Sequential Pattern Classification Without Explicit Feature Extraction

by

Hansheng Lei

October 21st, 2005

A DISSERTATION SUBMITTED TO

THE FACULTY OF THE GRADUATE SCHOOL

OF STATE UNIVERSITY OF NEW YORK AT BUFFALO

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Abstract

Feature selection, representation and extraction are integral to statistical pattern recognition systems. Usually features are represented as vectors that capture expert knowledge of measurable discriminative properties of the classes to be distinguished. The feature selection process entails manual expert involvement and repeated experiments. Automatic feature selection is necessary when (i) expert knowledge is unavailable, (ii) distinguishing features among classes cannot be quantified, or (iii) when a fixed length feature description cannot faithfully reflect all possible variations of the classes as in the case of sequential patterns (e.g. time series data). Automatic feature selection and extraction are also useful when developing pattern recognition systems that are scalable across new sets of classes. For example, an OCR designed with explicit feature selection process for the alphabet of one language usually does not scale to an alphabet of another language.

One approach to avoiding explicit feature selection is to use a (dis)similarity representation instead of a feature vector representation. The training set is represented by a similarity matrix and new objects are classified based on their similarity with samples in the training set. A suitable similarity measure can also be used to increase the classification efficiency of traditional classifiers such as Support Vector Machines (SVMs).

In this thesis we establish new techniques for sequential pattern recognition without explicit feature extraction for applications where: (i) a robust similarity measure exists to

distinguish classes and (ii) the classifier (such as SVM) utilizes a similarity measure for both training and evaluation. We investigate the use of similarity measures for applications such as on-line signature verification and on-line handwriting recognition. Paucity of training samples can render the traditional training methods ineffective as in the case of on-line signatures where the number of training samples is rarely greater than 10. We present a new regression measure (ER^2) that can classify multi-dimensional sequential patterns without the need for training with large number of prototypes. We use ER^2 as a preprocessing filter in cases when sufficient training prototypes are available in order to speedup the SVM evaluation. We demonstrate the efficacy of a two stage recognition system by using Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) in the supervised classification framework of SVM. We present experiments with off-line digit images where the pixels are simply ordered in a predetermined manner to simulate sequential patterns. The Generalized Regression Model (GRM) is described to deal with the unsupervised classification (clustering) of sequential patterns.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Number Page

ACKNOWLEDGMENTS

I wish to express special gratitude to my advisor, Dr. Venu Govindaraju, for his continuous support and encouragement throughout the dissertation process.

Many thanks to the members of my committee, Dr. Peter Scott and Dr. Matthew Beal for their thoughtful judgement and resolute commitment to student success.

I also appreciate the financial support from the Department of Computer Science and Engineering in the State University of New York at Buffalo for my Ph.D. study.

Chapter 1

INTRODUCTION

Pattern Recognition (PR) is one of the classical disciplines of Artificial Intelligence in computer science. It primarily involves three steps (Fig. 1 a): 1) data acquisition and preprocessing, 2) data representation and 3) decision making. In the first step, the raw pattern is preprocessed and normalized. The second step includes feature extraction/selection, where the pattern is represented as a set of features. While feature selection can be (semi)-automatic [30], feature extraction usually assumes that the features have predefined meaning. In order to select discriminative features, numerous handcrafted experiments have to be conducted. For example, the Gradient, Structure, Concavity (GSC) features for handwritten digits were achieved after many experiments by researchers at CEDAR[26]. The third step in PR involves the classification decision making. The design of a classifier has been extensively studied. There are several classifiers available, such as Nearest Neighbor (NN), Neural Networks, Decision Trees and Support Vector Machines (SVMs). Since most current classifiers can be efficiently adapted to any application and the data preprocessing stage is relatively simple, the complex part of PR system is actually the feature extrac-

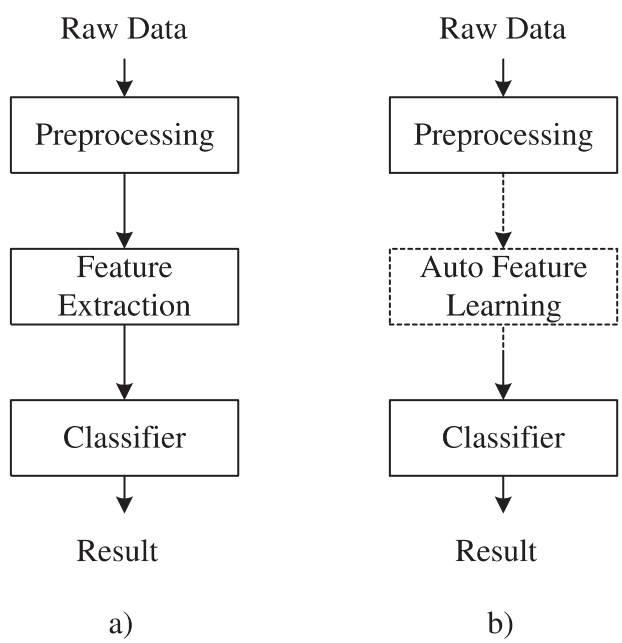


Figure 1.1: a) The basic procedure of recognition with explicit feature extraction. b) The procedure of recognition with implicit feature extraction.

tion stage. In this thesis, we explore a methodology to circumvent the difficult stage of feature definition, selection and extraction by directly inputting the raw data (with simple preprocessing) to the classifier, as show in Fig. 1 b).

In order to minimize human involvement in the design of a PR system, we can either use automatic feature selection and extraction or let the classifier absorb the function of feature extraction. The former method leads to *automatic feature extraction*, which is intrinsically tied to feature selection. The latter leads to *featureless classification* [23, 24]. While feature selection and extraction have been extensively studied in many application domains, such as handwriting, bioinformatics and multimedia, the approach of featureless classification is relatively new [54].

We will review featureless classification in section 1.1 and define the notion of a *sequential pattern* in section 1.2. These sections will serve as the motivation of this thesis : *Sequential Pattern Classification without Explicit Feature Extraction*.

1.1 Featureless Classification

Traditional pattern classification deals with objects represented in feature space. That is, the objects to be clustered or classified are represented in terms of *features* that are intrinsic to the object. The advantage of this approach is that it can make the representation of the object compact. However, in many applications, objects, such as handwritten characters/words, graphs and gene sequences, cannot be easily represented in a feature space without adding additional constraints. For example, representing all the scripts in the world at the same time in a feature space is nearly impossible. At the same time defining features for one script at a time is quite challenging and costly. Further, the objects with graph representations lack a canonical order or correspondence between nodes. Therefore, graphs are hard to represent in a feature vector. Even if a vector mapping can be established, the vectors would be of variable length and thus not belong to a single feature space.

The drawbacks of feature-based classification is a strong motivation for (dis)similarity-based classification, also called *featureless classification* (Duin et al. [23]). When the objects are hard to represent in form of feature vectors, it is often easier to obtain a similarity/dissimilarity measure and evaluate pairwise (dis)similarities between the objects.

Recently, there has been considerable interest in (dis)similarity-based techniques, both for developing and studying new effective distances or similarities between non vectorial entities, such as graphs, sequences and structures. By using (dis)similarity measures, the classification algorithm can be kept generic and independent of the actual data representation. Thus, the approaches based on (dis)similarity measures are readily applicable even to problems that do not have a natural embedding in a uniform feature space, as in the case of clustering of structural or sequential patterns. Pairwise classification algorithms, such as kernel methods [9], spectral clustering techniques/graph partitioning and self-organizing maps, implicitly utilize the (dis)similarities between entities, even when the objects are represented by feature vectors. Therefore, (dis)similarity-based techniques are more general than feature-based techniques.

Duin et al. discussed the possibility of training statistical pattern recognizers based on a distance representation of the objects instead of a feature representation [23]. Distances or similarities are used between the unknown objects to be classified with a selected subset of the training objects. In this approach, the feature selection task is substituted by the task of finding good similarity measures. Their approach is to directly use the similarity matrix which is required by Support Vector Machines (SVMs). The training of SVM explicitly uses a similarity measure via a kernel function. The actual representation of data is not significant in SVM. What SVM needs for training is a similarity measure and the pairwise similarities between training objects. Therefore, SVM naturally supports featureless classification.

Mottl et al. applied the featureless pattern recognition methodology to the problem of protein fold classification [54]. They measure the likelihood of two proteins having the same evolutionary origin by computing the alignment score between two amino acid sequences. To solve the problem of determining the membership of a protein given by amino acid sequence in one of preset fold classes, the set of all feasible amino acid sequences are treated as a subset of isolated points in a Hilbert space, in which the linear operations and inner product are defined in an arbitrary and unknown manner.

The concept of *featureless* PR is relatively new. One might argue that the featureless approach uses the object itself as feature and thus strictly speaking, is not ”featureless”. Notwithstanding the name of the technique, the concept is promising. For example, there are many features that have been proposed for handwritten character recognition in recent decades and more are continually being invented. Do we really need so many features? We want to answer such questions by avoiding the costly feature definition procedure. The emphasis of this thesis is on exploring the featureless PR in the domain of sequential patterns.

1.2 Sequential Pattern

A pattern is a compact and semantically rich representation of raw data. A *Sequential Pattern* is the kind of pattern whose attributes are organized in some linear order. Typical examples of sequential patterns are Time Series data, on-line handwritten characters/words, on-line signatures, etc. Mining sequential patterns from large databases is also an important problem in data mining [5]. In this thesis, we focus on those patterns whose attributes are naturally ordered in a sequence, such as on-line handwritten characters and on-line signatures.

1.3 Sequential Pattern Classification without Explicit Feature Extraction

Sequential Pattern Classification without Explicit Feature Extraction is realizable under two conditions: (i) a robust similarity measure must exist which matches two sequential patterns directly. (ii) a classifier that can be seamlessly integrated with the similarity measure. Thus, the contributions of this thesis are two-fold: presenting an intuitive similarity measure for sequential pattern matching and seamlessly integrating a state-of-the-art learning machine such as SVM classifier with the similarity measure.

1.4 Organization of Thesis

This thesis is organized as follows. Chapter 1 serves as an introduction of background and motivation. In Chapter 2, we propose ER^2 as a similarity measure for sequential patterns. In Chapter 3, we describe one application of ER^2 in on-line signature verification. Chapter 4 presents an on-line handwritten character recognition method using ER^2 and SVM. Chapter 5 presents an implicit feature extraction method for off-line digit images where the pixels are simply ordered to simulate sequential patterns. In Chapter 6, we propose the Generalized Regression Model (GRM) for unsupervised sequence classification. Finally, Chapter 7 draws conclusions and outlines future work.

Chapter 2

DIRECT SIMILARITY MEASURE

One simple implementation of featureless pattern recognition can be realized by direct pattern matching. Direct similarity measure matches two objects in their original representations without explicit feature extraction. If we can define a direct similarity measure for the objects, a featureless classification can be implemented in the framework of the nearest-neighbor classifier. In this chapter, we present a direct and intuitive similarity measure that can be used for matching sequential patterns.

2.1 Background and Related Works

Sequence analysis has attracted research interest in a wide range of applications. Stock trends, on-line handwritten characters/words, medical monitoring data such as ERG, etc., can be represented in form of sequential patterns. While matching/sub-matching [71, 5], indexing [4, 34], clustering [57, 47] and rule discovery [17] are the fundamental techniques used in this field, the core problem remains to be that of defining the similarity measure. Several methods exist for measuring the (dis)similarity of two sequences. We briefly describe four commonly used methods and present their drawbacks.

2.1.1 Lp norms

Given two sequences $X = [x_1, x_2, \cdots, x_N]$ and $Y = [y_1, y_2, \cdots, y_N]$, Lp norms are defined as $Lp(X,Y) = (\sum_{i=1}^{N} |x_i - y_i|^p)^{\frac{1}{p}}$ [2, 78]. When p=2, this reduces to commonly used Euclidean distance. Lp norms are straightforward and easy to compute. However, in many cases such as in shifting and scaling, the distance of two sequences cannot reflect the "real" (dis)similarity between them. Suppose sequence $X_1 = [1, 2, \cdots, 10]$ and $X_2 = [101, 102, \cdots, 110]$. X_2 is the result of shifting X_1 by 100, i.e., adding 100 to each element of X_1. Although Lp distance between X_1 and X_2 is large, they can be considered similar in many applications [27, 14]. The same is true in the case of scaling. For example, let $X_2 = \beta X_1$, where β is a scaling factor. In some applications, X_2 is still considered to be similar to X_1. Unfortunately, Lp norms cannot capture these types of similarity. Furthermore, Lp distance has meaning only in the relative sense when used to measure (dis)similarity. That is, a distance between two sequences X_1 and X_2 alone (e.g., $Distance(X_1, X_2) = 95.5$) does not convey how (dis)similar X_1 and X_2 are, unless we have another distance to compare. For example, $Distance(X_1, X_3) = 100.5 > 95.5$ could mean that X_1 is "more similar" to X_2 than to X_3. Thus, Lp norms as measure of (dis)similarity has two disadvantages: (i) it cannot capture similarity in applications that involve shifting and scaling and (ii) the distance has meaning only in a relative sense. One could use the mean-deviation normalization ($\overrightarrow{X} = (X - mean(X))/std(X)$) to overcome the shifting and scaling factors. However, it does not tell what the shifting and scaling factors are. Those factors might be exactly what we need for sequences analysis. While it may appear that the scaling factor β_1 between

Y and X is $\frac{std(Y)}{std(X)}$, this is true only under the condition that Y and X have an *exact* linear relation of the form $Y = \beta_0 + \beta_1 X$ [47].

2.1.2 Transforms

Fourier Transform and Wavelet Transform are commonly used for sequence analysis [62, 69, 11]. Both transforms can concentrate most of the energy a small region of the frequency domain, thus allowing for the process to be carried out in a small region with few coefficients, reducing the dimension and saving computational time. Transforms are used primarily for feature extraction and therefore the task of defining some type of similarity measure still remains.

2.1.3 Time Warping

Dynamic Time Warping (DTW) [7, 79, 40] defines the distance between sequences $X_i = [x_1, x_2, \cdots, x_i]$ and $Y_j = [y_1, y_2, \cdots, y_j]$ as $D(i, j) = |x_i - y_i| + min\{D(i-1, j), D(i, j-1), D(i-1, j-1)\}$. This distance can be computed using dynamic programming. It has the advantage that it can tolerate some local non-alignment in the time phase so that the two sequences do not have to begin with the same length. DTW is generally more robust and flexible than Lp norms. However, it has the same disadvantages as Lp norms. It is sensitive to shifting and scaling and the warping distance has meaning only in a relative sense.

2.1.4 Linear relation

Linear transform is of the form $Y = \beta_0 + \beta_1 X$ [27, 14]. Sequence X is said to be similar to Y if we can determine β_0 and β_1 so that the $Distance(Y, \beta_0 + \beta_1 X)$ is minimized below a given threshold. Chu et al. solve the scaling factor β_1 and shifting offset β_0 from a geometrical standpoint [14]. However, the distance again has meaning only in a relative sense.

2.2 Linear Regression

Linear regression analysis originated from statistics and has been widely used in econometrics [53, 77]. For example, to test the linear relation between consumption Y and income X, we can establish the linear model as:

$$Y = \beta_0 + \beta_1 X + u \tag{2.1}$$

The variable u is called the *error term* and relation (2.1) is called "*the regression of Y on X*". Given a set of sample data, $X = [x_1, x_2, \cdots, x_N]$ and $Y = [y_1, y_2, \cdots, y_N]$, β_0 and β_1 can be estimated in the sense of the minimum-sum-of-squared-error. That is, a line, called regression line, can be found in the Y-X space, to fit the points $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$. We need to determine β_0 and β_1 such that $\sum_{i=1}^{N} u_i^2 = \sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_i)^2$ is minimized (Fig. 6.1 a). Using first order conditions, β_0 and β_1 can be solved as follows:

$$\beta_0 = \overline{y} - \beta_1 \overline{x} \tag{2.2}$$

$$\beta_1 = \frac{\sum_{i=1}^{N} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{N} (x_i - \overline{x})^2} \tag{2.3}$$

where $\overline{x}$ and $\overline{y}$ denote the average of sequence X and Y respectively.

After obtaining β_0 and β_1, one could pose a question: how well does the regression line fit the data? In order to answer such a question, *R-squared* is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N} u_i^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{2.4}$$

From (2.4) we can further derive:

$$R^2 = \frac{[\sum_{i=1}^{N}(x_i - \overline{x})(y_i - \overline{y})]^2}{\sum_{i=1}^{N}(x_i - \overline{x})^2 \sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{2.5}$$

The value of R^2 is always between 0 and 1. The closer the value is to 1, the better the regression line fits the data points. Thus, R^2 is a measure for the *Goodness-of-fit*.

