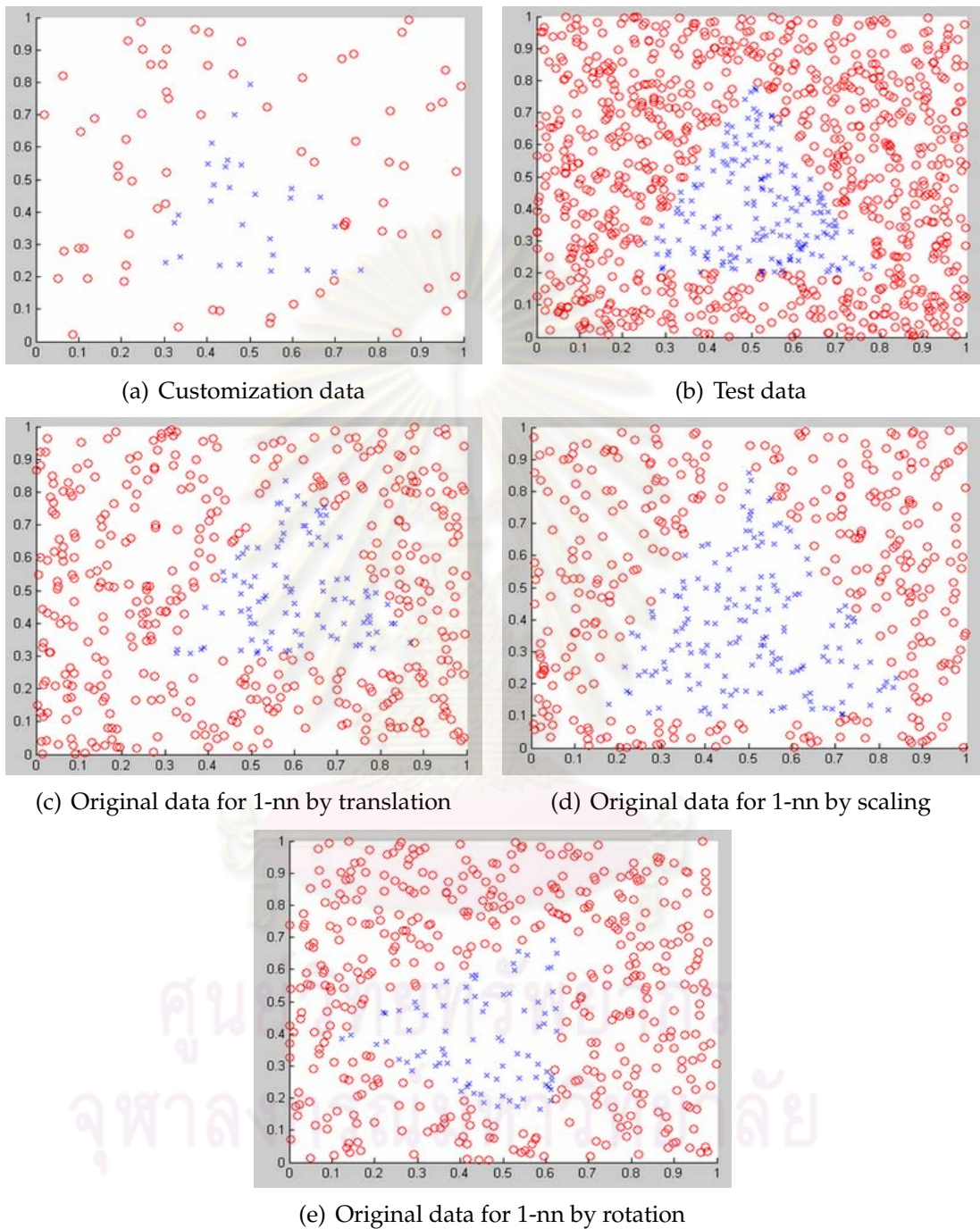


Figure 4.7: Visualization of generated dataset.

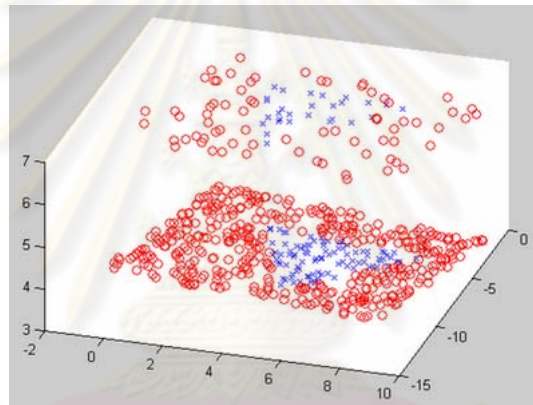


Figure 4.8: Embedding the data into a higher dimension.

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

# CHAPTER V

## CUSTOMIZATION FOR DIMENSIONALITY REDUCTION

In this chapter, we will propose a framework for performing customization for the task of dimensionality reduction. We will also propose an algorithm for dimensionality reduction with additional constraint that the dimensionality reduction is performed with linear function; it is linear dimensionality reduction. The additional constraint in this case reduces the generality for customization to a subgroup of dimensionality reduction. However, the group of algorithms that perform linear dimensionality reduction is still large enough to have considerable impact. With this limitation, we can provide theoretical result up to much more extent than performing it on the much larger set that includes nonlinear dimensionality reduction.

### 5.1 Framework of Customization for Dimensionality Reduction

The goal of dimensionality reduction is finding a mapping function  $\mu : \mathbb{R}^n \rightarrow \mathbb{R}^m$  that will retain the most usefulness of the data, where the definition of usefulness can be different for each algorithm by the objective and constraint on  $\mu$ . In the most general case where there is no constraint on function  $\mu$ , this problem can be solved by training another dimensionality reduction  $\nu : \mathbb{R}^n \rightarrow \mathbb{R}^m$  based on the customization data and then combining the result when the set of customization data is transformed by  $\mu$  and  $\nu$ .

There seem to be no other feasible approaches for customization on dimensionality reduction, due to its nature that is different from classification and regression. The goal for dimensionality reduction is that the resulted data must retain the most useful information, and by any definition of the usefulness, it cannot be measured with just a sole data,  $\mathbf{k}$ . This is because the usefulness cannot be determined by just its own nonmapped quality such as class for classification or continuous value for regression that can be used to determine the objective, but it must be measured by the information of values of many data



in a dataset. This is completely different from classification and regression that we want the result from the process to be as close as possible to another value associated with the data.

Suppose that  $\mathbf{k}$  is transformed by  $\mu$  and  $\nu$  to be  $\mu(\mathbf{k})$  and  $\nu(\mathbf{k})$  respectively, and that the values of  $\mu(\mathbf{k})$  and  $\nu(\mathbf{k})$  are dependent on each other up to some degree since they are both generated from  $\mathbf{k}$  by the procedure. Let us define  $\rho$  as a function which  $\rho(\mathbf{k}) = [\mu(\mathbf{k})^T \nu(\mathbf{k})^T]^T$ . If we perform unsupervised dimensionality reduction on the dataset that is mapped into  $\rho(\mathbf{k})$  then it will result in the mapping onto a coordinate of a space that will try to align the value of  $\mu$  and  $\nu$  on each other based on the dataset. Thus we can find the most feasible mapping using a dataset  $X$  by using all the data used in generating the original dimension reducer and customization data. However, since we might not have the data used in generating the original dimension reducer by the setting, we would have to compromise with just using the customization data to represent the data distribution in the space.

The process of combining two results from dimensionality reduction can be thought of as performing unsupervised dimensionality reduction on the data resulted from both dimensionality reduction. The data resulted from  $\mathbf{k}$  which will be used in the unsupervised dimensionality reduction is written as  $[\mu(\mathbf{k})^T \nu(\mathbf{k})^T]^T$ . Following this procedure, we give the framework for customization for dimensionality reduction as shown in Figure 5.1 and the resulted new dimension reducer is as shown in Figure 5.2.

## 5.2 Combining Results from Linear Dimensionality Reduction

### 5.2.1 Algorithm

For the most general formulation of linear dimensionality reduction with  $n$  attributes, we can write the linear transformation (Wylie and Barrett, 1982; Nicholson, 2001) as  $f(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$  where  $A$  is an orthogonal matrix of size  $n$  whose rows are sorted in the order of importance and  $\mathbf{b}$  is a vector of size  $n$ . Given a training procedure for linear dimensionality reduction  $f(\mathbf{x})$ , one can determine all the  $n^2 + n$  parameters of  $f(\mathbf{x})$  within  $A$  and  $\mathbf{b}$  by observing the

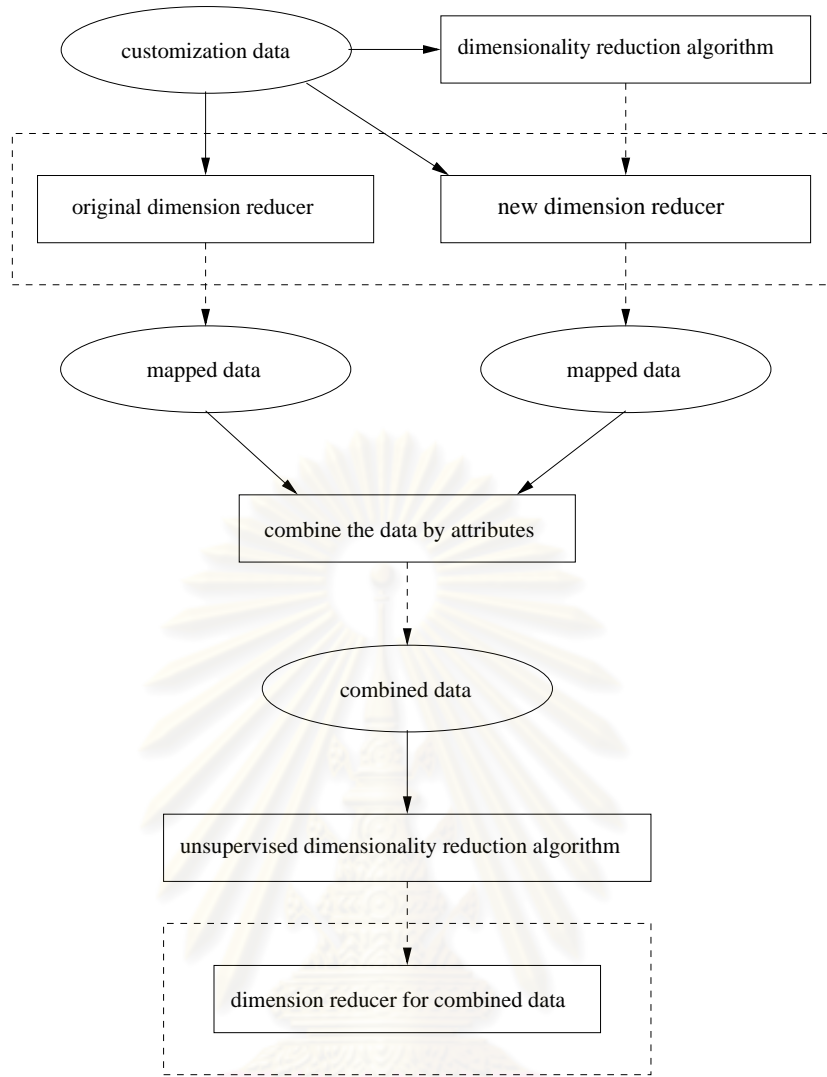


Figure 5.1: Framework for customization for dimensionality reduction.

output given to a set of  $(n + 1)$  vectors. The simplest of them is a normalized vector in each dimension and the zero vector, which results in the following:

$$f(I_n) = AI_n + \mathbf{b}\mathbf{1}_n^T = A + \mathbf{b}\mathbf{1}_n^T,$$

$$f(\mathbf{0}_n) = A\mathbf{0}_n + \mathbf{b} = \mathbf{b},$$

$$A = f(I_n) - \mathbf{b}\mathbf{1}_n^T$$

$$= f(I_n) - f(\mathbf{0}_n)\mathbf{1}_n^T.$$

The customization data will be used to train another linear dimension reducer with the same objective as the original given procedure, resulting in another transformation matrix. We will use  $M^T$  and  $N^T$  to denote the transformation

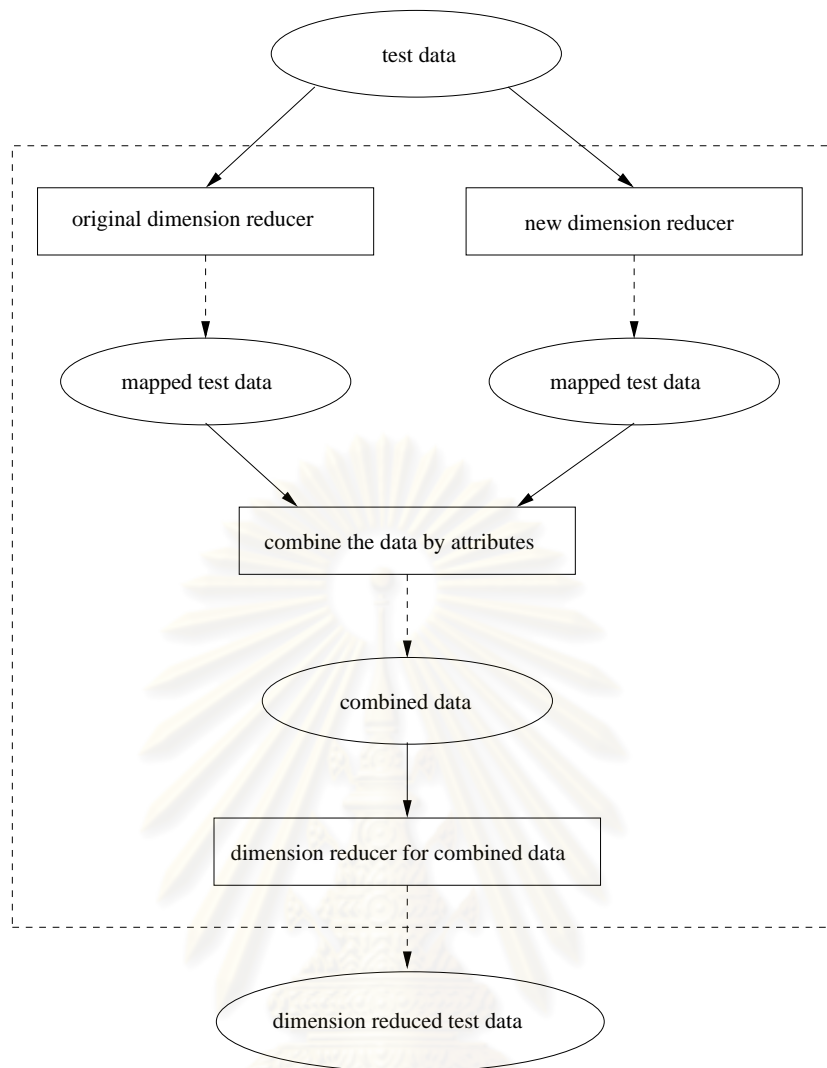


Figure 5.2: The new dimension reducer as the result of using the framework.

matrix belonging to the original procedure and the transformation matrix created from customization data, respectively.

If we want to perform dimensionality reduction of a dataset with  $n$  dimensions to a space with  $m$  dimensions, the most idealistic way will be to combine  $m$  most important linear spaces resulted from both procedure. The basis for the most important linear space with  $m$  dimensions of a linear transformation can be represented with the first  $m$  columns of  $M$  where each of the columns is a basis vector. For combining two such spaces of the same dimension into a new linear space with the same dimension, we propose considering covariance from all the basis vectors of both spaces, fixing the mean to be at origin. The later process will be to find the  $m$  most suitable basis vectors for a new

$m$ -dimensional linear space that gives the greatest variance. This process can be easily described as using the  $m$  most important orthonormal basis vectors of both transformation matrices as data to perform PCA, with the mean fixing to be the zero vector. With this interpretation, the procedure will yield the same result as using each normal basis vector along with the unit vector in the opposite direction which will have in the mean of the all the vectors to be zero, and have a covariance matrix with twice the value of covariance matrix of basis vectors with zero mean.

In the case that we want to give a certain weight  $\sqrt{\alpha}$  and  $\sqrt{1-\alpha}$  to  $M$  and  $N$  respectively, we can perform scaling on the basis vectors from each matrix with that weight.

**Proposition 2.** *Performing PCA on the data which are any orthonormal basis vectors that span the same linear space as  $M$  and  $N$ , with the mean fixed to be zero vector, will always yield the same result.*

*Proof.* PCA is dependent on calculation of covariance matrix  $C$ , which follows the formula  $C = DD^T$  when  $D$  is the matrix with each column as a data that is translated to have zero mean. From the view of each element in the matrix, this formula can alternatively be written as,

$$C_{ij} = \sum_k D_{ik}D_{jk},$$

when  $C_{ij}$  and  $D_{ij}$  are the values of the matrix in row  $i$  and column  $j$  respectively. If we view it from the perspective of each data,  $C$  could be written as follows,

$$C = \sum_k \mathbf{D}_k \mathbf{D}_k^T,$$

where  $\mathbf{D}_k$  is the  $k^{\text{th}}$  row of  $D$ .