The regression model described by 2.1 is called the *Simple Regression Model*, as it involves only one independent variable X and one dependent variable Y. We can add more independent variables to the model as follows:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_K X_K + u \tag{2.6}$$

The regression model described by 2.6 is called the *Multiple Regression Model*. $\beta_0, \beta_1, \cdots, \beta_K$ can be estimated using the first order conditions.

R^2 has following properties:

- **Reflexivity**, i.e., $R^2(X,X) = 1$.

- **Symmetry**, i.e., $R^2(X,Y) = R^2(Y,X)$.

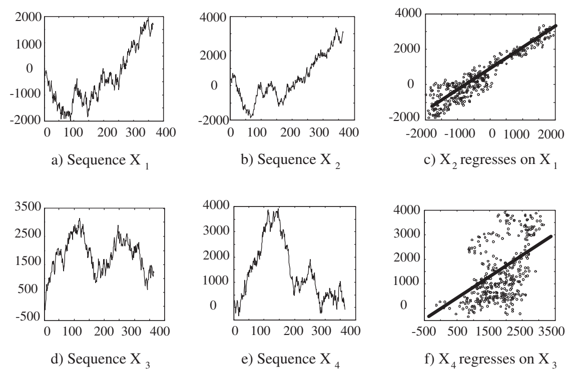


Figure 2.1: Two pairs of sequences. $R^2(X_1,X_2) = 0.91(91\%)$ and $R^2(X_3,X_4) = 0.31(31\%)$. X_1 and X_2 have high similarity because the points in c) are distributed along a line. X_3 and X_4 have low similarity with scattered distribution in f).

- $R^2 \in [0,1]$. The closer the value is to 1, more are the number of points that fall along the regression line indicating a stronger linear relation between the two sequences. $R^2 = 1$ means the two sequences have a perfect linear relation, while $R^2 = 0$ means they have no linear relation at all.

Based on the properties above, R^2 is a good measure for similarity. Fig. 2.1 shows examples of two pairs of 1-dimensional sequences (with length 400) illustrating high (low) R^2 value means high (low) similarity. Thus, given two sequences, R^2 directly indicates their similarity.

R^2 has an intrinsic relation with the normalized Euclidean distance by mean-deviation. Let $\nu(X)$ denote the normalized version of X by mean-deviation, i.e., $\nu(X) = (X - \overline{X})/\sigma(X)$. Let $D_2(X,Y)$ denote the Euclidean distance between sequence X and Y, then we have the following equation:

$$D_2(\nu(X),\nu(Y)) = \sqrt{2N(1-r)}, \tag{2.7}$$

where r is the Pearson's coefficient, $R^2 = r^2$ and N is the length of the sequence. The proof is as follows.

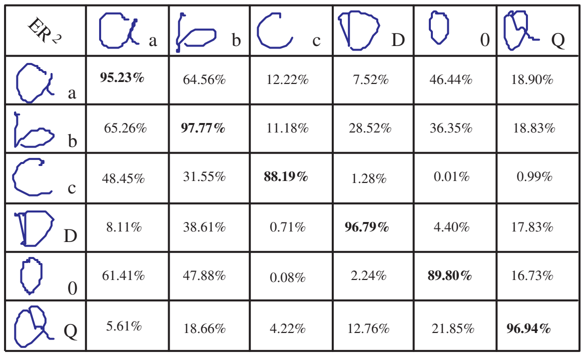
$$
\begin{aligned}
[D_2(\nu(X), \nu(Y)]^2 &= \sum_{i=1}^{N} (\frac{x_i - \overline{X}}{\sigma(X)} - \frac{y_i - \overline{Y}}{\sigma(Y)})^2 \\
&= \frac{\sum_{i=1}^{N}(x_i - \overline{X})^2}{\sigma(X)^2} + \frac{\sum_{i=1}^{N}(y_i - \overline{Y})^2}{\sigma(Y)^2} \\
&\quad -2\frac{\sum_{i=1}^{N}(x_i - \overline{X})(y_i - \overline{Y})}{\sigma(X)\sigma(Y)} \\
&= N + N - 2Nr = 2N(1-r)
\end{aligned}
$$

Note that $\sigma(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{X})^2$.

After performing the mean-deviation normalization, the shifting and scaling factors are discarded [14]. In fact, normalization loses the information of the shifting and scaling factors. From equations of (2.2) and (2.3), we can see that both β_0 and β_1 are more sophisticated than what we can obtain from normalization. Only under the *exact* linear relation ($R^2 = 1$), we can derive $\beta_1 = \frac{\sigma(Y)}{\sigma(X)}$ according to (2.3) and (2.5). However, such an exact linear relation rarely exists in real applications.

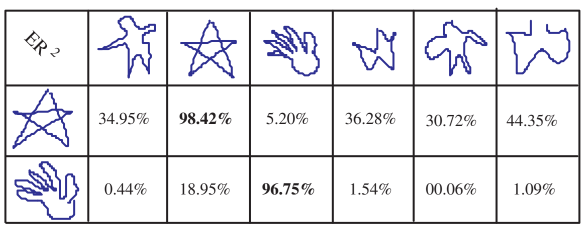
2.3 ER^2: Extended R-squared

Unfortunately, R^2 is applicable only to 1-dimensional sequences, but in real applications, sequences are often multi-dimensional. For example, the on-line handwritten character sequences are multi-dimensional, including at least the X-, Y-coordinates. To match a pair



ER^2	a	b	c	D	0	Q
a	**95.23%**	64.56%	12.22%	7.52%	46.44%	18.90%
b	65.26%	**97.77%**	11.18%	28.52%	36.35%	18.83%
c	48.45%	31.55%	**88.19%**	1.28%	0.01%	0.99%
D	8.11%	38.61%	0.71%	**96.79%**	4.40%	17.83%
0	61.41%	47.88%	0.08%	2.24%	**89.80%**	16.73%
Q	5.61%	18.66%	4.22%	12.76%	21.85%	**96.94%**

Figure 2.2: Similarities by ER^2 between on-line handwritten characters. The similarities between same class of characters are intuitively high.



ER^2						
	34.95%	**98.42%**	5.20%	36.28%	30.72%	44.35%
	0.44%	18.95%	**96.75%**	1.54%	00.06%	1.09%

Figure 2.3: Similarities by ER^2 between artificial 2D shapes. The similarities between visually similar objects are intuitively high.

of D-dimensional sequences, ER^2 (Extended R-squared) is defined as:

$$ER^2 = \frac{[\sum_{j=1}^{D}(\sum_{i=1}^{n}(x_{ji}-\overline{X_j})(y_{ji}-\overline{Y_j}))]^2}{\sum_{j=1}^{D}\sum_{i=1}^{n}(x_{ji}-\overline{X_j})^2\sum_{j=1}^{D}\sum_{i=1}^{n}(y_{ji}-\overline{Y_j})^2} \tag{2.8}$$

where $\overline{X_j}$ ($\overline{Y_j}$) is the average of the j-th dimension of sequence X (Y).

ER^2 has properties similar to R^2, i.e., *reflexivity*, *symmetry* and $ER^2 \in [0,1]$. ER^2 measures multidimensional sequences and directly indicates how similar they are. This is in contrast to the Euclidean or DTW distances which are not as intuitive.

Sequential pattern classification can benefit from ER^2 as the similarity metric. Fig. 2.2 shows some examples of similarities by ER^2 between on-line handwritten characters. The similarities between characters of the same class are intuitively high (all above 88%), while similarities between different characters are low (all below 70%). Fig. 2.3 shows examples on several artificial 2D shapes.

2.4 Chapter Summary

Similarity measure ER^2 is defined for direct sequential pattern matching. ER^2 is extended from R^2 which is the Goodness-of-fit in linear regression. The meaning of similarity by ER^2 for 2D shapes and on-line handwritten characters is intuitive.

Chapter 3

ON-LINE SIGNATURE VERIFICATION USING ER^2

In this chapter, ER^2 (Extended R-squared) is presented as a similarity measure for on-line signature verification. SLR (Simple Linear Regression) defines R^2 as a measure of the *goodness-of-fit*. We have observed that R^2 is an intuitive similarity measure for 1-dimensional sequences [47]. However, many kinds of sequences are multidimensional, such as on-line signature sequences, 2D curves, etc. Therefore, we extend R^2 to ER^2 for multidimensional sequence matching. Coupled with optimal alignment, ER^2 outperforms DTW-based curve matching on on-line signature verification.

3.1 On-line Signature Verification Background

As one of the biometric authentication methods, signature has been widely accepted, because it is more user-friendly than fingerprint, iris, retina and face. On-line signatures are acquired using a digitizing tablet which captures both temporal and spatial information, such as coordinates, pressure, inclinations, etc.

There are two challenging issues in on-line signature verification: (i) intra-personal variation and (ii) paucity of training samples. Some people provide signatures with poor consistency. The speed, pressure and inclinations pertaining to the signatures made by the same person can differ significantly, which makes it difficult to extract consistent features. Further, we can only expect a few samples from each person and no samples of forgeries are available in practice. This makes the task of determining the consistency of extracted features difficult. Due to limited number of training samples, the determination of threshold that decides rejection or acceptance is also an open problem.

Given two signatures to compare, it is natural to ask *"how similar are they?"* or *"what is their similarity?"* It is intuitive to answer the similarity with a value between 0%-100% and this value should make sense. For example, when we quantize the similarity of two signatures as 90%, they should be very close to each other.

No matter what features are used, such a similarity measure is required. Euclidean distance, Dynamic Time Warping(DTW) [7] or other distances have a meaning only in a relative sense. That is, the distance itself does not convey the similarity without comparing it with other distances. We observed that R^2 is a good similarity measure with intuitive meaning [47]. Given two sequences, R^2 answers the similarity with a value between 0%-100%. This kind of similarity measure is useful for signature verification, especially given that only a few genuine signatures are available in practice. However, R^2 comes from SLR (Simple Linear Regression) which traditionally measures two 1-dimensional sequences. Extended from R^2, ER^2 is suitable for multidimensional sequence matching.

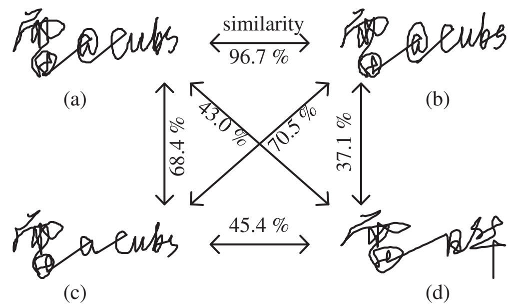


Figure 3.1: Four signatures and their similarities with each other defined by ER^2.

3.2 The Intuition behind ER^2

Fig. 3.1 shows examples of on-line signatures and their similarity by ER^2. Signature (a) and (b) are genuine signatures. Signature (c) and (d) are forgeries, although they are made by the same person. We can see that signature (a) has high similarity with signature (b). The ER^2 between them is 96.7%. The major difference in signature (c) is that the character "@" is replaced by "a". So, the similarity between (c) and the genuine ones drops to 68.4% or 70.5%. Signature (d) is made up of two Chinese characters. The first character is the same as other signatures while the second one is totally different from "@cubs". Therefore, its similarities with other signatures are all below 50%.

The signatures here are made up of Chinese character, English character and special symbol as "@". The examples also demonstrate that ER^2 is a language-independent measure.

3.3 Coupling DTW into ER^2

ER^2 has its own drawback. It only allows one-one matching between two sequences, just like the Euclidean norm. If two sequences are not aligned properly or they have different lengths, ER^2 cannot be applied directly. Usually, signature sequences are neither of the same length nor aligned properly. DTW can be used to determine the optimal alignment between two sequences with different lengths. Thus, by combing DTW with ER^2, we can overcome the alignment problem while retaining all the advantages of ER^2.

3.3.1 DTW background

Given two sequences $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_m)$, the distance DTW(X,Y) is similar to edit distance. To compute DTW distance D(X,Y), we can first construct an n-by-m matrix, as shown in Fig. 3.2. Then, we find a *path* in the matrix which starts from cell $(1,1)$ to cell (n,m) so that the average cumulative cost along the path is minimized. If the path passes cell (i,j), then the cell (i,j) contributes $cost(x_i,y_j)$ to the cumulative cost. The *cost* function can be defined flexibly depending on the application, for example, $cost(x_i,y_j) = |x_i - y_i|^2$. This path can be determined using dynamic programming, because the recursive equation holds: $D(i,j) = cost(x_i,y_j) + min\{D(i-1,j), D(i-1,j-1), D(i,j-1)\}$.

The path may pass through several cells horizontally along X or vertically along Y, which makes the matching between the two sequences not strictly one-one but one-many and many-one. This is the primary advantage that DTW provides for aligning sequences.

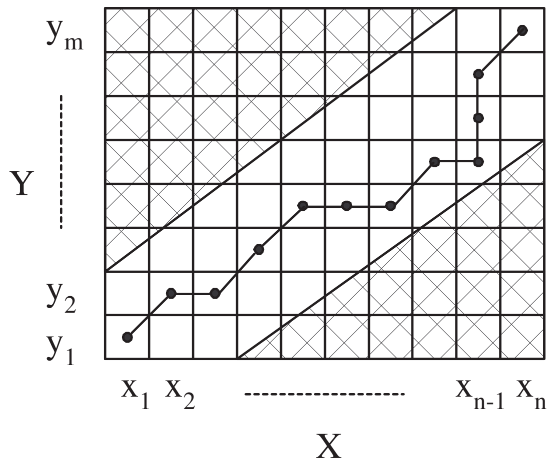


Figure 3.2: The path determined by DTW in the n-by-m matrix has the minimum average cumulative cost. The marked area is the constraint that path cannot go. The path indicates the optimal alignment: (x_1,y_1), (x_2,y_2), (x_3,y_2), $\cdots$, (x_{n-1},y_{m-3}), (x_{n-1},y_{m-2}), (x_{n-1},y_{m-1}), (x_n,y_m).

3.3.2 ER^2 coupled with optimal alignment

We first use DTW to determine the optimal alignment between two sequences. Then, we stretch the two sequences to have the same length. If point x_i in sequence X is aligned with $k(k>1)$ points in sequence Y, we stretch X by duplicating point x_i ($k-1$ times). For sequence Y, we stretch it in the same way. For example, in Fig. 3.2, we stretch X to be $(x_1,x_2,\cdots,\underline{x_{n-1},x_{n-1},x_{n-1}},x_n)$ and Y to be $(y_1,\underline{y_2,y_2},\cdots,y_{n-2},y_{n-1},y_n)$.

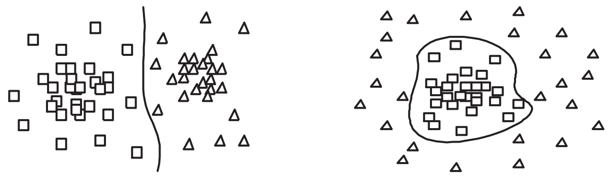
After being stretched, the two sequences have the same length. Then, we can apply equation (2.8) to calculate ER^2 similarity.

3.4 Experiments in on-line signature verification

Our experiments addresses two important issues: i) *What features are consistent and reliable for on-line signature verification ?* A large number of features have been proposed by researchers for on-line signature verification. However, little work has been done in measuring the consistency and discriminative power of these features. So firstly, we will present a comparative study of features commonly used in on-line signature verification. A consistency model is developed by generalizing the existing feature-based measure to distance-based measure. Being aware of the most consistent features, we can move on to address the second issue: ii) *Will* ER^2 *coupled with optimal alignment improve the performance of DTW in signature verification?* Signature verification based on curve matching [55, 52] is a promising and general approach, because curve matching is language-independent. Segmentation-based methods [63] are dependent on languages to a large degree, since the words of some languages are complex with a lot of possible segments and the segments may not be consistent.

3.4.1 A Comparative Study on the Feature Consistency

Signature verification is essentially a one-category classification problem, where several genuine samples are available and counter-examples are very few, if any. Thus, it is different from the standard two-category problem where there are two clusters that can be separated by a decision boundary that is typically an *open* hyper-plane (Fig. 3.3 a). However, in the case of signature verification, there is only one cluster of genuine signatures,



a) b)

Figure 3.3: Difference between two-category and one-category classification problem. a) Regular two-category problem has two clusters with open decision boundary. b) Signature verification is a one-category problem with closed decision boundary around the clustering genuine signatures.

while forgeries have no clustering characteristic because they have no reason to be close to each other. Therefore, the decision boundary for the one-category classification is a *closed* hyper-plane.