The last equation implies that covariance matrix of a dataset is the summation of covariance matrix for each data, using the same mean. In matrix form, this can be written as:

$$C = [M|N][M|N]^T = MM^T + NN^T.$$

Any set of orthonormal basis vectors that represent the same  $m$ -dimensional linear space as a set of orthonormal basis vectors  $D$  can be written as  $DO$  where  $O$  is an orthogonal matrix with  $m$  rows. This can be visualized as  $O$  represents all possible sets of orthonormal basis vectors in  $m$  dimensions, and  $D$  is the transformation that aligns the sets of basis vectors to the linear subspace. As shown in the equation, covariance matrix  $C$  of a matrix with the data as orthonormal basis of two linear subspaces weighted by  $\sqrt{\alpha}$  and  $\sqrt{1-\alpha}$  can be calculated as:

$$\begin{aligned}
 C &= [\sqrt{\alpha}MO_M|\sqrt{1-\alpha}NO_N][\sqrt{\alpha}MO_M|\sqrt{1-\alpha}NO_N]^T \\
 &= (\sqrt{\alpha}MO_M)(\sqrt{\alpha}MO_M)^T + (\sqrt{1-\alpha}NO_N)(\sqrt{1-\alpha}NO_N)^T \\
 &= \alpha(MO_M)(MO_M)^T + (1-\alpha)(NO_N)(NO_N)^T \\
 &= \alpha MO_M O_M^T M^T + (1-\alpha) NO_N O_N^T N^T \\
 &= \alpha M I_m M^T + (1-\alpha) N I_m N^T \\
 &= \alpha M M^T + (1-\alpha) N N^T.
 \end{aligned}$$

Thus we will obtain the same covariance matrix from eigenvalue decomposition which will yield the same transformation matrix. ■

An important note is that the resulted basis vectors from the linear transformation will not be ordered by the importance in each dimension for the data according to the algorithm for linear dimensionality reduction, but with the probability that the dimension should be selected as the result from combination of the two linear transformations. So if the user want to sort the importance of each axis in the new linear space as in the capability of the linear dimensional reduction algorithm, he must perform the algorithm for linear dimensionality reduction on the customization data after they are transformed by the new linear transformation.

Another note is that this process can also be viewed as combining the unimportant linear space, as they will yield the same result. The importance of this aspect is that since the time taken in calculation of PCA also depends on the number of data, it will be faster to do this if the resulted dimension is more

than half of the original, i.e.  $2m > n$ .

**Proposition 3.** *Inverse view of combining linear space with low importance will yield the same result as combining linear space with high importance.*

*Proof.* Let  $M = [M_1|M_2]$  and  $N = [N_1|N_2]$  be both orthonormal vectors which fully span the space where  $M_1$  and  $N_1$  span an  $m$ -dimensional linear space. We want to prove that the  $m$ -dimensional linear space resulted from PCA of  $[\sqrt{\alpha}M_1|\sqrt{1-\alpha}N_1]$  with zero mean is the same to the least important  $m$ -dimensional linear space from PCA of  $[\sqrt{\alpha}M_2|\sqrt{1-\alpha}N_2]$ .

Let  $C_1$  and  $C_2$  be covariance matrices of  $[\sqrt{\alpha}M_1|\sqrt{1-\alpha}N_1]$  and  $[\sqrt{\alpha}M_2|\sqrt{1-\alpha}N_2]$  respectively.

$$\begin{aligned} C_1 &= [\sqrt{\alpha}M_1|\sqrt{1-\alpha}N_1][\sqrt{\alpha}M_1|\sqrt{1-\alpha}N_1]^T \\ &= \alpha M_1 M_1^T + (1-\alpha) N_1 N_1^T \\ C_2 &= \alpha M_2 M_2^T + (1-\alpha) N_2 N_2^T. \end{aligned}$$

Since  $M$  is an orthogonal matrix,

$$\begin{aligned} MM^T &= I_n = [M_1|M_2][M_1|M_2]^T = M_1 M_1^T + M_2 M_2^T, \\ M_2 M_2^T &= I_n - M_1 M_1^T. \end{aligned}$$

Let  $C_1^T = U\Lambda U^T$  be the result from eigenvalue decomposition of the covariance matrix.

$$\begin{aligned} C_2 &= \alpha M_2 M_2^T + (1-\alpha) N_2 N_2^T \\ &= \alpha(I - M_1 M_1^T) + (1-\alpha)(I - N_1 N_1^T) \\ &= I - \alpha M_1 M_1^T - (1-\alpha) N_1 N_1^T \\ &= I - (\alpha M_1 M_1^T + (1-\alpha) N_1 N_1^T) \\ &= I - C_1 \\ &= I - U\Lambda U^T \\ &= U(I - \Lambda)U^T. \end{aligned}$$

Since  $C_2$  is summation of two positive semidefinite matrices,  $C_2$  is a positive

semidefinite matrix as well. The result is that eigenvalues of  $C_2$  which are the diagonal elements of  $(I - \Lambda)$  are nonnegative and are sorted in reverse order to eigenvalues of  $C_1$ . Since eigenvectors of both  $C_1$  and  $C_2$  are the same but in reverse order, this proves that the resulted linear space spanned by the  $m$  most importance eigenvectors of  $C_1$  is the same to the linear space spanned by  $m$  least importance eigenvector of  $C_2$ . ■

We will let both spaces be equally important by setting  $\alpha = 0.5$ . This will yield the same result as letting both weighting terms  $\alpha$  and  $1 - \alpha$  be 1 which will simplify the calculation. This setting will be used for all the following proofs in this section.

### 5.2.2 Theoretical Results

**Proposition 4.** *We can fully describe a linear space spanned by the set of orthonormal basis vectors  $M$  with  $MM^T$ .*

*Proof.* First, we will prove that for any orthogonal matrix  $O$  with  $m$  rows, the space spanned by any  $MO$  which is the same space will result in the same value.

$$\begin{aligned} MO(MO)^T &= MOO^T M^T \\ &= MI_m M^T \\ &= MM^T. \end{aligned}$$

For the inverse, since  $MM^T$  is both symmetric and positive semidefinite, we can perform eigenvalue decomposition on  $MM^T$ . If the space spanned by  $M$  is an  $m$ -dimensional linear space then rank of  $M$  will be  $m$ , and as the result the rank of  $MM^T$  will be  $m$  as well since it is a product of two matrices with rank  $m$ . Other than that, all of its eigenvalues will be 1 as well. Therefore, by performing eigenvalue decomposition and eliminating all the elements with 0 in the eigenvalues, we will get the following,

$$\begin{aligned} MM^T &= NI_m N^T, \\ &= NN^T, \end{aligned}$$

where  $I_m$  is an identity matrix with  $m$  rows and  $N$  is another orthonormal basis vectors with the equal number of rows and columns to  $M$ . Hence, from the equation,

$$\begin{aligned} MM^T &= NN^T, \\ I_m &= M^T NN^T M, \\ &= M^T N(M^T N)^T, \end{aligned}$$

and thus  $M^T N$  is an orthogonal matrix which transforms  $M$  into  $N$  and that both  $M$  and  $N$  span the same linear space. ■

From the result of Proposition 4, we will now refer to a linear space spanned by a set of orthonormal basis vectors  $M$  with the matrix  $M_S = MM^T$ .

**Proposition 5.** *Let  $\mathbf{k}$  be a unit vector and  $X_S$  be a linear space,  $X_S \mathbf{k}$  is the projection of  $\mathbf{k}$  on the space  $X_S$ .*

*Proof.* Let  $X$  be a set of orthonormal basis of  $X_S$  that is  $X_S = XX^T$ .  $X^T \mathbf{k}$  is the projection of  $\mathbf{k}$  onto each of the orthonormal basis of space  $X_S$ .  $X(X^T \mathbf{k})$  can be viewed as multiplying each of the orthonormal basis vectors of  $X_S$  with the value that  $\mathbf{k}$  is projected on them. Hence,  $X_S \mathbf{k}$  is the projection of  $\mathbf{k}$  on the space  $X_S$ . ■

**Definition 1.** *Let  $A, B, C$  and  $D$  be the sets of orthonormal basis vectors which span an  $m$ -dimensional linear space. We say that the difference between the space spanned by  $A$  and  $B$  is equal to the difference between the space spanned by  $C$  and  $D$  if and only if there exists an orthogonal matrix  $O$  which causes the space spanned by  $AO$  and  $BO$  be equivalent to the space spanned by  $C$  and  $D$  respectively.*

Definition 1 arises from the property that the operation of an orthogonal matrix will be equivalent to a sequence of rotation and reflection centered at the origin, which will preserve the distance between any vectors in the space. We will have another constraint for a function which measures difference between two linear spaces.



**Definition 2.** Let  $M$  and  $N$  be two sets of orthonormal basis vectors. A function  $f(M, N)$  is a measure of difference between the spaces spanned by both  $M$  and  $N$  if and only if,

$$i) f(M, N) = f(N, M),$$

$$ii) f(M, N) = f(OM, ON),$$

$$iii) f(M, N) = f(MO_M, NO_N),$$

where  $O, O_M$  and  $O_N$  are any orthogonal matrices.