Given a few samples of genuine signatures (say, no more than 6), as is often the case in real applications, it is quite challenging to determine the decision boundary no matter what features are used. Because of limited training samples for genuine signatures, the consistency of features becomes extremely important. There are many potential features to choose from [56, 43, 33] and many new features are continually being invented. Therefore it is necessary to study the consistency of features for on-line signature verification.

Existing consistency model

A simple consistency measure has been previously defined as [43]:

$$d_i(a) = \frac{|m(a,i) - m(f,i)|}{\sqrt{\sigma^2(a,i) + \sigma^2(f,i)}}, \tag{3.1}$$

where $d_i(a)$ denotes the consistency of feature i for subject a. $m(a,i)$ and $m(f,i)$ are, respectively, the sample mean among feature i of genuine and forged signatures. $\sigma^2(a,i)$ and $\sigma^2(f,i)$ are the corresponding sample variations of feature i.

This model has an intuitive geometric meaning. It captures the separability between two clusters of features that are assumed to have Gaussian distribution. While the model is suitable for standard two-category classification, it is inadequate for signature verification primarily because the model is based directly on feature values. There are many kinds of features used in signature verification. Some are scalar and some are sequential and of variable length. The mean $m(a,i)$ or $m(a,i)$ of sequential features can not be computed directly. For example, if we define the coordinate sequence itself (X, Y or $[X,Y]$) as a feature, it is difficult to obtain the mean because the sequences are of different lengths. Although sequences can be uniformly re-sampled [56] to be of the same length, the mean of sequences does not carry a practical sense. Moreover, each feature is associated with a certain distance measure, not limited to the Euclidean norm. Dynamic Time Warping (DTW) [52] and correlation [56] are also widely used. The existing model implicitly assumes that the features are scalar or vectors of the same length and the distance measure is the Euclidean norm.

Proposed consistency model

The model described above is feature value based, which restricts its application to Euclidean space. If we transform its description from ”feature value based” to ”feature distance based", the model can be enhanced. When we select and extract features to identify true and false objects, we implicity use the distances between features to distinguish objects instead of using the feature values themselves. Thus, a distance measure in some form is always involved in classification. We generalize model (3.1) as follows:

$$d_i(a) = \frac{M_{DM_i}(g,f) - M_{DM_i}(g,g)}{\sqrt{\sigma^2_{DM_i}(g,g) + \sigma^2_{DM_i}(g,f)}}, \tag{3.2}$$

where DM_i is the distance measure associated with feature i. Different features may have different distance measures, such as Euclidean norm, DTW and correlation.

$M_{DM_i}(C_1,C_2)$ is the mean of the feature distances (by distance measure DM_i) between pairwise objects in class C_1 and class C_2. Formally,

$$M_{DM_i}(C_1,C_2) = \frac{1}{|C_1||C_2|} \sum_{c_1 \in C_1, c_2 \in C_2, c_1 \neq c_2} DM_i(c_1,c_2), \tag{3.3}$$

($DM_i(c_1,c_2)$ denotes the distance of feature i between object c_1 and c_2; $|\cdot|$ denotes the cardinality.)

g represents a set of genuine signatures and f represents a set of forged signatures. $\sigma^2_{DM_i}(g,g)$ denotes the variation of the feature distances (by distance measure DM_i) within genuine signatures and $\sigma^2_{DM_i}(g,f)$ denotes variation of the feature distances (by distance measure DM_i) between genuine and forged signatures.

Our model takes into account the fact that the features always come with a certain kind of distance measure. So, we use DM_i to denote the distance measure associated with feature i. We compute the mean of feature distances instead of the feature values themselves because it is the distance that discriminates objects. The inter-class mean distance $M_{DM_i}(g,f)$

is assumed to be larger than the intra-class mean distance $M_{DM_i}(g,g)$. The larger the difference between $M_{DM_i}(g,f)$ and $M_{DM_i}(g,g)$, the higher the disciminability of feature i. The intra-class distance variance $\sigma^2_{DM_i}(g,g)$ is an indicator of the consistency of feature i. Inter-class distance variance is $\sigma^2_{DM_i}(g,f)$. If both the inter-class and intra-class distances do not vary much, then it indicates that the feature i is reliable. As a whole, the value of $d_i(a)$ measures the consistency of feature i on subject a. It should be noted that the same feature can have different consistency values across subjects. Given a data set which consists of a set of subjects, we need to calculate the mean and the standard deviation of the consistency values from equation (3.2).

Commonly Used Features

We compare the consistency of 22 commonly used features. We briefly discuss these features and their associated distance measures. Readers are referred to the corresponding papers for details.

- *Coordinate sequences*. X, Y, $[X,Y]$ are often called *featureless features*. The lengths of these features are variable. Usually, DTW is used as distance measure because DTW allows elastic matching [52]. It is believed that re-sampling the sequences with uniform arc-length makes the matching more robust [56]. However, this hypothesis has not been tested adequately. We will compare the consistency of the coordinate sequences with and without re-sampling using our proposed model.

- *Speed sequences.* V (Speed), V_x (speed of the X-coordinate) and V_y (speed of Y-coordinate) can be derived from sequence $[X,Y]$ directly by subtracting neighboring points. From the speed sequence, acceleration V_a can be further derived.

- *Pressure, Altitude, Azimuth.* Pressure is a typical dynamic feature in on-line signatures. Some digitizing devices can capture additional information, such as the Azimuth (the clockwise rotation of cursor about the z-axis) and Altitude (the upward angle towards the positive z-axis).

- *Center of Mass $\bar{x}(l)$ and $\bar{y}(l)$, Torque $\overline{T}(l)$, Curvature-ellipse $s_1(l)$ and $s_2(l)$.* These five features were defined in [56]. Center of Mass is actually a Gaussian smoothed coordinate sequence. Torque measures the area swept by the vector of the pen movement. $s_1(l)$ and $s_2(l)$ measure the curvature ellipses based on moments. The associated distance measure is the cross-correlation (Pearson's r) weighted by the consistency of points.

- *$\overline{V}$ (average speed), $\overline{V}_{x^+}$ (average positive speed on X-axis), $\overline{V}_{y^+}$ (average positive speed on Y-axis), T_s (total signing duration).* Lee et al. [43] list two sets of scalar features. The four features have the highest preference in the first set. The distance measure is the Euclidean norm.

- $\cos(\alpha)$*,* $\sin(\alpha)$*, Curvature* β. α is the angle between the speed vector and the X-axis. The three features were proposed in [33]. Two other features δx and δy were also

Table 3.1: Commonly used features and corresponding distance measures.

#	Feature	Dist. Measure
1	X-coordinate: X	DTW
2	Y-coordinate: Y	DTW
3	Coordinates: $[X,Y]$	DTW
4	Speed: V	DTW
5	Speed X: V_x	DTW
6	Speed Y: V_y	DTW
7	Pressure: P	DTW
8	Acceleration: V_a	DTW
9	Altitude: A_l	DTW
10	Azimuth: Z_u	DTW
11	Center of Mass X: $\overline{x}(l)$	Weighted r
12	Center of Mass Y: $\overline{y}(l)$	Weighted r
13	Torque: $T(l)$	Weighted r
14	Curvature-ellipse: $s_1(l)$	Weighted r
15	Curvature-ellipse: $s_2(l)$	Weighted r
16	Average speed: $\overline{V}$	Euclidean
17	Average positive V_x: $\overline{V}_{x+}$	Euclidean
18	Average positive V_y: $\overline{V}_{y+}$	Euclidean
19	Total signing time: Ts	Euclidean
20	Curvature: β	DTW
21	Angle: $\sin(\alpha)$	DTW
22	Angle: $\cos(\alpha)$	DTW

proposed. δx and δy are the coordinate sequence difference, same as feature # 5 and # 6 in Table 3.1 respectively.

The features described above and the corresponding distance measures are summarized in Table 3.1. These features are only a small subset of features used in on-line signature verification.

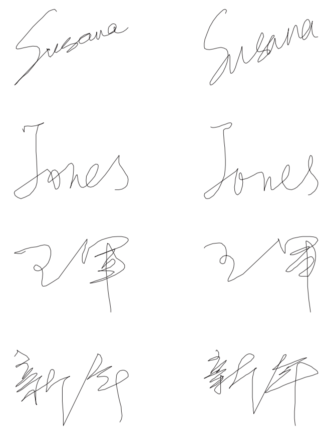


Figure 3.4: Examples of signatures from SVC. Signatures in the first column are genuine and those in the second column are forgeries.

Comparison of consistency

The SVC database [1] was used to calculate the consistency of the features in Table 3.1. SVC has two sets of signatures, namely task 1 and task 2. Each signature is represented as a sequence of points, which contains *X coordinate*, *Y coordinate*, *time stamp* and *pen status (pen-up or pen-down)*. In task 2, additional information like *Azimuth*, *Altitude* and *Pressure* are available. The information of time stamp and pen status was ignored in our experiments. There are 40 subjects in each task with 20 genuine signatures and 20 skilled forgeries for each subject. Examples of signatures from SVC are shown in Fig. 3.4.

To compare the consistency of different features, we normalize the raw signatures by the same preprocessing steps: 1) smooth the raw sequence by Gaussian filter, 2) rotate if necessary [55], 3) normalize the coordinates of each as follows:

$$X = \frac{X - min(X)}{max(X) - min(X)}, Y = \frac{Y - min(Y)}{max(Y) - min(Y)} \tag{3.4}$$

The feature distances are also normalized. We apply DM_i to class g (genuine signatures) and class f (forgeries). We first calculate all the pairwise feature distances within class g by DM_i. Then, we calculate all the pairwise feature distances between class g and f. Let the maximum feature distance be D_{max}. Every $dist$ is normalized by $e^{\frac{-dist}{2*D_{max}}}$ to map to a value between 0 and 1. The larger the distance, the closer the mapped value is to 0.

It is possible that the same feature may have different consistency values across subjects. Therefore, we calculate the average consistency of each feature and its standard deviation over subjects. Furthermore, to see the relation between feature consistency and its discriminatory ability in verification, the EER (Equal Error Rate) is calculated for each feature. EER is the point where the FRR (False Rejection Rate) equals the FAR (False Acceptance Rate). We choose the first 5 genuine signatures from each subject for reference templates and utilize the remaining 35 signatures for verification testing. Given a test signature S, the distances between feature i of S and the feature i of the 5 genuine signatures are determined. The maximum normalized distance is returned as the similarity output. To determine the EER, the threshold is varied from 0% to 100%. We have two modes of operation: (i) *universal* threshold, we choose the same threshold for all subjects and average the error rate and (ii) *user-dependent* threshold, we choose the optimal threshold for each subject.

The results are summarized in Table 3.2 in decreasing order of mean consistency value. We have the following observations:

- Although Z_u (Azimuth) and A_l (Altitude) have relatively high mean consistency val-

Table 3.2: Consistency (mean and standard deviation) and EER (universal threshold and user-dependent threshold) of the commonly used features.

Feature	Consistency		EER	
	Mean	Std.	Univ. T.	User. T.
Z_u	1.38	3.23	35.06%	26.63%
V_y	1.33	0.32	22.06%	11.38%
V_x	1.26	0.37	22.65%	16.81%
V	1.24	0.41	23.44%	17.63%
T_s	1.20	0.81	30.31%	28.18%
A_l	1.18	4.60	37.63%	29.06%
$\cos(a)$	1.12	0.28	26.72%	16.19%
$[X,Y]$	1.11	0.18	22.91%	16.83%
$\sin(a)$	1.10	0.25	29.56%	20.63%
V_a	1.10	0.32	29.29%	22.50%
P	1.06	0.38	36.86%	25.56%
β	0.99	0.18	28.90%	20.81%
Y	0.93	0.19	25.86%	18.68%
X	0.78	0.13	29.59%	25.13%
$\bar{y}(l)$	0.64	0.14	28.81%	19.19%
$\bar{x}(l)$	0.60	0.12	29.59%	23.00%
$T(l)$	0.53	0.15	33.63%	25.68%
$\overline{V}$	0.48	0.18	34.50%	31.94%
$\overline{V}_{y+}$	0.46	0.17	36.69%	33.88%
$\overline{V}_{x+}$	0.40	0.17	36.69%	34.00%
$s_2(l)$	0.36	0.03	43.86%	42.19%
$s_1(l)$	0.32	0.03	43.96%	42.13%
Average	0.89	0.57	31.30%	24.91%

ues, they also have high standard deviations (3.23 and 4.58 respectively), which means their discriminatory ability is not stable across subjects. Therefore, Z_u and A_l are not reliable features. The corresponding high EERs confirm this.

- Curvature-ellipse $s_1(l)$ and $s_2(l)$, Torques $T(l)$, Center of Mass $\bar{x}(l)$ and $\bar{y}(l)$ are of low consistency and thus high EERs. For instance, the mean consistency of $s_1(l)$ is only 0.32 and the EER with universal threshold is as high as 43.96%. We can conclude that these features are not suitable in detecting skilled forgeries, although they might be sufficient to detect random forgeries.

- Scalar features like $\overline{V}_{x^+}$ and $\overline{V}_{y^+}$ are too simple to carry enough discriminatory information. Their EERs are also relatively high. For $\overline{V}_{x^+}$, the EER (even with user-dependent threshold) is 34%. Scalar feature T_s also has high EER because its standard deviation of consistency is relatively high (0.82) compared to the average of all features (0.57). So, scalar features could be used to prune random forgeries but are not very reliable for accepting genuine ones.

- Pressure (P) is one of the most commonly used dynamic features. Since pressure is invisible to visual observation, it is difficult to forge. However, its mean consistency is not very high (1.06), which means that the pressure varies even for the same person's writing. Its relatively high EER indicates the following: large variation in pressure does not necessarily mean that the signature is a forgery, but very similar pressure variation is a strong indication that the signature is genuine.

- V_y (speed of the Y-coordinates) and V_x (speed of the X-coordinates) are among the most consistent features. Their mean consistency values are 1.33 and 1.26 respectively, while the EERs with universal threshold are both around 22%, which is low compared to the average (31.30%). V (speed), X, Y, $[X,Y]$, $\cos(a)$ and $\sin(a)$ are also reliable.

- EER has a negative relation with the level of consistency, although the EER does not strictly increase while the mean consistency decreases. The standard deviation of consistency also impacts the verification performance. The consistency of a feature is positively related to its performance of verification.

It must be noted that none of the EERs in Table 3.2 are low enough for real applications because we used only one feature at a time for verification in our experiments. How to combine these features optimally is an open problem. Further experiments on real and larger signature databases are necessary to claim the consistency of any given feature.

There is an unsubstantiated claim in on-signature verification community that the curve of the signature should be re-sampled with uniform arc-length [33, 55, 56]. Experiments were conducted to verify this claim. We re-sampled all the signatures to be of length N and calculated the consistency of features and corresponding EERs. The results are summarized in Table 3.3 with $N = 200$ points. We varied N and found the results had no significant change. Comparing the consistency values and EERs in Table 3.2 and Table 3.3, we can see that re-sampling does not necessarily improve performance. On the con-

Table 3.3: Consistency and EER of features with uniform arc-length re-sampling.

Feature	Consistency		EER	
	Mean	Std.	Univ. T.	User. T.
V_y	0.88	0.24	32.36%	20.81%
V_x	0.87	0.30	30.44%	21.00%
V	1.14	0.31	27.94%	17.63%
$\cos(a)$	1.08	0.24	29.41%	20.25%
$[X,Y]$	1.05	0.58	26.63%	20.94%
$\sin(a)$	0.81	0.27	32.56%	23.69%
β	0.89	0.24	32.19%	24.75%
Y	0.87	0.27	29.94%	21.56%
X	0.79	0.44	34.03%	29.63 %

trary, the performance is reduced to some degree. For example, the mean consistency of V_y decreased from 1.33 to 0.88 and EER increased from 22.06% to 32.36%, after re-sampling.

To conclude, comparative results show that the consistency of a feature is positively related to the feature's performance in signature verification. The results also show that some features such as the *coordinate sequence* (X, Y, $[X,Y]$), *speed* (V_y, V_x, V), and the angle α ($\cos(a)$, $\sin(a)$) have high consistency, and thus are reliable. We found that uniformly re-sampling the sequences does not necessarily increase verification performance. In the following experiments, we will use only the *coordinate sequence* X, Y as features.

3.4.2 ER^2 coupled with optimal alignment for On-line Signature Verification

We assume that signature verifications has two basic requirements: 1) At most 6 genuine signatures are provided and no forgeries are available for training. 2) Given signature to be verified, the output is a similarity score between 0% and 100% and not simply a ”yes/no”. The decision of rejection/acceptance is left to the system operator [56]. In real applications, these assumptions are reasonable. Since only few genuine signatures are available for enrollment, it is appropriate to use all of them as reference prototypes rather than statistically generating a single average prototype. When a signature is input for verification, we compare it with each of these prototypes and return the one with the highest similarity. ER^2 gives similarity measure with a confidence value between 0% and 100%. We briefly describe two signature verification algorithms here: 1) ER^2 coupled with optimal alignment; and 2) DTW-based Curve matching.

1). ER^2 coupled with optimal alignment

Enrollment. Given $K(K \leq 6)$ genuine signatures, $Sig_i = [X_i, Y_i], (i = 1, \cdots, K)$, we first preprocess each signature and save all of them in a template. Each signature here is a 2-dimensional sequence of the X-, Y- coordinates. Preprocessing includes the following steps: 1) Smooth the raw sequence by a Gaussian filter [56], 2) Rotate if necessary [55], and 3) Normalize each signature Sig_i by: $X_i = \frac{X_i - min(X_i)}{max(X_i) - min(X_i)}$, and $Y_i = \frac{Y_i - min(Y_i)}{max(Y_i) - min(Y_i)}$.

Matching. Given a signature $S = [X_s, Y_s]$, we preprocess it in the same way as we did during enrollment and match it against each of the preprocessed signatures in the template. The highest similarity score is returned. Matching S with Sig_i is done as follows. First, we use a function, namely $[dist, path] = DTW(S, Sig_i)$ to obtain the *path*. Note that we do not need the *dist* (DTW distance) here. Second, we stretch both S and Sig_i to have the same length according to the optimal alignment indicated by the *path*. Then, we calculate ER^2 by equation (2.8).

2). DTW-based Curve matching

Enrollment. Given $K (K \leq 6)$ genuine signatures, $Sig_i = [X_i, Y_i], (i = 1, \cdots, K)$, we first preprocess the signatures and save all of them in a template. Preprocessing is the same as above. The additional requirement is to find the maximum distance between these preprocessed signatures. We need it to calculate the similarity score. We can obtain this by calculating the pairwise DTW distances within the given samples and save the maximum (denoted as M_d) in the template.

Matching. Given a signature $S = [X_s, Y_s]$, we preprocess it in the same way as during enrollment and match it against each of the preprocessed signatures in the template. The highest similarity score is returned. Matching S with Sig_i is done as follows. First, we use function $[dist, path] = DTW(S, Sig_i)$ to obtain the *dist*

(Here, we do not need the $path$). Then, we translate $dist$ to similarity score by the formula, $exp(\frac{-dist}{2*M_d})$.

We can see that the only difference between above two algorithms is the mechanism of generating the similarity score. We have considered DTW for comparison because it is among the best methods for curve matching with optimal alignment. Alignment is absolutely necessary, because no user writes his/her signatures exactly the same each time. There always exists some difference in the total length and overall shape. Through experiments on signature verification, we will compare the accuracy brought by ER^2 and the DTW distance normalized by the Gaussian formula as $exp(\frac{-dist}{2*M_d})$. The reason we choose the Gaussian formula is that there is no statistical approach to directly translate DTW distance to a score given only a few genuine samples are available. Alternative curve matching based on DTW [55] is done by $dist = DTW(speed(S), speed(Sig_i))$, where $speed(S)$ transforms signature sequence S to speed sequence by $S_i = S_{i+1} - S_i, i = 1, \cdots, length(S) - 1$. Actually, we tried this approach and found the results were worse.

The experimental database we used was from [1]. Each signature is a 2-dimensional sequence. Totally, 80 subjects are available. For each subject, there are 20 genuine signatures and 20 skilled forgeries. We used the first 5 signatures for enrollment and the remaining 35 signatures were used in the test with skilled forgeries. Signatures from different subjects including genuine or skilled samples were considered as random forgeries.

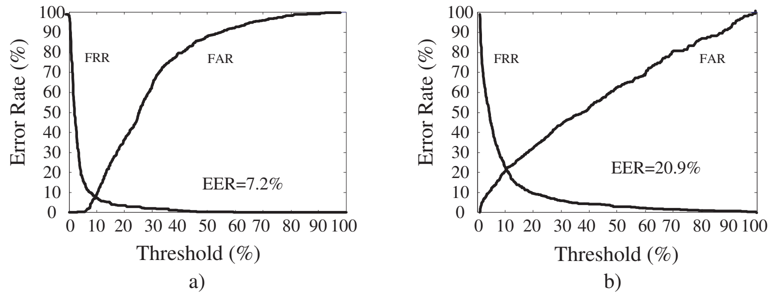


Figure 3.5: a) FRR and FAR curves by ER^2. b) FRR and FAR curves by DTW.

Table 3.4: EERs with universal/user-dependent threshold.

	Skilled Forgery		Random Forgery	
	DTW	ER^2	DTW	ER^2
Univ. T	20.9%	**7.2%**	5.7%	**0.9%**
User. T	10.8%	**4.9%**	1.3%	**0.2%**

Fig. 3.5 a) shows the curves of FAR (False Acceptance Rate) and FRR (False Rejection Rate) by ER^2 with universal threshold varying from 0% to 100%. The EER (Equal Error Rate) is 7.2%. Fig. 3.5 b) shows the results by DTW also with universal threshold. The EER is 20.9%, significantly higher than that obtained by ER^2.

We also tested on random forgeries. The results are summarized in Table 3.4. Both by universal threshold and user-dependent threshold, ER^2 coupled with optimal alignment outperforms DTW-based curve matching. It should be noted that if dynamic features such as speed, pressure and inclinations are used to prune forgeries here, the EER is expected to decrease further.

3.5 Chapter Summary

We present ER^2 as a similarity measure for multidimensional sequence matching. Signature verification systems can use ER^2 coupled with optimal alignment for intuitive similarity output and high performance. The experimental results are encouraging. Further evaluation on large and real databases is necessary. With direct similarity measure like ER^2, no feature extraction is necessary. Only the X-, Y- coordinate sequences in their origin representations are used for verification. We have implemented an on-line signature verification without explicit feature extraction.

Chapter 4

SIMILARITY-DRIVEN ON-LINE HANDWRITTEN CHARACTER RECOGNITION

On-line signature verification is only one of the possible applications of ER^2. Given a limited number of training samples, the verification task usually uses a simple nearest-neighbor classifier. In this chapter, we present another sequence classification application, on-line handwritten character recognition. The classifier chosen is the Support Vector Machines (SVM) as the application is not limited by paucity of training samples as in the case of signature verification. We will use the similarity measure ER^2 to speed up the decision-making process of SVM. In on-line handwriting recognition, the characters are represented by 2D sequences of the X-,Y-coordinates. Experiments on the benchmark database (UNIPEN) show that the classification driven by ER^2 can be about three times faster than standard SVM with comparable classification accuracy.

The intuition underlying ER^2 proves to be useful in the SVM classification. If pattern X has a high similarity with pattern Y (say, $ER^2(X,Y) = 90\%$), we could take it to be a strong indication that X belongs to the same class as Y.

A typical implementation of multi-class SVM is called ”One-vs-One” (OVO) [41].

OVO trains pairwise binary SVM classifiers. For M classes, there are $M(M-1)/2$ classifiers. On a test sample, each of the $M(M-1)/2$ classifiers casts one vote for its favored class. The class with the maximum votes wins. Given the advantage of the similarity measure ER^2, we do not have to run all the binary classifiers. As soon as the input pattern is found to have high similarity with samples from one class, an early decision can be made and the rest of the binary SVMs can be skipped. To make the recognition adaptive, a set of representative samples can be chosen from each class. By dynamically updating representative samples, the system saves the users' writing styles as seeds and adapts to them automatically.

We present the background of SVM briefly (a more detailed description is introduced in Chapter 5). First, we will discuss how ER^2 is seamlessly integrated into SVM for sequential pattern classification.

4.1 ER^2-driven SVM

SVM was originally designed for two-class (binary) classification. In many applications multi-class problems are encountered. A successful implementation of multi-class SVM is to decompose the multi-class problem into a series of sub-problems. Experiments have shown that, "One-Vs-One" (OVO), "One-Vs-All" (OVA) and DAGSVM [61] are among the methods suitable for practical use [32, 64]. In our experience, OVO is fast in training and has a slightly better classification accuracy than OVA and DAGSVM. Therefore, we choose OVO as an example to illustrate the ER^2-driven technique. The same procedure can be followed with OVA and DAGSVM.

To realize a speedup in the decision-making of SVM, a small set of *seeds* are chosen from each class. All binary SVMs are trained in the same way as regualr OVO SVM. To classify one input pattern X, the trained binary SVMs are evaluated one by one. After every binary classifier evaluation, the similarities between X and the seeds from the favored class are calculated to decide whether classification decision can be made or not. Suppose the classifier evaluates class i vs. class j and favors i, then X is matched against the seeds from class i and the maximum similarity (by ER^2) is returned. If the maximum similarity is $\geq T$ (T is a threshold, say $T = 90\%$), then an early decision is made that X belongs to class i. If the maximum similarity is $< T$, then we move to the next binary SVM evaluation.

Fig. 4.1 illustrates the seed samples in a binary SVM. The initial choosing of seed samples can be random, since the seeds will be gradually replaced by the samples from the feedback of users, which makes the classification adaptive. If no feedback is available, it is better to select seeds from the center of training samples.

4.2 Adaptive on-line handwriting recognition

Handwriting recognition has two modalities: off-line and on-line. Off-line refers to the case where the source of handwritten characters/words is scanned images, while on-line handwritten data is collected by a device which records the dynamic trajectory of the pen. On-line characters/words are usually represented by sequences of the X-, Y- coordinates together with pen down/up information. Sequence matching methods, such as DTW and

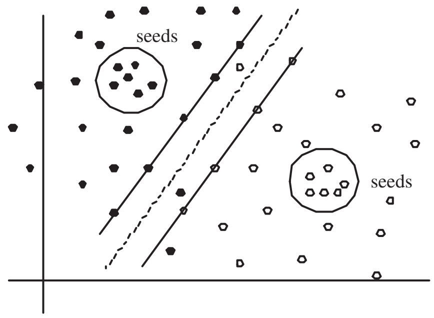


Figure 4.1: Binary SVM with seed samples. The points inside the circle are the seeds. The points on the solid lines are support vectors. The dotted line in the middle is the separating hyper-plane.

Euclidean norm, are commonly used in on-line handwriting recognition. Gaussian DTW kernel has been used in SVM for on-line handwriting recognition [6, 66]. DTW is defined as $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{DTW(\mathbf{x}_i, \mathbf{x}_j)}{\sigma^2}}$. However, the time complexity of DTW is $O(n^2)$ (n is the length of the sequence), which is computationally expensive, especially when the training dataset is large. Even if a band restriction is imposed on DTW to bring down its complexity to $O(wn)$ (w is the width of the band), it does not help significantly in the context of SVM. Moreover, during testing, the input sequence needs to run DTW with all the support vectors.

Since the writing style varies widely across subjects, it is desirable that the recognition system adapts to new writing styles automatically. The adaptation of on-line handwriting recognition is typically implemented with *prototype matching* [75]. The initial prototype set is formed by clustering handwritten samples from a large number of subjects. This is to allow the recognition system to handle a variety of writing styles. The system adapts to the user's writing style by modifying the prototypes. The ER^2-driven SVM supports this adaptation strategy nicely. The seed samples can be considered as the prototypes. The set of seeds can be updated by users' feedback. If one character written by a user is misclassified, then the character sample is inserted in the corresponding seed set. A complete pseudo-code description of the ER^2-driven SVM for adaptive on-line handwriting recognition is as follows.

Training of ER^2-driven SVM

Inputs: Training samples $\mathbf{V} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$; class labels $\mathbf{L} = [y_1, y_2, ..., y_N]$; M(number of classes); R(size of seed set for every class)

Outputs: Trained multi-class SVM model *OVO*; Seeds collection S

for i=1 to M **do**

  $C_i = \mathbf{V}(:, find(\mathbf{L} == i))$; /*data of class i*/

  $S\{i\}$=Random_Select(C_i, R); /*randomly choose R samples as seeds from C_i*/

end for

for i=1 to M **do**

  for j=i+1 to M **do**

    $C_1 = \mathbf{V}(:, find(\mathbf{L} == i))$; /*data of class i*/

    $C_2 = \mathbf{V}(:, find(\mathbf{L} == j))$; /*data of class j*/

    $OVO\{i\}\{j\} \leftarrow$ Train SVM on C_1 & C_2; /*save the binary model to OVO*/

  end for

end for/*end of training*/

Evaluation of ER^2-driven SVM

Inputs: Testing samples $\mathbf{V} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$; Trained multi-class SVM model *OVO*; *M*(number of classes); Seeds collection *S*; Similarity threshold *T*

Outputs: Labels $\mathbf{Y} = [y_1, y_2, ..., y_N]$; Update seeds collection *S*

for *k*=1 to *N* **do**
  $\mathbf{x} = \mathbf{V}(:, k)$; /*one testing sample*/
  Votes=zeros(1,*M*); /*votes for each class*/
  break_flag=0; /*indication of whether break happens*/
  for *i*=1 to *M* **do**
    Sim(*i*)=Max_ER^2($S\{i\}$,**x**); /*Determine the max similarity between **x** and seed set*/
  end for
  for *i*=1 to *M* **do**
    for *j*=*i*+1 to *M* **do**
      SVM $\leftarrow OVO\{i\}\{j\}$;
      label $\leftarrow$ SVM(**x**); /*binary SVM evaluation*/
      Votes(label)=Votes(label)+1; /*label=*i* or *j**/
      if Sim(label)$\geq T$ **then**
        Decision=label; /* make early decision*/
        break_flag=1; /*set break flag*/
        break; /*skip the rest binary evaluations*/

```
17:         end if
18:       end for
19:     end for
20:     if break_flag==1 then
21:       L(k)=Decision;
22:     else
23:       Y(k) = find(Votes == max(Votes)); /*the one with max votes wins*/
24:     end if
25:     S=Update_Seeds(S, True_Label, x); /* Update the seeds according to feedback*/
26: end for
```

The code presented above is in MATLAB style. In line 3 of the training part, we use a function *Random_select* for sake of brevity. It randomly chooses R samples as seeds for each class. Its implementation is trivial and therefore we omit the details. For the same reason, function *Max_ER*^{2} and *Update_Seeds* are used in line 6 and 25 of the evaluation part, respectively. Max_ER^2 returns the maximum similarity between the test input **x** and a set of seeds. The maximum similarities are determined in advance to avoid repeating similarity calculation. after binary SVM evaluation. *Update_Seeds* is to update the seed set according to the feedback from the system user(s). For example, if the user indicates that the true label of pattern **x** is i, **x** is inserted to the front of the seed set (queue) of class i and the seed at the end is removed from the set.

4.3 Sequence Classification Experiments

First, a small synthetic dataset was used to compare the performance of DTW kernel and RBF kernel. DTW is usually more robust than the Euclidean norm in sequence matching because DTW allows elastic alignment. We conducted an experiment to verify if the DTW kernel is more efficient than the Euclidean based RBF kernel in sequence classification. Experiments were also conducted to simulate the adaptive on-line handwritten character recognition on a benchmark database. The performance of the ER^2-driven SVM was compared with the standard OVO SVM.

4.3.1 RBF kernel vs. DTW kernel

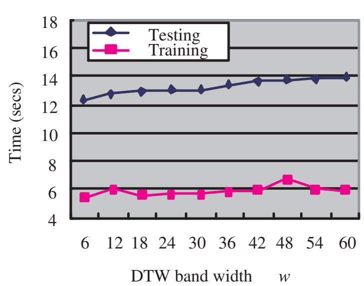
DTW and the Euclidean norm are both suitable and commonly used measures for sequence matching [35]. Euclidean norm is used in the RBF kernel, that is, $K(X_i, X_j) = exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2})$. Similarly, DTW has been used in the kernel as $K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{DTW(X_i, X_j)}{\sigma^2})$ [6, 66]. DTW is more robust than the Euclidean norm in a single pair of sequence matching. However, in the context of SVM, DTW kernel does not necessarily outperform the RBF kernel.