In Definition 2, the first condition is from that the function must be reflexive since it is a measure of difference. The second condition is from Definition 1. The third condition is from that the function should yield the same output for any orthonormal basis vectors that span the same space.

**Proposition 6.** Let  $M$  and  $N$  be two sets of orthonormal basis vectors which span an  $m$ -dimensional linear space within an  $n$ -dimensional space. We can measure the difference between them with the function  $f(M, N) = |\det(M^T N)|$

*Proof.* We will prove that the function satisfies every condition in Definition 2.

For the first condition, by the property of determinant that determinant of a matrix yields the same value to determinant of its transposed matrix, we can conclude that the function is reflexive.

$$f(M, N) = |\det(M^T N)| = |\det((M^T N)^T)| = |\det(N^T M)| = f(N, M).$$

For the second condition, we will prove that if the differences within two sets of linear spaces are equivalent then the function will provide the same value. Suppose that  $M$  and  $N$  be two sets of orthonormal basis vectors and  $O$  be an

orthogonal matrix.

$$\begin{aligned}
 f(OM, ON) &= |\det((OM)^T ON)| \\
 &= |\det(M^T O^T ON)| \\
 &= |\det(M^T N)| \\
 &= f(M, N).
 \end{aligned}$$

For the third condition, we will prove that any basis vectors which span the same space will yield the same result. Let  $O_M$  and  $O_N$  be orthogonal matrices with  $m$  rows. All the set of basis vectors which span the same space as  $M$  and  $N$  can be written as  $MO_M$  and  $NO_N$ .

$$\begin{aligned}
 f(MO_M, NO_N) &= |\det((MO_M)^T NO_N)| \\
 &= |\det(O_M^T M^T NO_N)| \\
 &= |\det(O_M^T) \det(M^T N) \det(O_N)| \\
 &= |\det(O_M)| |\det(M^T N)| |\det(O_N)| \\
 &= |\det(M^T N)| \\
 &= f(M, N).
 \end{aligned}$$

This results from the property of multiplicative distribution of determinant and from the fact that the determinant of an orthogonal matrix is  $\pm 1$  from its definition that an orthogonal matrix multiplying to its transposed matrix will result in an identity matrix. ■

To get a better understanding of the function in Proposition 6, we first consider the case that  $m = 1$ . The function  $|\det(M^T N)|$  will become the absolute value of dot product between two vectors. Its absolute value is the result from that the spanned linear space is in both positive and negative directions of the basis vectors.

Furthermore, we will consider  $\min(\text{eig}(N^T M_S N))$  where  $\text{eig}(X)$  is the function that returns all of the eigenvalues of  $X$ . The minimum eigenvalue will reflect the result of using  $\mathbf{k}$ , a vector in  $N_S$ , which minimizes the function

$\mathbf{k}^T M_S \mathbf{k}$  for all the vectors in  $N_S$ . The function can be interpreted as projecting  $\mathbf{k}$  onto  $M_S$  and projecting it back onto  $N_S$ . Hence, the eigenvalue will be  $\cos^2(\alpha)$  when  $\alpha$  is the angle between  $\mathbf{k}$  and  $M_S$  which also serves as the angular difference between  $N_S$  and  $M_S$ .

**Proposition 7.** *Let  $M$  and  $N$  be two sets of orthonormal basis vectors which span an  $m$ -dimensional linear space. The function  $f(M, N) = |\det(M^T N)|$  will have its maximum value if and only if  $M$  and  $N$  span the same linear space.*

*Proof.* We will state a proof that both sets of orthonormal basis vectors will span the same space if and only if the value of the function is 1. The sufficient condition uses the knowledge that has been previously applied.

$$\begin{aligned} f(M, M O_M) &= |\det(M^T M O_M)| \\ &= |\det(M^T M)| |\det(O_M)| \\ &= 1. \end{aligned}$$

For the necessary condition, let us consider the following equation:

$$\begin{aligned} f^2(M, N) &= |\det(M^T N)|^2 \\ &= (\det(M^T N))^2 \\ &= \det(M^T N) \det(M^T N) \\ &= \det(M^T N) \det(N^T M) \\ &= \det(M^T N N^T M) \\ &= \det(M^T N N^T M). \end{aligned}$$

By the property of determinant, its value will be equivalent to the multiplication of all singular values of the matrix. So if there is a pair of  $M$  and  $N$  that cause the value of  $f(M, N) > 1$  then there will be at least an eigenvalue of  $M^T N N^T M$  that is higher than 1 since  $M^T N N^T M$  is a symmetric positive semidefinite matrix. Suppose that  $\mathbf{k}$  is the unit eigenvector with the maximum eigenvalue of

$M^T N N^T M$ , it will follow that the eigenvalue of  $\mathbf{k}$  will be equal to:

$$\begin{aligned} \mathbf{k}^T M^T N N^T M \mathbf{k} &= (M\mathbf{k})^T N N^T (M\mathbf{k}), \\ &= (M\mathbf{k})^T N_S (M\mathbf{k}). \end{aligned}$$

Since  $M$  is an orthonormal basis vector and  $\mathbf{k}$  is a unit vector,  $M\mathbf{k}$  will be a unit vector that lies in the space spanned by  $M$  and  $(M\mathbf{k})^T N_S (M\mathbf{k})$  can be viewed as dot product between the unit vector and the projection of that unit vector onto the space  $N_S$ . Since projection does not increase the size of vector, it is clear that  $(M\mathbf{k})^T N_S (M\mathbf{k}) \leq 1$  and that 1 is the maximum value of  $f(M, N)$  ■

From Proposition 7, the main contribution of the function  $|\det(M, N)|$  is that it can provide an answer if two sets of orthonormal basis vectors span the same space, without requiring any further interpretation. However, difference between two linear spaces cannot be fully described with one real value due to its degree of freedom.

Now, let us consider space  $M_S = M M^T$  and  $N_S = N N^T$ . If the space  $N_S$  is exactly the same as  $M_S$ , then all orthonormal basis vectors of  $N_S$  must be in  $M_S$ . Therefore,

$$N^T M_S N = N^T N = I_m,$$

where  $I_m$  is an identity matrix with  $m$  rows.

The diagonal elements of  $N^T M_S N$  will be 1 if and only if each basis vector of  $N_S$  in  $N$  is in  $M_S$  since  $M_S$  can be interpreted as projection of the vector onto that space. Moreover, if that is the case, then  $N_S$  will span the same space as  $M_S$  and the projection of each different orthonormal basis vector in  $N$  will still be orthogonal to each other. So we can conclude that  $N^T M_S N = I_m$  if and only if  $M_S = N_S$ .

From Definition 2 which defines the constraint for functions that measure the difference between linear spaces from its orthonormal basis vectors. We can derive the constraints for functions that measure the difference between linear spaces using the matrix representing the spaces directly.

**Proposition 8.** Let  $M_S$  and  $N_S$  be two  $m$ -dimensional linear spaces. A function  $g(M_S, N_S)$  is a measure of difference between the spaces if and only if,

$$i) \quad g(M_S, N_S) = g(N_S, M_S),$$

$$ii) \quad g(M_S, N_S) = g(OM_S O^T, ON_S O^T),$$

where  $O$  is any orthogonal matrix with  $n$  rows.

*Proof.* From Definition 2, we can obviously see that  $g(AA^T, BB^T) = g(CC^T, DD^T)$  if and only if  $f(A, B) = f(C, D)$ .

For the first condition, assume the first condition in Definition 2.

$$\begin{aligned} f(M, N) &= f(N, M) \\ g(MM^T, NN^T) &= g(NN^T, MM^T) \\ g(M_S^T, N_S^T) &= g(N_S^T, M_S^T). \end{aligned}$$

For the second condition, assume the second condition in Definition 2.

$$\begin{aligned} f(M, N) &= f(OM, ON) \\ g(MM^T, NN^T) &= g(OM(OM)^T, ON(ON)^T) \\ g(MM^T, NN^T) &= g(OMM^T O^T, ONN^T O^T) \\ g(M_S, N_S) &= g(OM_S O^T, ON_S O^T). \end{aligned}$$

Assuming the third condition in Definition 2.

$$\begin{aligned} f(M, N) &= f(MO_M, NO_N) \\ g(MM^T, NN^T) &= g(MO_M(MO_M)^T, NO_N(NO_N)^T) \\ g(MM^T, NN^T) &= g(MO_M O_M^T M^T, NO_N O_N^T N^T) \\ g(MM^T, NN^T) &= g(MM^T, NN^T) \\ g(M_S, N_S) &= g(M_S, N_S). \end{aligned}$$

Since the third condition from Definition 2 is always true by the definition of the linear space, it needs no further consideration. ■

**Proposition 9.** *We can measure difference between two linear spaces  $M_S$  and  $N_S$  with  $g(M_S, N_S) = eig(M_S + N_S)$  where  $eig(X)$  is the sorted eigenvalues of matrix  $X$ .*

*Proof.* We will prove that the function  $g(M_S, N_S) = eig(M_S + N_S)$  satisfies all the conditions in Proposition 8.