The Synthetic Control Time Series (SCTS) data sequences were used [31]. The dataset consists of 6 classes with 100 instances each. The first 70 sequences from every class were used for training, and the remaining were used for testing. The multi-class implementation method used was OVO. The software package, LIBSVM [12] was used with some modification. The regularizing parameters C and σ were determined via cross validation on the training set. The validation performance was measured by training on 70% of the training set and testing on the remaining 30%. The C and σ that lead to the best accuracy were selected. Experiments were conducted on a personal computer with 1G Hz CPU and 512 MB memory.

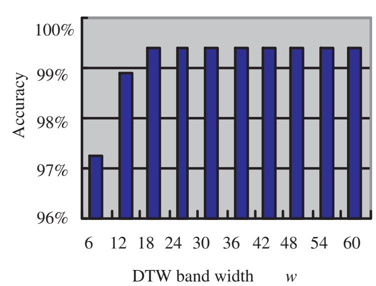
We compared the RBF kernel and the DTW kernel in three categories: accuracy, training time, and testing time. For DTW, there can be an additional parameter w (the width of band). We let w be the length of sequence (without band restriction). The results are summarized in Table 4.1. It shows that DTW can achieve the same level of accuracy as RBF. However, the training and testing time of DTW is about 10 times greater than RBF. Imposing the band restriction does not significantly reduce the computational time in SVM training or testing time. We varied the band width w from 6 to 60 (length of the SCTS sequence) in steps of 6. The training and testing times are shown in Fig. 4.2 a and the accuracies are shown in Fig. 4.2 b. The results show that both the training and testing time are insensitive to w. The accuracy increases slightly and stays at the highest level of 99.44% after $w \geq 18$. Therefore, DTW kernel does not outperform RBF kernel in terms of classification accuracy. Moreover, its training and testing are both extremely slow. RBF kernel works better, although Euclidean distance is usually not robust for matching a single pair of sequences. This is because the RBF kernel implicity maps the sequence to an infinite feature space where accurate separations can be achieved. In the following experiments, we use the RBF kernel in SVM.

Table 4.1: Comparison of RBF kernel and DTW kernel in accuracy, training time and testing time.

Kernel	σ	C	Accuracy	Training (secs)	Testing (secs)
RBF	10^2	10	99.44%	0.539	0.148
DTW	10	5	99.44%	6.59	13.98



a)



b)

Figure 4.2: a) The training and testing time comparison between RBF kernel and DTW kernel on the SCTS dataset while the band width w varies from 6 to 60 by step 6. b) Classification accuracies of DTW kernel on the SCTS dataset while the band width w varies from 6 to 60.

4.4 Adaptive On-line Handwritten Character Recognition Simulation

The simulations were carried out on the UNIPEN data [29]. UNIPEN is the benchmark database for on-line handwriting recognition. Two subsets of UNIPEN were used: the isolated digits (DIGIT) and the uppercase characters (UPPER). The first 70% samples of each subset were used for training and the remaining 30% for testing. Each sample was initially a 3-dimensional sequence: X-, Y-coordinates, and pen down/up. The pen down/up indicates the segmentation of a written sample. For simplicity, we used only the X-, Y-coordinates and ignored the pen down/up information. Thus, each sample in the experiments was simply a 2-dimensional sequence. Every sequence was preprocessed as follows: 1) Re-sample the sequence to be 32 points with uniform arc-length. That is, within one sequence, the Euclidean distances of two neighboring points are all equivalent. 2) Suppose the sequence is represented by $[X,Y]$, we normalize each sequence by: $X = \frac{X - min(X)}{max(X) - min(X)}$, and $Y = \frac{Y - min(Y)}{max(Y) - min(Y)}$. 3) Concatenate X- and Y-coordinates so that the SVM can treat them as a 1x64 vector.

Experiments were conducted on the DIGIT, UPPER and DIGIT+UPPER with and without the use of ER^2 (Table 4.2). The standard OVO SVM was tuned (parameters C and σ) by cross-validation on the training dataset and run on the testing dataset. Both accuracy and total testing time are recorded. ER^2 is then plugged in and the accuracy and testing time are again recorded. The threshold T of similarity is set at 90% empirically, since when $ER^2 \geq 90\%$, we consider that as sufficient evidence that the two patterns belong to the

Table 4.2: Description of the multi-class datasets used in the experiments.

Name	# of Training Samples	# of Testing Samples	# of Classes	Length
SCTS	420	180	6	60
DIGIT	10505	4704	10	64
UPPER	18219	7904	26	64

same class. We have not encountered any case where two sequences from different classes have similarity $\geq 90\%$. In rare cases, two sequences from different classes have similarity above 85%. Another parameter for ER^2-driven technique is the size of the seed set R for each class. Initially, we set $R = 5$. Then, we varied R to see how the performance was affected.

To simulate the feedback from system users, the seeds were continuously updated after the classification decision was made. With the feedback of true label, the classifier adds the testing sample to the seed set of the corresponding class as the newest seed. To avoid the overflowing of the seed set, the oldest seed is removed whenever a new seed is added. The updating is implemented by the function Update_Seeds (line 25 in the evaluation algorithm of the ER^2-driven SVM).

To measure the speedup in the decision-making of SVM achieved by the ER^2, *Reduc-*

tion Ratio was defined as:

$$\text{Reduction Ratio} = \frac{\text{\# of skipped binary SVMs}}{\text{Total \# of binary SVMs w.o. } ER^2}. \tag{4.1}$$

For a M-class problem, the total number of binary SVM evaluations is $M(M-1)/2$ without ER^2. If a break takes place because of high similarity in the middle, the number of skipped SVMs is $M(M-1)/4$, thus, roughly half the testing time is saved.

Table 4.3 summaries the results when $R = 5$ and $T = 90\%$. The ER^2-driven SVM speeds up by skipping binary SVM evaluations. The Reduction Ratio is around 69%, which means the speedup is about as $1/(1-0.69) \approx 3$ times. This is further confirmed by the testing times shown in the same table. The overhead of computation introduced by the seeds is not significant, because only 5 seeds are used. The number of support vectors from each class is on average a few hundred. For example, in DIGIT, the average number of support vectors for each class after SVM training is 165. Besides the testing speed, ER^2 also improves the accuracy by 1.38% on the DIGIT and 0.69% on DIGIT+UPPER sets. On UPPER, the accuracy of ER^2 is slightly lower than OVO SVM (by 0.64%).

Fig. 4.3 a and b show the accuracies and the Reduction Ratio when the size of the seed set R varies from 3 to 30 in steps of 3. The Reduction Ratio steadily increases from 60% to 80% while R increases. The accuracy decreases slightly from 91.5% to 90.07%, but always stay higher than 90.18% (accuracy of the standard OVO SVM+. R between 3 and 6 is a suitable choice in practice. It drives the decision-making of SVM about three times faster and can also give a slight accuracy improvement.

With the help of the feedback mechanism described above, we assume that the recog-

Table 4.3: Performance comparison between ER^2-driven SVM ($R = 5$, $T = 90\%$) and OVO SVM.

	σ	C	Accuracy	Testing (secs)	Reduction Ratio
DIGIT					
OVO SVM	2	5	90.18%	7.79	–
ER^2 SVM	2	5	**91.56%**	**2.90**	**67.28%**
UPPER					
OVO SVM	2	5	86.66%	40.16	–
ER^2 SVM	2	5	**86.02%**	**14.42**	**69.09%**
DIGIT+UPPER					
OVO SVM	2	5	81.78%	139.14	–
ER^2 SVM	2	5	**82.47%**	**45.51**	**69.53%**

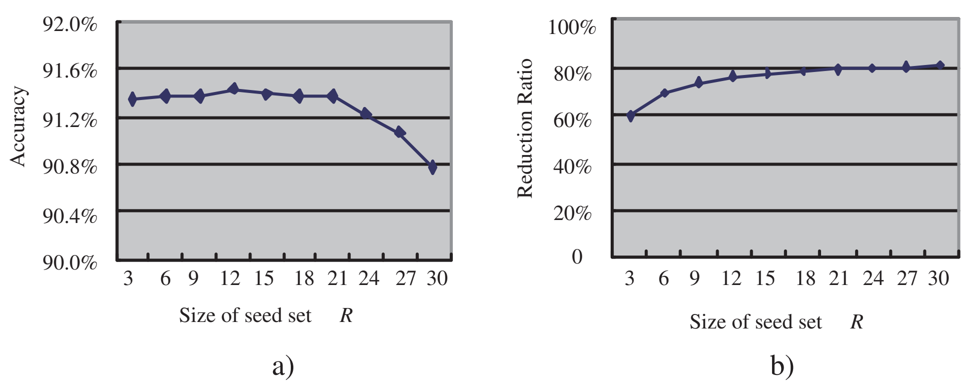


Figure 4.3: a) The classification accuracy decreases slightly while the size of seed set increases from 3 to 30 by step 3. b) The Reduction Ratio increase steadily while the size of seed set increases from 3 to 30 by step 3.

nition system is used interactively by all the users who provided the testing samples. The UNIPEN datasets were donated by a variety of institutes with many writers. From a practical viewpoint, only a limited number of users use the recognition system interactively. The dataset UPPER was used to study the recognition performance when the number of users is restricted.

The testing samples were extracted in the order of the writers, therefore, we assume that the testing dataset is organized as a list , [samples from writer 1, sample from writer 2, ...]. We divided the testing dataset of UPPER into 10 continuous parts and each part is considered as a set of testing samples from roughly 1/10 writers. We choose the first p (p=1, ..., 10) parts for testing to evaluate the performance (accuracy and Reduction Ratio). We simulated the performance when the system was used by p tenths users. For every p varying from 1 to 10, we applied the same processing steps and compared the performance of ER^2-driven SVM with OVO SVM. The results are shown in Fig. 4.4 a and b. For $p < 8$, the accuracy of ER^2 stays higher than that of OVO SVM. When $3 \leq p \leq 7$, ER^2 significantly outperforms OVO SVM. As a whole, fewer the number of writers, higher is the recognition accuracy observed. On the other hand, the Reduction Ratio gained by ER^2 stays between 58% and 70%, which means that the speedup is between 2-3 times. With greater number of writers, a higher speedup can be achieved.

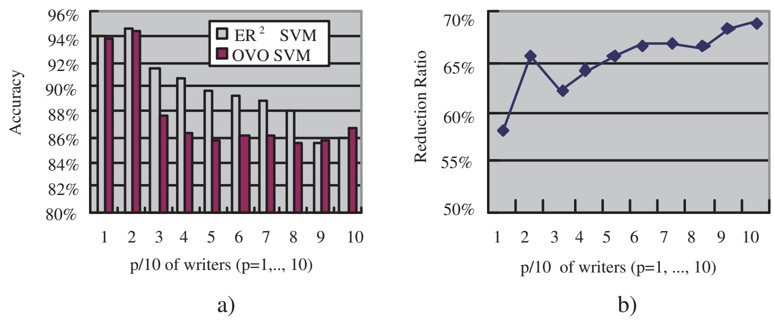


Figure 4.4: a) The accuracy comparison between ER^2-driven SVM and OVO SVM while the number of writers varies from 1 tenth to 10 tenths. b) The Reduction Ratio gained by ER^2 while the number of writers varies from 1 tenth to 10 tenths.

4.5 Chapter Summary

2D shape recognition problems, such as on-line handwriting, can benefit from ER^2. ER^2 speeds up conventional SVM by making early classification decision when the input pattern is found to be similar to the samples of a certain class. For sequence classification based on SVM, RBF kernel is more suitable for practice than the DTW kernel, although DTW is usually more robust than the Euclidean norm. Results on adaptive on-line handwritten character recognition are encouraging. The writer-dependent character recognition accuracy was above 94% with no any explicit feature extraction.

Chapter 5

OFF-LINE HANDWRITTEN DIGIT RECOGNITION

In chapter 4, we described featureless on-line handwritten character recognition without explicit feature extraction using ER^2 and SVM. In this chapter, we present off-line handwritten digit recognition by considering digit to be sequences of pixels.

To avoid feature extraction, we use digits in their original form as input sequences to the SVM classifier. By combining Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) into multi-class SVM for feature reduction and automatic feature selection, we reduce the data size. SVM is invariant under PCA transform, therefore PCA is a desirable dimension reduction method for SVM. RFE is a suitable feature selection method for binary SVM. However, it requires many iterations and each iteration needs to train SVM. This makes RFE prohibitively expensive for multi-class SVM (without PCA dimension reduction), especially when the training set is large. We have developed a method where PCA feature reduction is done before RFE when the number of features is large.

5.1 Support Vector Machines background

The Support Vector Machine (SVM) was originally designed for binary classification problems [9]. It maximizes the margin between classes by selecting a minimum number of Support Vectors (SV), which are determined by solving a Quadratic Programming(QP) optimization problem. The training of SVM is dominated by the QP optimization task which is slow. Speeding up of the QP problem and improving its scalability have been tackled by researchers [72, 58, 60]. Suppose we have N data points for training, then the size of the kernel matrix is $N \times N$. When N is even a few thousand (say, $N = 5000$), the kernel matrix is too large to reside in the memory of an average PC. This had been a challenge for SVM till it was recently solved by the Sequential Minimum Optimization (SMO) algorithm which reduces the space complexity for training to $O(1)$ [60]. This SMO approach has been used in machine learning, pattern recognition and data mining[80]. However, the fundamental SVM problem requires improvement in its testing phase [10, 20].

In this section, we introduce binary SVM and briefly describe how it can be extended for the multi-class problem. We also discuss methods to speed up the SVM testing phase.

5.1.1 Binary SVM

The basic SVM classifier can be expressed as:

$$g(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + b, \tag{5.1}$$

where the input vector $\mathbf{x} \in \Re^n$ and $\mathbf{w}$ is a normal vector of a separating hyper-plane in the *feature space* produced by the mapping of function $\phi(\mathbf{x}) : \Re^n \mapsto \Re^{n'}$. $\phi(\mathbf{x})$ can be linear or non-linear and n' can be finite or infinite. b is the bias. The sign of $g(\mathbf{x})$ determines the class of vector $\mathbf{x}$.

Given a set of training samples $\mathbf{x}_i \in \Re^n$, $i = 1, \cdots, N$ and corresponding labels $y_i \in \{-1, +1\}$, the separating hyper-plane (given by $\mathbf{w}$) is determined by minimizing the *structure risk* instead of the *empirical error*. Minimizing the structure risk is equivalent to finding the optimal margin between the two classes. The width of the margin is $\frac{2}{\mathbf{w}\cdot\mathbf{w}} = \frac{2}{\|\mathbf{w}\|^2}$. Taking into consideration the trade-off between structure risk and generalization, the training of SVM can be defined as an optimization problem:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{N}\xi_i \tag{5.2}$$

subject to

$$y_i(\mathbf{w}\cdot\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \forall i,$$

where parameter C is the trade-off and ξ_i is the slack variable in case some training points are non-separable. Fig. 5.1 illustrates the optimal margin and slack variables ξ_i.

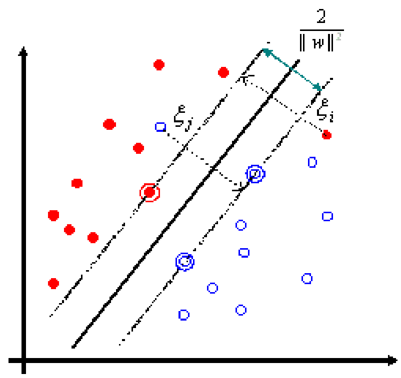


Figure 5.1: Optimal margin and slack variables in binary SVM. The support vectors are circled.

The solution to (5.2) is reduced to a QP optimization problem:

$$\max_{\boldsymbol{a}} \quad \boldsymbol{a}^T\boldsymbol{a} - \frac{1}{2}\boldsymbol{a}^T\mathbf{H}\boldsymbol{a} \tag{5.3}$$

subject to

$$0 \leq \alpha_i \leq C, \forall i,$$

$$\sum_{i=1}^{N} y_i\alpha_i = 0,$$

where $\boldsymbol{a} = [\alpha_1, \cdots, \alpha_N]^T$, and $\mathbf{H}$ is a $N \times N$ matrix, called the *kernel matrix*, with each element $\mathbf{H}(i, j) = y_i y_j \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$.