For the first condition, since matrix addition is associative, this is really obvious.

$$g(M_S, N_S) = eig(M_S + N_S) = eig(N_S + M_S) = g(N_S, M_S)$$

For the second condition,

$$\begin{aligned} g(OM_S O^T, ON_S O^T) &= eig(OM_S O^T + ON_S O^T), \\ &= eig(O(M_S + N_S)O^T). \end{aligned}$$

From the eigenvalue decomposition, let  $M_S + N_S = V\Lambda V^T$ .

$$\begin{aligned} g(OM_S O^T, ON_S O^T) &= eig(O(M_S + N_S)O^T) \\ &= eig(OV\Lambda V^T O^T) \\ &= eig((OV)\Lambda(OV)^T) \\ &= eig(\Lambda). \end{aligned}$$

The last line of the equation can be considered as extracting the eigenvalues from the matrix. In the same way,

$$\begin{aligned} g(M_S, N_S) &= eig(M_S + N_S), \\ &= eig(V\Lambda V^T), \\ &= eig(\Lambda), \\ &= g(OM_S O^T, ON_S O^T). \end{aligned}$$

■

Let us further analyse the function in Proposition 9. It is quite obvious that the function is better as a measurement of difference between two linear spaces than the function in Proposition 6, due to its output which has  $n$  degree of freedom when  $n$  is the number of dimensions. For the case that both spaces are the same:

$$\begin{aligned} \text{eig}(M_S + M_S) &= \text{eig}(2M_S), \\ &= 2\text{eig}(M_S), \end{aligned}$$

which means that it will have  $m$  eigenvalues which are 2 and the rest of  $n - m$  eigenvalues from calculation will be zero.

**Proposition 10.** *Let  $M_S$  and  $N_S$  be two  $m$ -dimensional linear spaces. We can fully describe the difference between  $M_S$  and  $N_S$  with the matrix  $D_{MN} = U^T M_S^T U$  when  $U^T$  is the eigenvectors of  $N_S$  sorted by its eigenvalues.*

*Proof.* Since the number of dimensions of  $M_S$  and  $N_S$  are equal, they will have same set of eigenvalues. As a result of eigenvalue decomposition, we have:

$$\begin{aligned} M_S &= V\Lambda V^T, \\ N_S &= U\Lambda U^T, \end{aligned}$$

where  $V$  and  $U$  are orthogonal matrices. From Proposition 8 the difference between  $M_S$  and  $N_S$  will be equal to the difference between  $\Lambda$  and  $U^T M_S U$ . Since  $\Lambda$  is also the eigenvalue part from eigenvalue decomposition of  $U^T M_S U$ . It follows that the difference between  $M_S$  and  $N_S$  can be fully described with the matrix  $D_{MN} = U^T M_S U$ .

Also, a function  $h(D)$  can completely measure the difference from the difference matrix  $D$  with  $h(D) = g(\text{eig}(D), D)$ . By Proposition 8, we will have the following constraint:  $h(D) = h(O D O^T)$  if and only if  $O \text{eig}(D) O^T = \text{eig}(D)$ . ■

From Proposition 10, the first note is that  $D_{MN} = U^T M_S U$  and  $D_{NM} = V^T N_S V$  are not necessarily the same matrix but they both fully describe the differences which are equivalent to each other.

The constraint in Proposition 10 gives us some guideline in performing pairwise comparison of the difference. However, there is the following problems that given two symmetric positive semidefinite matrices  $A$  and  $B$ , how to efficiently compute an orthogonal matrix  $O$  such that  $A = OBO^T$ , and how to compute  $h(D)$  that will yield the same output if and only if the difference are equivalent.

**Theorem 1.** *Let  $M_S$  and  $N_S$  be two  $m$ -dimensional linear spaces, the  $m$ -dimensional linear space  $X_S$  which is spanned by the eigenvectors with  $m$  highest eigenvalues of  $M_S + N_S$  is the optimal mean of  $M_S$  and  $N_S$  in the sense that it will minimize the value  $\max_{\mathbf{k}^T M_S \mathbf{k} = 1 \text{ or } \mathbf{k}^T N_S \mathbf{k} = 1} \mathbf{k}^T Y_S \mathbf{k}$  when  $Y_S$  is any  $m$  dimensional linear space, meaning that it will minimize the angular difference between any vectors in  $M_S$  or  $N_S$  and the new linear space.*

*Proof.* From Proposition 3, it is enough to prove the result when  $m$  is not greater than  $n/2$ , since performing the process on the none important space will yield the same result. From this, we will be able to use the assumption that  $M_S$  and  $N_S$  do not intersect each other at any place other than the origin.

Let  $X_S$  be an  $m$ -dimensional linear space that serves as a mirror plane which transforms the space of  $M_S$  into  $N_S$  back and forth; i.e. for a unit vector  $\mathbf{k}$  in  $M_S$  or  $N_S$ , there will be another unit vector  $\mathbf{j}$  in the other linear space such that the result from both of their projection on  $X_S$  will be exactly the same but has the exactly opposite direction of projection. This can be visualized that the linear space  $X_S$  will act as a mirror between  $M_S$  and  $N_S$ . If  $\mathbf{k}$  and  $\mathbf{j}$  are both unit vectors that yield the value of angular difference between  $M_S$  and  $N_S$  then it can be seen that the angular difference between  $X_S$  and  $\mathbf{k}$  will be equivalent to the angular difference between  $X_S$  and  $\mathbf{j}$  since the projection from both of them on  $X_S$  will be at the same place. It becomes obvious that the angular difference between other pair of unit vectors in  $M_S$  and  $N_S$  will be lower than this value and thus this  $X_S$  is an optimal mean space calculated from  $M_S$  and  $N_S$  in the sense of angular difference.

Let  $X_S$ ,  $M_S$  and  $N_S$  be created from their orthonormal bases  $X$ ,  $M$  and  $N$ . From the previous definition of  $X_S$ ,  $M_S$  and  $N_S$  as in the previous paragraph,



and let  $\mathbf{k}$  be a vector in  $M_S$ , we can calculate  $\mathbf{j}$ , a vector in  $N_S$  corresponding to  $\mathbf{k}$ , which will result in the following equation:

$$\begin{aligned}\mathbf{k} &= X_S \mathbf{k} + (\mathbf{k} - X_S \mathbf{k}), \\ \mathbf{j} &= X_S \mathbf{k} - (\mathbf{k} - X_S \mathbf{k}), \\ &= 2X_S \mathbf{k} - \mathbf{k}, \\ &= (2X_S - I_n) \mathbf{k}.\end{aligned}$$

This relies on the fact that  $M_S$ ,  $N_S$  and  $X_S$  have the same number of dimensions and do not intersect each other. That is why for a vector in  $M_S$ , there should be a vector in  $N_S$  which has equal angular difference from  $X_S$  and will be projected on  $X_S$  at the same position but with exactly opposite direction of the projection. That is why we can write the space  $N_S$  as following:

$$\begin{aligned}N_S &= NN^T, \\ &= (2X_S - I_n)M((2X_S - I_n)M)^T, \\ &= (2X_S - I_n)MM^T(2X_S - I_n)^T, \\ &= (2X_S - I_n)M_S(2X_S - I_n).\end{aligned}$$

Now let us consider the matrix  $M_S + N_S$ .

$$\begin{aligned}M_S + N_S &= M_S + (2X_S - I_n)M_S(2X_S - I_n) \\ &= M_S + (4X_S M_S X_S - 2X_S M_S - 2M_S X_S + M_S) \\ &= 4X_S M_S X_S - 2X_S M_S - 2M_S X_S + 2M_S.\end{aligned}$$

Let  $\mathbf{k}$  be an eigenvector of  $M_S + N_S$ . We will observe the result when separating

$\mathbf{k}$  into its elements that are in  $X_S$  and that are not.

$$\begin{aligned}
\mathbf{k} &= X_S \mathbf{k} + (I_n - X_S) \mathbf{k} \\
(M_S + N_S) \mathbf{k} &= (4X_S M_S X_S - 2X_S M_S - 2M_S X_S + 2M_S)(X_S \mathbf{k} + (I_n - X_S) \mathbf{k}) \\
&= ((4X_S M_S X_S - 2X_S M_S - 2M_S X_S + 2M_S) X_S \mathbf{k}) \\
&\quad + ((4X_S M_S X_S - 2X_S M_S - 2M_S X_S + 2M_S)(I_n - X_S) \mathbf{k}) \\
&= ((4X_S M_S - 2X_S M_S - 2M_S + 2M_S) X_S \mathbf{k}) \\
&\quad + ((-2X_S M_S + 2M_S)(I_n - X_S) \mathbf{k}) \\
&= 2X_S M_S X_S \mathbf{k} + 2(I_n - X_S) M_S (I_n - X_S) \mathbf{k}.
\end{aligned}$$