Solving the QP problem yields:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i), \tag{5.4}$$

$$b = -\sum_{k=1}^{N} \alpha_k y_k \phi(\mathbf{x}_k) \cdot \phi(\mathbf{x}_i) + y_i, \forall \alpha_i = 0. \tag{5.5}$$

Each training sample $\mathbf{x}_i$ is associated with a Lagrange coefficient α_i. Those samples whose coefficient α_i is nonzero are called *Support Vectors* (SVs). Only a small portion of the training samples become SVs.

Substituting eq. (5.4) in (5.1), we have the following formal expression for the SVM classifier:

$$g(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) + b \tag{5.6}$$
$$= \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b,$$

where the K is a kernel function: $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$. Given the kernel functions, we do not have to explicitly know $\phi(\mathbf{x})$. The commonly-used kernel functions are as following: 1) Linear kernel, i.e., $\phi(\mathbf{x}_i) = \mathbf{x}_i$, thus, $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{x}_i^T \mathbf{x}_j$; 2) Polynomial kernel, i.e., $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^d$, where c and d are some positive constants. 3) Gaussian Radial Basis (RBF) kernel, i.e., $K(\mathbf{x}_i, \mathbf{x}_j) = e^{(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})}$. If the kernel used is linear, the SVM is referred to as a *linear SVM*, otherwise, as a *non-linear SVM*.

The QP problem (5.3) involves a matrix $\mathbf{H}$ that has a number of elements equal to the square of the number of training samples. If there are more than 5000 training samples, $\mathbf{H}$ will not fit in a 128-Megabyte memory (assume each element is a 8-byte double) making the QP problem intractable by standard QP techniques. It was recently (1999) shown that the SMO algorithm produces an efficient solution to the problem[60]. SMO breaks the larger QP problem into a series of smaller QP problems. Solving these smaller QP optimizations analytically only needs $O(1)$ memory. Thus, the scalability of SVM is improved. More recent research focus has been on the evaluation (testing phase) of SVM [10] to improve accuracy and speed. The time complexity of binary SVM evaluation (eq. 5.6) is $O(d * N_{SV})$, where N_{SV} is the number of support vectors and d is the length of vector. The following are some possible ways in which we can explore to achieve speedup.

1. Reducing the total number of SVs can directly reduce the computational time of SVM in the testing phase. Burges et al. attempted to find a $\mathbf{w}'$ that approximated $\mathbf{w}$ in the feature space [10]. $\mathbf{w}'$ is expressed by a reduced set of vectors and corresponding coefficients (α_i'). However, the method for determining the reduced set is computationally expensive. Exploring this idea further, Downs et al. found a method to identify and discard *unnecessary* SVs (those SVs which depend linearly on other SVs) while leaving the SVM decision function unchanged [20]. A reduction in SVs of around 40% was reported.

2. Reducing the size of the quadratic program can reduce the number of SVs indirectly [45]. A subset of training samples are preselected as SVs and the smaller QP problem is solved. A study by Lin et al.[50] showed that the standard SVM possesses higher generalization ability and the method of reducing SVs maybe suitable when there is a large training task or when a large portion of training samples become SVs.

3. In this chapter, we propose to reduce the length of the input vector while recursively eliminating features, guided by the Leave-One-Out (LOO) approach.

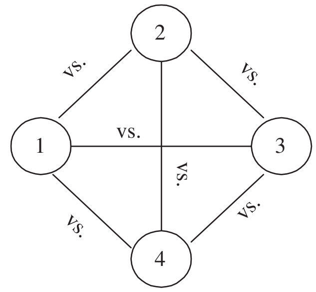


Figure 5.2: Implementation of Multi-class SVM by "One-Verse-One". For four class problem, six binary SVMs are trained to separate each pair of classes.

5.1.2 Multi-class SVM

There are two basic approaches for implementing multi-class SVM: (i) consider all the data in a single optimization function [15] or (ii) decompose the multiple classes into a series of binary SVMs, such as "One-Verse-All" (OVA) [73] and "One-Verse-One" (OVO) [41]. Although there are variances of OVO [61] and other sophisticated approaches for multi-class SVM, OVA and OVO are well suited for practice [32, 64].

For a K-class problem in OVO, K binary SVMs are constructed. The ith SVM is trained with all the samples of the ith class against the samples from all the other classes. Given a sample x to classify, K SVMs are evaluated and the label of the class that has the largest decision function value is chosen.

OVO is constructed by training binary SVMs between pairwise classes. Thus, OVO model consists of $\frac{K(K-1)}{2}$ binary SVMs for the K-class problem. Each of the $\frac{K(K-1)}{2}$ SVMs casts one vote for its favored class, and finally the class with most votes wins [41]. Fig. 5.2 shows an example of four-class OVO SVM.

Although OVA only requires K binary SVM, its training is computationally more expensive than OVO because each binary SVM is optimized on all the N training samples, while in OVO, each SVM is trained on $\frac{2N}{K}$ samples. The overall training speed of OVO is usually faster than OVA. We describe our speedup methods in the framework of OVO while observing that the same techniques can be applied to OVA SVM with minor adjustments.

5.2 *Speeding Up SVM Evaluation by PCA and RFE*

We speed up SVM evaluation by using the Principal Component Analysis (PCA) for dimension reduction and Recursive Feature Elimination (RFE) for feature selection. RFE has been combined with linear SVM in gene selection applications. We incorporate RFE into standard SVM (linear or non-linear) to eliminate the non-discriminative features and thus enhance SVM during the test phase. We consider applications such as handwriting recognition and face detection, where the features can be the pixels of the input images. Reducing the length of the feature vector while maintaining the accuracy is the objective. This is usually possible because not all features are discriminative. For example, to distinguish digits '4' and '9', the upper part of the digit (closed or open) is sufficient to tell them apart.

5.2.1 Recursive Feature Elimination (RFE)

RFE is an iterative procedure to remove non-discriminative features [30] in binary classification. The framework of RFE consists of the following steps: 1) Train the classifier; 2)

Rank the features based on their contribution to classification; and 3) Remove the feature with lowest ranking. Goto step 1) until no more features remain.

Algorithm.1 SVM-RFE[30]

Inputs: Training samples $\mathbf{X_0} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$ and class labels $\mathbf{Y} = [y_1, y_2, \cdots, y_N]$.

Outputs: feature ranked list $\mathbf{r}$.

1: $\mathbf{s} = [1, 2, \cdots, d]$; /*surviving features*/

2: $\mathbf{r} = []$; /*feature ranking list*/

3: **while** $\mathbf{s} \neq []$ **do**

4: $\mathbf{X} = \mathbf{X}_0(\mathbf{s}, :)$; /*only use surviving features of every sample*/

5: Train linear SVM on $\mathbf{X}$ and $\mathbf{Y}$ and obtain $\mathbf{w}$;

6: $c_i = w_i^2, \forall i$ /*weight of the ith feature in $\mathbf{s}$ */

7: $f = argmin_i(c_i)$; /*index of the lowest ranking*/

8: $\mathbf{r} = [\mathbf{s}(f), \mathbf{r}]$; /*update list*/

9: $\mathbf{s} = \mathbf{s}(1 : f-1, f+1 : length(\mathbf{s}))$; /*eliminate the lowest ranked feature*/

10: **end while**. /*end of algorithm*/

Note that the ranking criterion for the discriminability of features is based on $\mathbf{w}$, the normal vector of the separating hyper-plane (line 6). The idea is that a feature is discriminative if it significantly influences the width of the margin of the SVM. Recall that SVM tries to seek the maximum width of margin ($\frac{2}{\|\mathbf{w}\|^2} = 2/\sum_{i=1}^{n} w_i^2$).

RFE requires many iterations. If we have d features in total and want the top F features, the number of iterations is $d - F$ if only one feature is eliminated at a time. More than one feature can be removed at a time by modifying line 7-9 in the algorithm above, at the expense of possible degradation in performance.

5.2.2 Principal Component Analysis (PCA)

RFE works well in gene selection problems, because there are usually few (hundreds) training samples of gene expressions. In applications with large number of training samples like OCR, we use PCA to preprocess the data and perform dimension reduction before RFE.

The benefit of projecting $\mathbf{x}_i$ into the PCA space is that those eigenvectors associated with large eigenvalues are more ”principal”. The less important components can be removed thus reducing the dimension of the space. PCA is a suitable dimension reduction method for SVM classifiers because SVM is invariant under PCA transform. A proof is presented later.

5.2.3 Feature Reduced Support Vector Machines

Incorporating both PCA and RFE into a multi-class SVM, we propose the following algorithm in the OVO framework:

Algorithm 2 Training of FR-SVM

Input: Training samples $\mathbf{X_0} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$; class labels $\mathbf{Y} = [y_1, y_2, \cdots, y_N]$; M(number of classes); P(chosen principal components); and F(chosen top ranked features).

Output: Trained multi-class SVM classifier *OVO*; [V,D,P](the parameters for PCA transformation).

$[V,D] = PCA(X_0)$; /*Eigen vectors & values */

$\mathbf{X} = PCA\_Transform(V,D,\mathbf{X}_0,P)$; /*dimension reduction by PCA to P components */

for i=1 to M **do**

  for j=i+1 to M **do**

    $C_1 = \mathbf{X}(:,find(\mathbf{Y} == i))$; /*data of class i*/

    $C_2 = \mathbf{X}(:,find(\mathbf{Y} == j))$; /*data of class j*/

    $\mathbf{r} = \texttt{SVM-RFE}(C_1,C_2)$;   /*ranking list*/

    $Fc \leftarrow F$ top ranked features from $\mathbf{r}$;

    $C'_1 = C_1(Fc,:)$; /*only the F components*/

    $C'_2 = C_2(Fc,:)$;

    Binary SVM model $\leftarrow$ Train SVM on C'_1 & C'_2;

    $OVO\{i\}\{j\} \leftarrow \{$`Binary SVM Model, Fc`$\}$; /*save the model and selected features*/

  end for

end for/*end of algorithm*/

Algorithm 3 Evaluation of FR-SVM

Inputs: Evaluation samples $\mathbf{X_0} = [\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_N]$; Trained multi-class SVM classifier (*OVO*);

M(number of classes); [V, D, P](the parameters for PCA transformation).

Outputs: Labels $\mathbf{Y} = [y_1, y_2, \cdots, y_N]$.

$X = PCA\_Transform(V, D, X_0, P)$;
for k=1 to N **do**
  $\mathbf{x} = \mathbf{X}(:, k)$; /*one test sample*/
  Votes=zeros(1,M); /*votes for each class*/
  for i=1 to M **do**
    for j=i+1 to M **do**
      {`Binary SVM, Fc`} $\leftarrow OVO\{i\}\{j\}$;
      $\mathbf{x}' = \mathbf{x}(Fc)$; /*only the selected components*/
      Label $\leftarrow$ SVM($\mathbf{x}'$); /*binary SVM evaluation*/
      Votes(Label)=Votes(Label)+1;
    end for
  end for
  $\mathbf{Y}(k) = find(Votes == max(Votes))$; /*the one with max votes wins*/
end for/*end of algorithm*/

Algorithms 2 and 3 are for the training and evaluation of FR-SVM respectively. For every binary SVM, we apply RFE to obtain the top F most discriminative features (components). During the evaluation phase, we use only the selected features. The F features may be different for different pairs of classes. Instead of predefining the number of features F, an optimal F can be determined automatically by cross validation. This approach may be time-consuming when the training set or the number of classes is large. To make the FR-SVM more efficient, we let F be user-defined or determined by the Leave-One-Out (LOO) procedure, which will be described in the next section. Similarly, FR-SVM leaves P (number of chosen principal components) to be user-defined.

We will now prove that SVM is invariant under PCA transform.

Theorem. *(5.2) The evaluation result of SVM with linear, polynomial and RBF kernels is invariant if the input vector is transformed by PCA.*

Proof. Suppose $V = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_n]$, where $\mathbf{v}_i$ is an eigenvector. The PCA transformation of vector $\mathbf{x}$ is $V^T\mathbf{x}$. Recall the optimization problem (5.3). If the kernel matrix $\mathbf{H}$ does not change, then the optimization does not change (under the same constraints). Since $\mathbf{H}(i,j) = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, we simply need to prove invariance of $K(V^T\mathbf{x}_i, V^T\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$. Note that all eigenvectors are normal, i.e., $\mathbf{v}_i^T\mathbf{v}_i = 1, \forall i$ and mutually orthogonal, i.e., $\mathbf{v}_i^T\mathbf{v}_j = 0 (i \neq j)$. Therefore, we have $V^T V = I$, where I is a unit matrix.

Now, for the linear case, $K(V^T\mathbf{x}_i, V^T\mathbf{x}_j) = V^T\mathbf{x}_i \cdot V^T\mathbf{x}_j = (V^T\mathbf{x}_i)^T V^T\mathbf{x}_j = \mathbf{x}_i^T(VV^T)\mathbf{x}_j = \mathbf{x}_i^T I\mathbf{x}_j = \mathbf{x}_i^T\mathbf{x}_j = K(\mathbf{x}_i, \mathbf{x}_j)$. Similarly, we can prove for the polynomial case.

For RBF kernel, it is enough to show $\|V^T\mathbf{x}_i - V^T\mathbf{x}_j\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. Expanding the Euclidean norm, we have $\|V^T\mathbf{x}_i - V^T\mathbf{x}_j\|^2 = (V^T\mathbf{x}_i)^2 + (V^T\mathbf{x}_j)^2 - 2(V^T\mathbf{x}_i)^T(V^T\mathbf{x}_j) = \mathbf{x}_i^2 + \mathbf{x}_j^2 - 2\mathbf{x}_i\mathbf{x}_j = \|\mathbf{x}_i - \mathbf{x}_j\|^2$. $\square$

Given the theorem above, we can utilize PCA to preprocess the training samples and reduce the dimension of the feature space. This preprocessing is optional, if the original dimension of samples is not high (say, below 50). We use PCA to reduce the dimension before RFE because it leads to reduction of RFE iterations and saves the computational time of each iteration during training.

5.3 Leave-One-Out Guided Recursive Feature Elimination

In algorithm 2, the number of top ranked features F is decided empirically and can be adjusted by experiments. The feature selection problem is defined in two ways [76]: (i) given a fixed $F \ll d$, find the F features that give the smallest expected generalization error; or (ii) given a maximum allowable generalization error, find the smallest number of features, F. In both cases, the generalization error must be estimated. It has been shown that determining the minimum F in (ii) is NP-complete [18].

The performance of SVM can be measured by the generalization error. Cross-validation is a typical technique for estimating the generalization error. In the k-fold cross-validation, the training set is randomly divided into k subsets of equal size. The SVM is trained using $(k-1)$ subsets and tested on the left out subset. The procedure is repeated k times and the average test error is taken as the estimated generalization error. LOO error is an extreme case of k-fold cross-validation with $k =$number of training samples. Since LOO gives a nearly unbiased estimate of classifier performance [13], we use LOO as a measure to guide the feature elimination procedure. Intuitively, we can select the feature subset which gives the minimum LOO error. A major concern in the use of LOO is that it is expensive. To address this concern, there are two possible approaches. One is to derive error estimators that upper bounds the LOO error rate and the other is to compute the LOO directly but efficiently by managing the classifier training. There are several versions of LOO error bounds, such as Support Vector bound, Radius Margin bound and Span bound [74]. However, it has been shown empirically that these bounds are not sufficiently accurate for parameter tuning [21]. Since there exist efficient methods to compute LOO by modifying the SMO SVM training algorithm [44], we prefer to use LOO to determine the minimum feature subset.

In RFE, we can compute the LOO error and record the number of SVs in each round of feature elimination. When F varies from d to 1, we have a list of corresponding LOO rates. Thus, the F (and of course the corresponding subset of features) that yields the minimum LOO error rate can be determined. However, since our goal is to speed the evaluation of SVM, which is dominated by two factors (number of features and number of SVs), we also attempt to minimize the number of SVs. The following example will illustrate our technique.