Suppose that  $\mathbf{k}$  has elements in both  $X_S$  and in the space orthogonal to  $X_S$ , which is the  $(n - m)$ -dimensional linear space that does not intersect with  $X_S$  except at the origin, denoted by  $X_{S\perp} = I_n - X_S$ . If there is only one optimal  $X_S$  by the definition then the angular difference between  $M_S$  and  $N_S$  must be less than  $\pi/2$  which causes the angular difference between any vectors in  $M_S$  and  $X_S$  to be less than  $\pi/4$  while also causes the angular difference between any vectors in  $M_S$  and  $X_{S\perp}$  to be higher than  $\pi/4$ . For  $\mathbf{k}$  to be an eigenvector of  $M_S + N_S$ ,  $X_S M_S X_S \mathbf{k}$  must be in the same direction as  $X_S \mathbf{k}$  and  $X_{S\perp} M_S X_{S\perp} \mathbf{k}$  must be in the same direction as  $X_{S\perp} \mathbf{k}$ . If we let the angular difference between  $M_S$  and  $X_S \mathbf{k}$  and between  $M_S$  and  $X_{S\perp} \mathbf{k}$  be  $\alpha$  and  $\beta$  respectively then we will get the following equations:

$$\begin{aligned}
(M_S + N_S) \mathbf{k} &= 2X_S M_S X_S \mathbf{k} + 2(I_n - X_S) M_S (I_n - X_S) \mathbf{k} \\
&= 2X_S M_S X_S \mathbf{k} + 2X_{S\perp} M_S X_{S\perp} \mathbf{k} \\
&= 2\cos^2(\alpha) X_S \mathbf{k} + 2\cos^2(\beta) X_{S\perp} \mathbf{k}
\end{aligned}$$

Since  $0 < \alpha < \beta < \pi/2$ , it follows that either  $X_S \mathbf{k}$  or  $X_{S\perp} \mathbf{k}$  must be the zero vector. This means that eigenvectors of  $M_S + N_S$  must lie in either  $X_S$  or  $X_{S\perp}$  and since  $X_S$  is  $m$ -dimensional then there will be  $m$  eigenvectors in  $X_S$  and  $(n - m)$  eigenvectors in  $X_{S\perp}$ .

In the same way as  $M_S$ , angular difference between any vectors in  $X_S$  and  $N_S$  must be less than  $\pi/4$  and angular difference between any vectors in  $X_S$

and  $N_{S^\perp}$  must be higher than  $\pi/4$ . As a result, for any vector  $\mathbf{k}$  in  $X_S$ :

$$\begin{aligned}\mathbf{k}^T(M_S + N_S)\mathbf{k} &= \mathbf{k}^T M_S \mathbf{k} + \mathbf{k}^T N_S \mathbf{k}, \\ &> 2\cos^2(\pi/4), \\ &= 1, \\ &> \mathbf{p}^T(M_S + N_S)\mathbf{p},\end{aligned}$$

where  $\mathbf{p}$  is a vector in  $X_{S^\perp}$ . This means that the  $m$  eigenvectors with highest eigenvalues will all lie in  $X_S$  and are also orthonormal basis vectors. Thus, the linear space resulted from the process is the optimal linear space that minimizes the value of  $\max_{\mathbf{k}^T M_S \mathbf{k}=1 \text{ or } \mathbf{k}^T N_S \mathbf{k}=1} \mathbf{k}^T Y_S \mathbf{k}$  when  $Y_S$  is any  $m$  dimensional linear space. ■

In the case that the user want to weight  $M_S$  and  $N_S$  in a meaningful way, it seems that the most sensible way is to repeat using the algorithm to find the mean of two linear spaces until it reaches the satisfying precision, instead of providing the weight to the linear space directly. The algorithm for the weighted version of both spaces is shown in Algorithm 7.

ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

```

input :  $M, N, m, \alpha$ 
output:  $X$ 
1  $A \leftarrow M, B \leftarrow N, X \leftarrow A, a \leftarrow 0, b \leftarrow 1, x \leftarrow 0;$ 
2 while  $|\alpha - x| > \beta$  do
3    $X \leftarrow PCA(AA^T + BB^T, m);$ 
4    $x \leftarrow (a + b)/2;$ 
5   if  $\alpha > x$  then
6      $a \leftarrow x;$ 
7      $A \leftarrow X;$ 
8   else
9      $b \leftarrow x;$ 
10     $B \leftarrow X;$ 
11  end
12 end

```

**Algorithm 7:** Finding the optimal mean between two linear spaces with weight.

In Algorithm 7,  $PCA(X, m)$  is the result of using PCA to choose  $m$  dimensions from covariance matrix  $A_S$ ,  $\alpha$  is a real value in the range  $[0, 1]$  which is the weight for  $N_S$  while the weight  $M_S$  will be  $1 - \alpha$ ,  $\beta$  is the constraint for precision in calculation for the result, and  $m$  is the dimension for both  $M$  and  $N$ .

Note that in the case that  $M_S$  and  $N_S$  are obtained by the linear dimensionality reduction of the same dataset, but with a different algorithm, the process of combining them will act as finding the linear space that is good in both objectives.

**Theorem 2.** *Using the proposed framework of customization for dimensionality reduction in Figure 5.1 on linear dimensionality reduction will yield the optimal result if we can estimate the distribution of data perfectly.*

*Proof.* Let  $M_S$  and  $N_S$  be both spaces to be combined which have  $M$  and  $N$  as sets of their orthonormal basis vectors. According to the framework, to combine two nonlinear spaces we will have to use the results from mapping customization data on both of them in order to estimate the shape of the nonlinear spaces. However, in the case of linear space, the shape of the space can be fully

described with the matrix and we could estimate the distribution from all the data which are used to generate both spaces by the set of all available data in the spaces. Since the mapping is linear, we can represent the result from mapping the data of all input spaces with identity matrix and its negative. Thus, we get the dataset as  $[M|N]^T[I_n | -I_n]$ . The mean of this dataset is clearly zero due to that the vectors resulted from  $[M|N]^T[I_n]$  will come with their negative vectors  $[M|N]^T[-I_n]$ . The next step is to perform PCA using this dataset to obtain the following covariance matrix.

$$\begin{aligned}
 [M|N]^T[I_n | -I_n]([M|N]^T[I_n | -I_n])^T &= [M|N]^T[I_n | -I_n][I_n | -I_n]^T[M|N] \\
 &= [M|N]^T(I_n I_n^T + (-I_n)(-I_n)^T)[M|N] \\
 &= [M|N]^T(2I_n I_n^T)[M|N] \\
 &= 2[M|N]^T[M|N].
 \end{aligned}$$

Since scaling of a matrix will not affect its eigenvectors, we will instead use the matrix  $[M|N]^T[M|N]$  which can also be calculated as covariance matrix of the dataset  $[M|N]^T$  with its mean fix to zero. By further analysis of the matrix:

$$\begin{aligned}
 [M|N]^T[M|N] &= \begin{bmatrix} M^T \\ N^T \end{bmatrix} \begin{bmatrix} M & N \end{bmatrix}, \\
 &= \begin{bmatrix} M^T M & M^T N \\ N^T M & N^T N \end{bmatrix}, \\
 &= \begin{bmatrix} I_m & M^T N \\ N^T M & I_m \end{bmatrix}.
 \end{aligned}$$

Now, let us consider a vector  $\mathbf{k}$  which is an eigenvector of the matrix  $MM^T + NN^T$  with eigenvalue  $\lambda$ .

$$\begin{aligned}
\begin{bmatrix} I_m & M^T N \\ N^T M & I_m \end{bmatrix} \begin{bmatrix} M^T \\ N^T \end{bmatrix} \mathbf{k} &= \begin{bmatrix} M^T + M^T N N^T \\ N^T M M^T + N^T \end{bmatrix} \mathbf{k} \\
&= \begin{bmatrix} M^T \mathbf{k} + M^T N N^T \mathbf{k} \\ N^T \mathbf{k} + N^T M M^T \mathbf{k} \end{bmatrix} \\
&= \begin{bmatrix} M^T \mathbf{k} + M^T (\lambda \mathbf{k} - M M^T \mathbf{k}) \\ N^T \mathbf{k} + N^T (\lambda \mathbf{k} - N N^T \mathbf{k}) \end{bmatrix} \\
&= \begin{bmatrix} M^T \mathbf{k} + \lambda M^T \mathbf{k} - M^T M M^T \mathbf{k} \\ N^T \mathbf{k} + \lambda N^T \mathbf{k} - N^T N N^T \mathbf{k} \end{bmatrix} \\
&= \begin{bmatrix} M^T \mathbf{k} + \lambda M^T \mathbf{k} - I_m M^T \mathbf{k} \\ N^T \mathbf{k} + \lambda N^T \mathbf{k} - I_m N^T \mathbf{k} \end{bmatrix} \\
&= \begin{bmatrix} M^T \mathbf{k} + \lambda M^T \mathbf{k} - M^T \mathbf{k} \\ N^T \mathbf{k} + \lambda N^T \mathbf{k} - N^T \mathbf{k} \end{bmatrix} \\
&= \begin{bmatrix} \lambda M^T \mathbf{k} \\ \lambda N^T \mathbf{k} \end{bmatrix} \\
&= \lambda \begin{bmatrix} M^T \\ N^T \end{bmatrix} \mathbf{k}.
\end{aligned}$$

From the above equation, if a vector  $\mathbf{k}$  is an eigenvector of  $MM^T + NN^T$  with eigenvalue  $\lambda$  then  $[M|N]^T \mathbf{k}$  will be an eigenvector of  $[M|N]^T [M|N]$  with eigenvalue  $\lambda$ . Since the matrix  $[M|N]^T [M|N]$  is multiplication between  $[M|N]$  and its transpose, the rank of  $[M|N]^T [M|N]$  will be equal to the rank of  $[M|N]$ , and thus its maximum value will be the lesser between its number of rows and columns which are  $n$  and  $2m$  respectively. The result is that in the case that the number of rows of  $[M|N]^T [M|N]$  is higher than  $n$  the other eigenvectors which are not corresponding to any  $[M|N]^T \mathbf{k}$  when  $\mathbf{k}$  is an eigenvector of  $MM^T + NN^T$  will have zero eigenvalue. One may say that the matrix will have no other eigenvector. So a vector  $\mathbf{k}$  will be the eigenvector of  $MM^T + NN^T$  with the  $i$ -th maximum eigenvalue if and only if  $[M|N]^T \mathbf{k}$  is the eigenvector of  $[M|N]^T [M|N]$  with the  $i$ -th maximum eigenvalue. Hence, the space resulted from performing PCA on  $MM^T + NN^T$  will be the same as the space

resulted from performing PCA on  $[M|N]^T[M|N]$  after the vector is mapped onto  $[M|N]^T$ , that is the result from applying the framework for dimensionality reduction on linear dimensionality reduction. Thus, we can say that using our proposed framework for dimensionality reduction on linear dimensionality reduction will yield the optimum result. ■

### 5.3 Numerical Results

We demonstrate the performance of our proposed algorithm with some datasets from the UCI Machine Learning Repository, whose details are shown in Table 5.1. In the same way to all of our previous experiments, the data were normalized into the range of  $[0, 1]$ . After that, we separated the data into 4 groups of equal size and performed cross-validation among them. Two of the groups were used as original training data to train the given linear dimensionality reduction process, while the other two of them were used as customization data and test data, resulted in  $\frac{4!}{2!1!1!} = 12$  combinations in total. The result was compared with the result from original linear dimensionality reduction and the one trained from customization data.

Table 5.1: Detail of all datasets used in the experiments of combining results from linear dimensionality reduction. The numbers of attributes shown are the numbers of attributes used in dimensionality reduction, omitting some useless attributes like index or specific name.

	No. of attributes	No. of instances
concrete compress	8	1,030
concrete slump	7	103
cpu	6	209
forest	12	517
housing	13	506
parkinsons	19	5,875
servo	4	167

For the linear dimensionality reduction used in the experiments, if one want to use some algorithms with clearly defined objective as the value in optimization then the score can be easily measured from the transformed test data in the same way. However, in our experiments, we used PCA. Since PCA sorts basis vectors for linear space by variance of the data in each axis, the objective

is not well-ordered. So as in many research works, we decide to use the value of determinant and trace of covariance matrix which are the values of products and summation of eigenvalues of the matrix respectively. The perfect result of PCA then will align the axis so that all non-diagonal elements of the covariance matrix will be zero. Since we calculate the mean of the value from cross validation, we will use the natural logarithmic value of determinant instead of using them directly to retain the sense of arithmetic mean. The result is shown in Table 5.2.

Table 5.2: Result from using the proposed frameworks with linear dimensionality reduction. Numbers whose values are minimal in their rows are typeset bold. The second column shows the reduced dimension. All of the equal values are different in higher precision.

dataset	$m$	logdet(G)	logdet(S)	logdet(N)	trace(G)	trace(S)	trace(N)
concrete	1	3.325	3.182	<b>3.346</b>	28.344	25.128	<b>28.996</b>
	3	7.957	<b>8.191</b>	8.190	56.180	57.380	<b>57.707</b>
	5	11.452	<b>12.434</b>	12.257	78.178	<b>81.335</b>	80.777
	7	<b>12.952</b>	12.889	12.944	<b>86.828</b>	86.567	86.810
concrete slump	1	1.022	1.030	<b>1.093</b>	2.856	2.888	<b>3.024</b>
	3	1.969	<b>2.372</b>	2.349	7.322	7.647	<b>7.757</b>
	5	0.909	<b>1.914</b>	1.636	9.703	<b>10.269</b>	10.035
cpu	1	1.279	1.241	<b>1.294</b>	3.782	3.660	<b>3.843</b>
	3	0.559	0.555	<b>0.566</b>	5.623	5.607	<b>5.663</b>
	5	-1.609	-1.679	<b>-1.575</b>	6.968	6.965	<b>6.989</b>
forest	1	3.007	3.000	<b>3.012</b>	20.251	20.114	<b>20.352</b>
	3	8.279	8.253	<b>8.282</b>	48.808	48.411	<b>48.855</b>
	5	11.588	11.519	<b>11.600</b>	59.888	59.526	<b>59.938</b>
	7	13.757	13.720	<b>13.772</b>	66.417	66.256	<b>66.474</b>
	9	<b>13.357</b>	12.794	13.328	68.430	68.262	<b>68.442</b>
housing	1	3.947	3.943	<b>3.948</b>	51.814	51.602	<b>51.872</b>
	3	8.512	8.475	<b>8.516</b>	72.337	71.891	<b>72.386</b>
	5	11.754	11.638	<b>11.769</b>	82.914	82.307	<b>83.033</b>
	7	14.132	14.059	<b>14.152</b>	90.018	89.703	<b>90.115</b>
	9	<b>15.172</b>	14.859	15.070	<b>94.082</b>	93.572	93.974
parkinsons	1	5.775	5.774	<b>5.775</b>	322.042	321.963	<b>322.063</b>
	3	15.148	15.145	<b>15.148</b>	545.028	544.694	<b>545.090</b>
	5	22.983	22.981	<b>22.983</b>	647.335	647.225	<b>647.361</b>
	7	28.320	28.315	<b>28.321</b>	676.847	676.749	<b>676.868</b>
	9	31.271	31.267	<b>31.272</b>	686.197	686.153	<b>686.208</b>
servo	1	2.065	2.040	<b>2.085</b>	7.993	7.829	<b>8.180</b>
	3	5.349	5.346	<b>5.351</b>	19.025	19.005	<b>19.032</b>

It is clearly shown in Table 5.2 that the result from combining both linear



spaces is better than that from the covariance matrix of mapped data, measured by the value of logarithm of determinant and trace which both reflect the eigenvalues of the covariance matrix. One might think that combining two models is similar to the idea of customization for classification by patching, but there are some significant differences. The first thing is that combining two linear spaces with this algorithm performs with both of them on equal ground, while the patching approach in customization for classification treats them differently. The second issue is that, customization for classification by patching predicts the result using one of the two models, and with plausible bias, the result is expected to be an improvement. However, combining linear spaces based on the hypothesis that both of these spaces should be the same but is differed due to variation or error in both datasets. Thus, the most likely best linear space is the linear space taken as a mean of them and the improvement in the experimental result is based on different reason which is less obvious than customization for classification by patching.

In order to demonstrate that this algorithm is applicable for dimensionality reduction that will be later used in classification, we conducted other experiments by performing customization on dimensionality reduction before classification. The experiments were conducted on datasets *optdigits* and *pendigits* whose data were normalized into the range of  $[0, 1]$  as in all of our previous experiments. However, the dataset *optdigits* used in this experiment did not have dimensionality reduced by PCA since this experiment was about dimensionality reduction as well, and thus dataset *optdigits* had 64 attributes. We performed 10-fold cross validation between data used for customization and test data. We used linear discriminant analysis as dimensionality reduction algorithm and 1-nearest neighbor for classification. The result is shown in Table 5.3.