Fig. 5.3 shows the SV bounds ($\frac{\text{\# of SVs}}{\text{\# of Training Samples}}$)[74], LOO error rates and test error rates on separating digits '0' and '9' in the MNIST database [42]. Feature elimination is carried out by removing 10% of the surviving features in each round of iteration. It can be seen that the SV bound is quite loose and therefore cannot accurately predict the minimum test error, while LOO is an accurate performance estimator. After the SV bound reaches its minimum at the 30th iteration (the circled point in Fig. 5.3), the test error increases significantly. Therefore, feature elimination after the 30th iteration is not reliable. Since

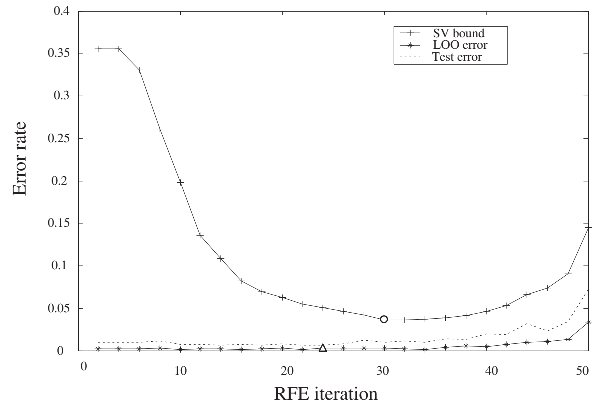


Figure 5.3: The SV bounds, LOO error rates and test error rates on separating digit '0' and '9' SVM with Gaussian kernel ($C = 5$ and $\sigma = 20$). The circle and triangle denote the minimum point of the SV bounds and LOO rates respectively.

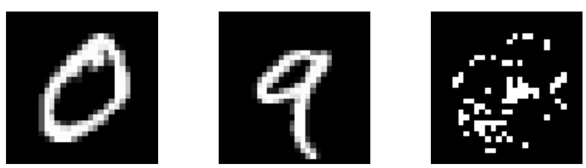


Figure 5.4: Digit '0', '9' and the selected 76 discriminative features that separate '0' and '9'. The triangle in Fig. 5.3 corresponds to the 23rd iteration after which the 76 features survived.

we prefer a smaller number of SVs, we can choose the minimum LOO before the lowest SV bound. In Fig. 5.3, we choose the triangled point which corresponds to the 23rd iteration where 76 features remain. These 76 features are sufficient to distinguish digit '0' and '9', as show in Fig. 5.4.

5.4 Experiments

The main purpose of the experiments is two fold: 1)Test the accuracy of FR-SVM. Since our primary goal is to enhance the SVM evaluation speed, we want to ensure that the performance of SVM is not comprised. 2) Observe the speedup that can be achieved without negatively influencing the accuracy of SVM.

Three datasets were used in our experiments (Table 5.1). Iris is a classical dataset for testing classification, available from the UCI KDD archives [31]. It has 150 samples with 50 in each of the three classes. We used the first 35 samples from every class for training and the remaining 15 for testing. The MNIST database [42] contains 10 classes of handwritten digits (0-9). There are 60,000 samples for training and 10,000 samples for testing. The digits have been size-normalized and centered in a fixed-size image (28×28). It is a benchmark database for machine learning techniques and pattern recognition methods. The third dataset, Isolet was generated from spoken letters [31]. Thus, the number of the classes varies from 3 to 26 in our experiments and the number of samples varies from hundreds (Iris) to tens of thousands (MNIST).

We used the OSU SVM Classifier Matlab Toolbox [51], which is based on LIBSVM [12]. On each dataset, we trained the multi-class OVO SVM using the RBF_kernel. The regularizing parameters C and σ were determined by cross validation on the training set. The validation performance was measured by training on 70% of the training set and testing on the remaining 30%. The C and σ that give the highest accuracy were selected. The FR-

Table 5.1: Description of the multi-class datasets used in the experiments.

Name	# of Training Samples	# of Testing Samples	# of Classes	# of Attributes
Iris	105	45	3	4
MNIST	60000	10000	10	784
Isolet	6238	1559	26	617

SVM was trained with exactly the same parameters (C and σ) and conditions as OVO SVM except that we varied two additional parameters P (the number of principal components) and F (the number of top ranked features). We compared the performance of OVO SVM and FR-SVM in three aspects: classification accuracy, training and testing speed.

5.4.1 PCA Dimension Reduction

Our first experiment involved varying P (let $F = P$) which implies PCA without feature selection. Since PCA reduces dimensions before SVM training and testing, PCA is able to improve both the training and testing speed. Table 5.2 summarizes our results on the Iris dataset. The first row with $\emptyset$ indicates that there was no PCA transformation, i.e., it shows the results of standard OVO SVM. The execution time of PCA (2nd column) is separated from the regular SVM training time (3rd column). The total training time of FR-SVM is the sum of PCA and SVM training times.

Since the Iris set is small, the contribution of PCA on training and testing time was not observed. However, it is interesting to note that the accuracy is not affected when $P = 4$ or $P = 3$. And even when $P = 1$, the accuracy is still high (97.78%).

The results on the MNIST and Isolet are shown in Table 5.3 and 5.4 respectively. It is encouraging that the PCA dimension reduction improves the accuracy of SVM on the MNIST dataset. When $P = 50$, the accuracy of FR-SVM is 98.30%, while that of OVO SVM is 97.90%. When $P = 25$, a speedup of 11.5 (278.9/24.3) in testing and 6.8 (4471.6/(228.6+441.4)) in training is achieved. The speedups are in line with our expectations, since the dimensions of samples were reduced before SVM training and testing. The results on the Isolet are similar, as shown in Table 5.4. The difference for MNIST is that the accuracy of FR-SVM decreases as P decreases, but not significantly before $P = 100$. The computational time of training and testing is saved significantly by PCA dimension reduction. An interesting observation on Isolet is that the training time is reduced from 249.4 (secs) to 86.5 (secs) with just PCA transformation and no dimension reduction (when $P = 617$). This could be possibly traced back to the implementation of the SVM optimization toolbox. From Tables 5.2, 5.3 and 5.4, we can see that the accuracy of SVM remains the same with PCA transformation but without dimension reduction (when $P =$the number of original dimensions). This validates our theorem 5.2.

Table 5.2: The results of FR-SVM on the Iris(without RFE feature selection). P is the number of principal components chosen. The line with $\emptyset$ is the results of OVO SVM. $C = 500$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

P (F=P)	Accuracy	PCA (secs)	Training (secs)	Testing (secs)
$\emptyset$	100%	NA	0	0
4	100%	0.016	0	0
3	**100%**	0.015	0	0
2	97.78%	0.015	0	0
1	97.78%	0.015	0	0

5.4.2 RFE Feature Selection

The second experiment involved varying F without PCA transformation. Since the feature elimination procedure required many iterations, we fixed the number of features eliminated each time (1 on Iris, and 20 on MNIST and Isolet empirically). The accuracy, time of RFE, training time and testing time are shown in Table 5.5, 5.6 and 5.7 on the three datasets respectively. The first row shows the results of basic OVO SVM when no feature is eliminated. As in the previous experiment, the execution time of RFE is separated from the regular SVM training time. The total training time of FR-SVM is therefore the sum of RFE (2n column) and SVM training (3rd column) times.

On the Iris dataset, only RFE required additional time, since the dataset is small. We note that only one feature is sufficient to distinguish a pair of Iris classes (for $F = 1$, the

Table 5.3: The results of FR-SVM on the MNIST(without RFE feature selection). P is the number of principal components chosen. The line with $\emptyset$ is the results of basic OVO SVM. $C = 5$ and $\sigma = 20$ for both OVO SVM and FR-SVM.

P (F=P)	Accuracy	PCA (secs)	Training (secs)	Testing (secs)
$\emptyset$	97.90%	NA	4471.6	278.9
784	97.90%	258.6	4467.4	275.7
500	97.84%	255.8	4230.7	250.1
300	97.58%	252.4	3800.8	237.2
200	97.92%	243.6	3650.2	231.9
150	97.96%	237.5	2499.2	175.7
100	98.20%	234.3	1529.6	102.2
50	**98.30%**	230.1	628.8	**48.8**
25	**97.94%**	228.6	441.4	**24.3**
10	92.76%	228.0	128.7	12.8

Table 5.4: The results of FR-SVM on the Isolet(without RFE feature selection). P is the number of principal components chosen. The line with $\emptyset$ is the results of basic OVO SVM. $C = 10$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

P (F=P)	Accuracy	PCA (secs)	Training (secs)	Testing (secs)
$\emptyset$	96.92%	NA	**249.4**	36.7
617	96.92%	21.2	86.5	36.6
400	96.86%	19.7	66.1	24.5
300	96.86%	19.3	57.9	18.4
200	96.60%	18.6	43.5	12.7
150	96.54%	18.3	38.3	9.3
100	**96.28%**	18.0	33.4	**6.4**
50	95.51%	18.0	24.7	3.6
25	93.39%	18.0	20.3	2.3
10	81.21%	17.4	12.1	2.0

Table 5.5: The results of FR-SVM on the Iris(without PCA dimension reduction). F is the number of top ranked features chosen for each pair of classes. $C = 500$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

F	Accuracy (secs)	RFE	Training (secs)	Testing (secs)
4	100%	0	0	0
3	100%	0.016	0	0
2	100%	0.016	0	0
1	**100%**	0.016	0	0

accuracy is still 100%), as shown in Table 5.5.

On the MNIST, the accuracy steadily decreased as F decreased, but not significantly. Compared to PCA, the speedup gained in testing is quite close. In addition, we observe that RFE is computationally expensive because of its recursive iterations. To eliminate 684 features ($F = 100$), the number of iterations was 34 (684/20, 20 features eliminated at a time), which took 13.7 hours (49432 secs). Without PCA, the procedure of RFE was painfully long. From the results on the Isolet as shown in Table 5.7, we have similar observations.

5.4.3 Combination of PCA and RFE

The third experiment involved observing how the combination of PCA and RFE contributed to multi-category SVM classification. We chose the minimum P in the first experiment which guaranteed that the accuracy would be as high as the standard SVM. Then we chose

Table 5.6: The results of FR-SVM on the MNIST(without PCA dimension reduction). F is the number of top ranked features chosen for each pair of classes. $C = 5$ and $\sigma = 20$ for both OVO SVM and FR-SVM.

F	Accuracy	RFE (secs)	Training (secs)	Testing (secs)
784	97.90%	0	4471.6	278.9
500	97.87%	35306	3179.7	251.1
300	97.82%	45120	2535.3	243.6
200	97.74%	47280	1094.4	221.0
150	97.40%	49344	648.0	114.1
100	96.62%	**49432**	398.7	110.6
50	94.56%	49440	286.2	47.1
25	89.86%	49446	208.8	25.5
10	75.60%	52126	171.9	13.5

F to be as small as possible for comparison purposes. The parameters C and σ remained the same as in previous experiments. The results are shown in Table 5.8. Training speedup was calculated as $\frac{\text{Training time of OVO SVM}}{\text{Training time of FR-SVM}}$. Note that the training time of FR-SVM is the sum of PCA, RFE, and SVM training. Similarly, the evaluation speedup was $\frac{\text{Testing time of OVO SVM}}{\text{Testing time of FR-SVM}}$. On the Iris dataset, the training speedup of FR-SVM over OVO SVM was 1 because both execution times were negligible. On the MNIST dataset, FR-SVM achieved a speedup in both training and testing. The gain in training was due to the dimension reduction by PCA. The gain in testing was due to the combinatorial contribution of PCA and RFE. On the Isolet dataset, the training of FR-SVM is slightly slower than OVO SVM because of RFE (training speedup = 0.98). However, during testing, we observed a significant improvement (by an order of 10) with accuracy remaining comparable. When $P = 50$ and $F = 40$, the accuracy of FR-SVM on MNIST was 98.14%, which is higher than that of OVO SVM (97.90%). When $P = 200$ and $F = 60$, the accuracy of FR-SVM on Isolet was 96.28%, which is slightly lower than that of OVO SVM (96.92%). To summarize, PCA and RFE can significantly improve the evaluation speed of standard SVM while maintaining accuracy.

5.4.4 LOO error guided Feature Elimination

The minimum size of the feature subset for each binary classification can be automatically determined by the LOO error, as discussed in section 4. Here we use the Isolet database to evaluate the technique of LOO guided RFE. We compared three strategies that determined F in algorithm 2: (i) using the minimum number of Support Vectors (min-SVs), (ii) using

Table 5.7: The results of FR-SVM on the Isolet(without PCA dimension reduction). F is the number of top ranked features chosen for each pair of classes. $C = 10$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

F	Accuracy	RFE (secs)	Training (secs)	Testing (secs)
617	96.92%	0	249.4	36.7
400	96.98%	758.4	80.2	22.3
300	96.86%	868.6	50.0	15.7
200	96.60%	981.1	30.0	12.4
150	96.73 %	1065.3	23.6	11.5
100	96.28 %	1164.0	12.8	5.3
50	96.09 %	1284.8	4.6	3.1
25	95.2 %	1301.8	3.9	3.0
10	93.14 %	1314.7	2.6	2.8

Table 5.8: Combination of PCA and RFE. P is the number of principal components chosen. F is the number of top ranked features chosen for each pair of classes. Speedup is FR-SVM vs. OVO SVM.

Dataset	P	F	Accuracy	Training Speedup	Testing Speedup
Iris	3	3	**100%**	1	1.3
MNIST	50	40	**98.14%**	3.90	**10.9**
Isolet	200	60	**96.28%**	0.98	**11.1**

Table 5.9: Comparison of feature subset size determination strategies. The Isolet database was used. $C = 10$ and $\sigma = 200$ for both OVO SVM and FR-SVM.

F Strategy	Classifier size(KB)	Average F #	Average SVs #	Acc.
min-SVs	1825.6	31	15.7	95.38%
min-LOO	2223.9	16	24.2	95.38%
LOO+SVs	4752.0	53	18.8	96.02%
OVO SVM	17251.8	617	53.7	96.92%

the minimum LOO error rate (min-LOO), and (iii) choosing the minimum LOO error rate before the number of SVs reaches its minimum (LOO+SVs). The comparison was carried out in four aspects: the size of the trained classifier model; the average size of feature subset between a pair of classes; the average number Support Vectors between a pair of classes and the overall classification accuracy. The experimental results are shown in Table 5.9.

The LOO+SVs strategy yields more accurate results (96.02%) than LOO or SVs alone (both 95.38%), although the latter two strategies can achieve more concise feature subsets and consequently, more compact classifier (size-wise). Compared to OVO SVM, LOO+SVs achieves about 3.6 (17251.8/4752.0) times classifier compression. The average discriminative feature subset between each pair of classes can be as low as 53 out of 617 (by feature elimination). Moreover, the average number of SVs is only 18.8, which is about $53.7/18.8 \approx 2.9$ times less than the standard OVO SVM. Therefore, a significant speedup of classification is achieved at the sacrifice of slight performance degradation (the difference between accuracy 96.92% and 96.02% is only 14 errors out of 1559 test samples).

5.5 Chapter Summary

Effective off-line handwritten digit recognition is implemented by incorporating both PCA and RFE into the standard SVM. The accuracy of FR-SVM can be high (98.14%, Table 5.8). With FR-SVM, no feature is explicitly defined. PCA reduces dimensions and RFE selects the most discriminative features. The ”features” are just original data with simple preprocessing. By choosing proper number of principle components and features for pairwise classes, significant speedup in SVM evaluation can be achieved while maintaining accuracy. It should be noted that the FR-SVM is not only applicable to handwritten data, regular classification problems can also benefit from PCA and RFE.

Chapter 6

GENERALIZED REGRESSION MODEL FOR SEQUENTIAL PATTERN CLUSTERING

In the previous chapters, the classification problems we discussed are actually supervised classification, where the labels of training samples are available. In this chapter, we discuss the unsupervised problem of sequential pattern clustering. Again, no explicit feature extraction is performed. The original attributes of the data are input as features. We propose a Generalized Regression Model (GRM) to tackle the sequence clustering challenges.

Linear relation has been found to be valuable in rule discovery of stocks, such as, if stock X goes up a, stock Y will go down b. The traditional linear regression models the linear relation of two sequences faithfully. However, if a user requires clustering of stocks into groups where sequences have high linearity or similarity with each other, it is prohibitively expensive to compare sequences one by one. We present Generalized Regression Model (GRM) to match the linearity of multiple sequences at a time. GRM also gives strong heuristic support for graceful and efficient clustering.

6.1 Generalized Regression Model

6.1.1 Why not the traditional Regression Model.

As discussed in Chapter 2, the Simple Regression Model is well suited for testing the linear relation of two sequences and R^2 is a good measure for linear relations. For example, $R^2(X_1, X_2) = 0.95$ is strong evidence that the two sequences X_1 and X_2 are linearly related to each other. When we need to test only two sequences, the Simple Regression Model is adequate. However, when more than two sequences are involved as in the task of clustering, regression must be performed between each pair of sequences. We cannot directly compute R^2 from multiple sequences, because R^2 in Multiple Regression means the linear relation between Y and the *linear combination* of X_1, X_2, $\cdots$, X_K. Moreover, R^2 in the Multiple Regression is sensitive to the order of sequences. If we randomly choose X_i to substitute for Y as the dependent variable and let Y be independent variable, then the regression becomes $X_i = \beta_0 + \beta_1 X_1 + \cdots + \beta_i Y + \cdots + \beta_K X_K + u$. The R^2 here is therefore different from that of model (2.6). Model (2.6) describes a hyper-plane instead of a line in $(K+1)$-dimensional space which is what weed need too test the linearity (similarity) among multiple sequences.