In Table 5.3, for dataset *optdigits*, we can be quite certain that the proposed algorithm yielded better result, but for *pendigits*, it was only the case that the data was reduced to one dimension that was an improvement from both linear spaces. The main difference between both of these datasets is the numbers of attributes; *optdigits* has 64 attributes, *pendigits* has 16 attributes. Therefore, performing dimensionality reduction on *optdigits* with the same number of dimen-

Table 5.3: Classification accuracy after performing customization upon dimensionality reduction. (G) and (S) are the results from original data and customization data respectively, and (N) is the one created from both models with our algorithm.

dataset	$m$	LDA(G)	LDA(S)	LDA(N)
optdigits	1	33.28 $\pm$ 4.03	<b>33.61 <math>\pm</math> 6.01</b>	32.23 $\pm$ 5.09
	3	77.69 $\pm$ 2.98	78.19 $\pm$ 3.63	<b>78.58 <math>\pm</math> 3.19</b>
	5	87.65 $\pm$ 3.01	88.70 $\pm$ 4.74	<b>89.15 <math>\pm</math> 3.01</b>
	7	93.32 $\pm$ 2.12	92.65 $\pm$ 3.26	<b>93.82 <math>\pm</math> 2.43</b>
	9	95.55 $\pm$ 1.70	95.60 $\pm$ 2.35	<b>95.83 <math>\pm</math> 1.91</b>
pendigits	1	39.45 $\pm$ 2.11	38.88 $\pm$ 4.16	<b>40.56 <math>\pm</math> 3.33</b>
	3	<b>81.22 <math>\pm</math> 2.26</b>	78.33 $\pm$ 1.74	80.99 $\pm$ 2.48
	5	<b>93.45 <math>\pm</math> 1.41</b>	91.54 $\pm$ 1.05	91.57 $\pm$ 1.10
	7	<b>97.54 <math>\pm</math> 0.88</b>	97.51 $\pm$ 0.93	97.05 $\pm$ 0.77
	9	98.43 $\pm$ 1.14	<b>98.69 <math>\pm</math> 0.92</b>	98.31 $\pm$ 0.61

sions yielded the better result due to having larger numbers of choice, while it might be hard to determine a good projection for *pendigits*. Also note that the ratio of the number of data used in calculating LDA(G) to the number of data used in calculating LDA(S) was about twice for both *optdigits* and *pendigits*. The other noteworthy issue is where the value of each attributes of *optdigits* and *pendigits* came from. A value in each attributes of *optdigits* corresponds to the grayscale color in each pixel of an 8x8 image, while the attributes of *pendigits* has various meaning and many different interpretation. By the nature of linear dimensionality reduction, it is better fit to the task that each attribute has similar meaning like *optdigits* since the underlying distribution of the data will more likely be in linear subspace and yield the suitable result upon them. One might argue with the previous statement about linear dimensionality reduction being better on *optdigits* than *pendigits* using the accuracy when both of them are reduced to the same number of dimensions as shown in Table 5.3 that the value in *pendigits* is higher and both of them has the same numbers of classes corresponding to the numbers 0-9. This was likely resulted from that the data obtained directly from pixels of *optdigits* are distributed sparsely in high dimensional linear space while data obtained from feature extraction of *pendigits* lie densely in lower dimensions with more complicated distribution.

# CHAPTER VI

## CONCLUSION AND FUTURE WORK

### 6.1 Conclusion

We have introduced the idea of performing task-based customization. We consider the task in performing customization for classification, sometimes called adaptive classification, which is the field that research in customization is most active on. We introduce three frameworks in performing customization for classification and provide some algorithms according to the frameworks. We then conduct the experiments with real-world datasets, and show the numerical results.

In Chapter 5, we consider the task in performing customization for dimensionality reduction, which could be the preprocess of classification. We propose the framework of customization for dimensionality reduction which seems to be the only sensible one due to the constraint of dimensionality reduction. We further consider the special case of dimensionality reduction which is linear dimensionality reduction in the process of combining result from two dimension reducer which is the only way to perform customization that can be applied to nonlinear dimensionality reduction as well. We propose an algorithm, and provide some theoretical results. We state that the algorithm will yield the optimal result in combining the result from two linear dimensionality reduction. We further state that this optimal algorithm in customization for linear dimensionality reduction can be thought of as a special case of our proposed framework for dimensionality reduction. After that we conduct the numerical results on the algorithm for linear dimensionality reduction to show its performance as the process of dimensionality reduction and as preprocess for classification.

### 6.2 Future Work

There are many other tasks in machine learning to which we can apply the concept of task-based customization. Other than that there is also room for many frameworks and variations of the algorithm to be created. As shown in

Chapter 5, the wide applicability of task-based customization could result in the weakness for theoretical bit, but by applying additional constraints on the model, it can provide theoretically satisfying result up to a degree. One might argue that restricting the process with additional constraints conflicts to naming it as task-based customization and its mentioned advantage to the model-specific ones, but the main point here is that if the range of the algorithm that this approach can be applied to is wide enough then it still be useful enough in this matter.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## References

- Asuncion, A., and Newman, D. UCI machine learning repository [Online]. 2007. Available from : <http://archive.ics.uci.edu/ml/> [2010, November 5].
- Cao, X., and Balakrishnan, R. Evaluation of an on-line adaptive gesture interface with command prediction. In Proceedings of Graphics Interface 2005, GI '05, pp. 187–194. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.
- Chatpatanasiri, R., Korsrilabutr, T., Tangchanachaianan, P., and Kijirikul, B. On kernelization of supervised mahalanobis distance learners. CoRR abs/0804.1441.
- Chatpatanasiri, R., Korsrilabutr, T., Tangchanachaianan, P., and Kijirikul, B. A new kernelization framework for mahalanobis distance learning algorithms. Neurocomputing 73(10-12) (2010): 1570–1579.
- Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., Kavraki, L.E., and Thrun, S. Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Cambridge, MA. June 2005.
- Cristianini, N., and Shawe-Taylor, J. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, 1 edition. 2000.
- Friedman, J.H. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University. 1996.
- Fu, H.C., Chang, H.Y., Xu, Y.Y., and Pao, H.T. User adaptive handwriting recognition by self-growing probabilistic decision-based neural networks. Neural Networks, IEEE Transactions on 11(6) (November 2000): 1373–1384.
- Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. Neighbourhood components analysis. Advances in Neural Information Processing Systems 17 (2005): 513–520.

- Halton, J.H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik 2 (1960): 84–90.
- Han, J., and Kamber, M. Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann, 1st edition. September 2000.
- Hastie, T., Tibshirani, R., and Friedman, J. The Elements of Statistical Learning. Springer Series in Statistics. Springer New York Inc., New York, NY, USA. 2001.
- Kijsirikul, B., Ussivakul, N., and Meknavin, S. Adaptive directed acyclic graphs for multiclass classification. In PRICAI '02: Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence, pp. 158–168. Springer-Verlag, London, UK.
- Lyu, R.Y., Chien, L.F., Hwang, S.H., Hsieh, H.Y., Yang, R.C., Bai, B.R., Weng, J.C., Yang, Y.J., Lin, S.W., Chen, K.J., Tseng, C.Y., and Lee, L.S. Golden mandarin (iii)-a user-adaptive prosodic-segment-based mandarin dictation machine for chinese language with very large vocabulary. In Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on, volume 1, pp. 57–60.
- Michie, D., Spiegelhalter, D.J., Taylor, C.C., and Campbell, J., editors. Machine learning, neural and statistical classification. Ellis Horwood, Upper Saddle River, NJ, USA. 1994.
- Mitchell, T.M. Machine Learning. McGraw-Hill, New York. 1997.
- Nicholson, W.K. Elementary Linear Algebra. McGraw-Hill, New York, USA, 1st international edition edition. 2001.
- Phetkaew, T., Rivepiboon, W., and Kijsirikul, B. Reordering adaptive directed acyclic graphs for multiclass support vector machines. JACIII 7(3) (2003): 315–321.
- Schölkopf, B., Smola, A., and Müller, K.R. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation 10(5) (July 1998): 1299–1319.

- Weinberger, K.Q., and Saul, L.K. Distance Metric Learning for Large Margin Nearest Neighbor Classification. J. Mach. Learn. Res. 10 (2009): 207–244.
- Wylie, C.R., and Barrett, L.C. Advanced Engineering Mathematics. McGraw-Hill, New York, USA, 5th edition. 1982.
- Zhang, W., Xue, X., Sun, Z., Guo, Y.F., and Lu, H. Optimal dimensionality of metric space for classification. In ICML '07: Proceedings of the 24th international conference on Machine learning, pp. 1135–1142. ACM Press, New York, NY, USA.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย

## Biography

Pasakorn Tangchanachaianan was born in Chonburi, Thailand, on 2 December 1982. He received Bachelor Degree of Engineering and Master Degree of Engineering both in the field of computer engineering from Chulalongkorn University.



ศูนย์วิทยทรัพยากร  
จุฬาลงกรณ์มหาวิทยาลัย