Generalizing the Simple Regression Model to multiple sequences, we propose the Generalized Regression Model (GRM) to address the issues described above.

6.1.2 GRM: Generalized Regression Model.

Given $K(K \geq 2)$ sequences $X_1, X_2, \cdots, X_K$ and

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_K \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ x_{K1} & x_{K2} & \cdots & x_{KN} \end{pmatrix}$$

We organize them as N points in K-dimensional space:

$$p_1 = \begin{pmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{K1} \end{pmatrix}, p_2 = \begin{pmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{K2} \end{pmatrix}, \cdots, p_N = \begin{pmatrix} x_{1N} \\ x_{2N} \\ \vdots \\ x_{KN} \end{pmatrix}$$

The objective is to find a line in the K-dimensional space that fits the N points in the sense of minimum-sum-of-squared-error. In traditional regression, the error term is defined as:

$$u_i = y_i - (\beta_0 + \beta_1 x_{1i} + \cdots + \beta_K x_{Ki}) \tag{6.1}$$

It is the distance between y_i and the regression hyper-plane in the direction of the axis Y. This makes sequence Y different from $X_i (i = 1, 2, \cdots, K)$. Fig. 6.1 a) shows the u_i of the traditional regression in 2-dimensional space. In GRM, we define the error term u_i as the *vertical* distance from point $(x_{1i}, x_{2i}, \cdots, x_{Ki})$ to the regression line. It should be noted that there is no Y here anymore, as no sequence is special. Fig. 6.1 b) shows the newly defined u_i in two-dimensional space.

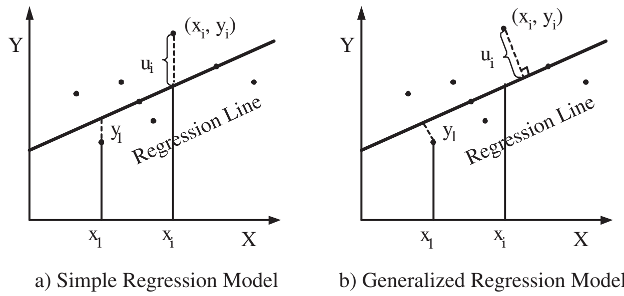


Figure 6.1: a) In the traditional Simple Regression Model, sequences Y and X compose a 2-dimensional space. $(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)$ are points in the space. The regression line is the line that fits these points in the sense of minimum-sum-of-squared-error. b) In GRM, two sequences also compose a 2-dimensional space. The error term u_i is defined as the vertical distance from the point to the regression line.

The Appendix of this chapter gives details on how to determine the regression line in K-dimensional space. For now, we assume we have found the regression line as follows:

$$\frac{p(1)-\overline{x_1}}{e_1} = \frac{p(2)-\overline{x_2}}{e_2} = \cdots = \frac{p(K)-\overline{x_K}}{e_K} \tag{6.2}$$

where $p(i)$ is the i-th element of point p and $[e_1,e_2,\cdots,e_K]^t$ is the eigenvector corresponding to the maximum eigenvalue of the scatter matrix (see Appendix). $\overline{x_j}(j=1,2,\cdots,K)$ is the average of sequence X_j.

If any point p in the K-dimensional space satisfies equation (6.2), it must lie on the regression line. For the N points $p_1,p_2,\cdots,p_N$, only some may lie on the regression line, but the sum of squared-distance from $p_j(j=1,2,\cdots,N)$ to the regression line is minimized. To guarantee the regression line exists and is unique, we need following two assumptions:

- **Assumption 1.** $\sum_{i=1}^{N}(x_{ji}-\overline{x_j}) \neq 0$. No sequence is constant. This guarantees the scatter matrix has an eigenvector.

- **Assumption 2.** There exist at least two points p_i, p_j among the N points such that $p_i \neq p_j$. This guarantees that the N points determine a unique line.

Since in practice, it is unlikely that a sequence is constant or all K sequences are exactly the same, these assumptions do not limit the applications of GRM.

As in the traditional regression case, we define a measure for the Goodness-of-fit:

$$GR^2 = 1 - \frac{\sum_{i=1}^{N} u_i^2}{\sum_{j=1}^{K} \sum_{i=1}^{N} (x_{ji} - \overline{x_j})^2} \tag{6.3}$$

If GR^2 is close to 1, then we know the regression line fits the N points well, implying that the K sequences have a high degree of linear relationship with each other.

We need the following two lemmas before we prove an important theorem of GRM.

Lemma. *(6.1.1) Two sequences X_1 and X_2 are linear to each other $\Leftrightarrow$ Two sequences X_1 and X_2 have a linear relation as $\frac{x_{1i} - \overline{x_1}}{\sigma(X_1)} = \frac{x_{2i} - \overline{x_2}}{\sigma(X_2)} (i = 1, 2, \cdots, N)$, where $\sigma(X)$ denotes the standard deviation of sequence X.*

Proof. Two sequences X_1 and X_2 are linear to each other implies

$$x_{2i} = \beta_0 + \beta_1 x_{1i} (i = 1, 2, \cdots, N) \tag{6.4}$$

So, we have:

$$\sum_{i=1}^{N} x_{2i} = N\beta_0 + \sum_{i=1}^{N} x_{1i} \tag{6.5}$$

That is:

$$\overline{x_2} = \beta_0 + \beta_1 \overline{x_1} \tag{6.6}$$

By subtracting (6.6) from (6.4), we get:

$$x_{2i} - \overline{x_2} = \beta_1 (x_{1i} - \overline{x_1}) \tag{6.7}$$

From (6.4) we know that $\sigma(X_2) = \sigma(\beta_0 + \beta_1 X_1) = \beta_1 \sigma(X_1)$, i.e., $\beta_1 = \sigma(X_2)/\sigma(X_1)$. Thus, (6.7) can be rewritten as $\frac{x_{1i} - \overline{x_1}}{\sigma(X_1)} = \frac{x_{2i} - \overline{x_2}}{\sigma(X_2)}$.

□

Lemma. *(6.1.2) K sequences $X_1, X_2, \cdots, X_K$ are linear to each other $\Leftrightarrow$ K sequences $X_1, X_2, \cdots, X_K$ have a linear relation as $\frac{x_{1i} - \overline{x_1}}{\sigma(X_1)} = \frac{x_{2i} - \overline{x_2}}{\sigma(X_2)} = \cdots = \frac{x_{Ki} - \overline{x_K}}{\sigma(X_K)} (i = 1, 2, \cdots, N)$, where $\sigma(X)$ denotes the standard deviation of sequence X.*

Proof. From lemma 6.1.1, this is obvious. □

Applying lemma 6.1.1 and 6.1.2, we can obtain the following theorem:

Theorem. *(6.1.2) K sequences $X_1, X_2, \cdots, X_K$ are linear to each other $\Leftrightarrow$ Points $p_1, p_2, \cdots, p_N$ are all distributed on a line in K-dimensional space and this line is the regression line.*

Proof. From right to left is obvious. Let's prove the implication from left to right.

According to lemma 6.1.2, if $X_1, X_2, \cdots, X_K$ are linear to each other, then:

$$\frac{x_{1i} - \overline{x_1}}{\sigma(X_1)} = \frac{x_{2i} - \overline{x_2}}{\sigma(X_2)} = \cdots = \frac{x_{Ki} - \overline{x_K}}{\sigma(X_K)} \tag{6.8}$$

Recall that point $p_i = [x_{1i}, x_{2i}, \cdots, x_{Ki}]^t$. Let $p_i(k) = x_{ki}$, (6.8) can be rewritten as:

$$\frac{p_i(1) - \overline{x_1}}{\sigma(X_1)} = \frac{p_i(2) - \overline{x_2}}{\sigma(X_2)} = \cdots = \frac{p_i(K) - \overline{x_K}}{\sigma(X_K)} \tag{6.9}$$

Equation (6.9) implies that point $p_i (i = 1, \cdots, N)$ is on the line. Therefore, we conclude that $p_1, p_2, \cdots, p_N$ are distributed on a line in K-dimensional space.

Now we need to show that the line is the regression line. The N points $p_1, p_2, \cdots, p_N$ determine the line (6.8) and it is unique (assumption 2). Also, our regression line guarantees that the sum of squared-error is minimized. If and only if the regression line is same as line (6.8), the sum of squared-error is minimized.

□

Now, let us derive some properties of GR^2:

1. $GR^2 = \frac{\lambda}{\sum_{i=1}^{N} \|p_i - m\|^2}$ and $1 \geq GR^2 \geq 0$.

2. $GR^2 = 1$ means the K sequences have an exact linear relationship with each other.

3. GR^2 is invariant to the order of $X_1, X_2, \cdots, X_K$, i.e., we can arbitrarily change the order of the K sequences and the value of GR^2 does not change.

Proof. 1. According to 7 Appendix, we have:

$$\sum_{i=1}^{N} u_i^2 = -e^t S e + \sum_{i=1}^{N} \| p_i - m \|^2 \tag{6.10}$$

Thus,

$$
\begin{aligned}
GR^2 &= 1-\frac{\sum_{i=1}^{N}u_i^2}{\sum_{j=1}^{K}\sum_{i=1}^{N}(x_{ji}-\overline{x_j})^2}\\
&= 1-\frac{\sum_{i=1}^{N}u_i^2}{\sum_{i=1}^{N}\parallel p_i-m\parallel^2}\\
&= \frac{e^tSe}{\sum_{i=1}^{N}\parallel p_i-m\parallel^2}(\text{recall }\ (6.10))\\
&= \frac{e^t\lambda e}{\sum_{i=1}^{N}\parallel p_i-m\parallel^2}\\
&= \frac{\lambda}{\sum_{i=1}^{N}\parallel p_i-m\parallel^2}(\text{recall that }\parallel e\parallel^2=1)
\end{aligned}
$$

From (6.10), we know $\sum_{i=1}^{N}u_i^2=-e^tSe+\sum_{i=1}^{N}\parallel p_i-m\parallel^2\geq 0$, therefore:

$$
\sum_{i=1}^{N}\parallel p_i-m\parallel^2\geq e^tSe=\lambda\geq 0 \tag{6.11}
$$

so we conclude $1\geq GR^2\geq 0$.

2. If $GR^2=1$, then $\sum_{i=1}^{N}u_i^2=0$, which means that the regression line fits the N points perfectly. Therefore, according to theorem 6.1.2, the K sequences have an exact linear relation with each other.

3. This is obvious.

□

Because GR^2 has the above important properties, we define GR^2 as the degree of linear relation and the similarity measure in the same sense as in a simple linear relation.

6.2 Applications of GRM

The procedure of applying GRM to measure the linear relation of multiple sequences is described in algorithm GRM.1.

GRM.1: Testing linearity of multiple sequences

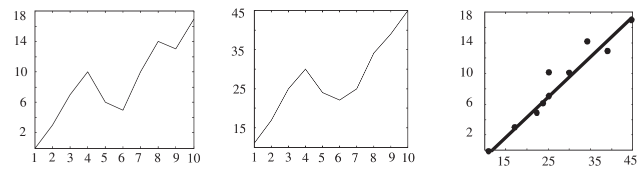
- Organize the given K sequences with length N into N points $p_1, p_2, \cdots, p_K$ in K-dimensional space as shown in §6.1.2.

- Determine the regression line. First, calculate the average $m = \frac{1}{N}\sum_{i=1}^{N} p_i$. Second, calculate the scatter matrix $S = \sum_{i=1}^{N}(p_i - m)(p_i - m)^t$. Then, determine the maximum eigenvalue λ and corresponding eigenvector e of S.

- Calculate GR^2 according to property 1 of GR^2.

- Let us say we only accept linearity with confidence no less than C (say, $C = 85\%$). If $GR^2 \geq C$, we can conclude that the K sequences are linear to each other with confidence GR^2 and the linear relation is as in (6.2).

In step 2), the eigenvector e (corresponding to the maximum eigenvalue λ) is the actually the first principle component by PCA analysis. However, the essential difference between GRM.1 and PCA is that GRM.1 first organizes the K sequences into N points and then uses the technique of PCA to determine the generalized regression line whose direction is indicated by the eigenvector e. Based on how well the N points are distributed



a) Sequence X_1 b) Sequence X_2 c) X_1 regresses on X_2

Figure 6.2: An example of applying GRM.1 to two sequences.

along the regression line, GR^2 is computed to measure the overall similarity of multiple sequences.

6.2.1 Apply GRM.1 to two sequences: an example.

Suppose we want to test two sequences X_1 and X_2 and

$$X_1 = [\ 0,\ 3,\ 7,\ 10,\ 6,\ 5,\ 10,\ 14,\ 13,\ 17];$$

$$X_2 = [11,\ 17,\ 25,\ 30,\ 24,\ 22,\ 25,\ 34,\ 39,\ 45],$$

as shown in Fig. 6.2 a) and b) respectively.

First, organize the two sequences into 10 points: (0, 11), (3, 17), (7, 25), (10, 30), (6, 24), (5, 22), (10, 25), (14, 34), (13, 39), (17, 45). If we draw the 10 points in the X_1-X_2 space, their distribution is as shown in Fig. 6.2 c).

Second, determine the regression line. Average $m = \frac{1}{N}\sum_{i=1}^{N} p_i = [8.5, 27.2]^t$. Maximum eigenvalue $\lambda = 1161.9$ and corresponding eigenvector $e = [0.4553, 0.8904]^t$.

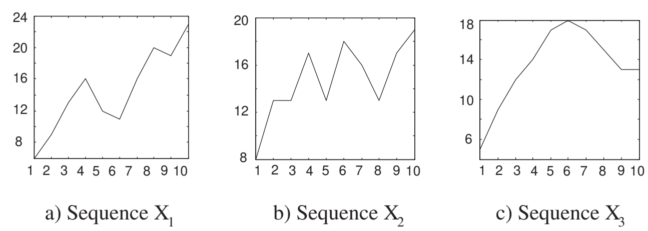


Figure 6.3: Three sequences with low similarity.

Third, calculate $GR^2 = 0.9896$. Since GR^2 is high (98.96%), we can conclude that X_1 is linearly related to X^2. In addition, we find their linear relation is $\frac{X_1-8.5}{0.4553} = \frac{X_1-27.2}{0.8904}$.

If we observe Fig. 6.2 a) and b), we can see they are visually similar (linear) to each other. So the conclusion based on GR^2 makes intuitive sense.

6.2.2 Apply GRM.1 to multiple sequences: an example.

GRM is intended to test whether multiple sequences are linear to each other or not. Let us consider an example for testing 3 sequences at a time.

Suppose we have three sequences, as shown in Fig. 6.3:

$$X_1 = [6,\ 9,\ 13,\ 16,\ 12,\ 11,\ 16,\ 20,\ 19,\ 23];$$

$$X_2 = [8,\ 13,\ 13,\ 17,\ 13,\ 18,\ 16,\ 13,\ 17,\ 19];$$

$$X_3 = [5,\ 9,\ 12,\ 14,\ 17,\ 18,\ 17,\ 15,\ 13,\ 13].$$

We can calculate $GR^2 = 0.7301$. This score is not very high, thus we can conclude that some sequences are not linear to others. If we observe the three sequences in Fig. 6.3, we can see that none of the three sequences are actually linear to each other. This example once again demonstrates that GR^2 is an intuitive measure.

6.2.3 Apply GRM to clustering of massive sequences.

We can make use of algorithm GRM 1 to obtain heuristic information for clustering sequences. From theorem 6.1.2, lemma 6.1.2 and the formula of the regression line equation (6.2), we know that if multiple sequences have linear relation with each other, then the eigenvector is directly related to the standard deviations, i.e., $\frac{\sigma(X_1)}{e_1} = \frac{\sigma(X_2)}{e_2} = \cdots = \frac{\sigma(X_K)}{e_K}$. Inversely, $\frac{\sigma(X_i)}{e_i} \approx \frac{\sigma(X_j)}{e_j}$ is a strong hint that they *may* be linear; if $\frac{\sigma(X_i)}{e_i}$ differs greatly from $\frac{\sigma(X_j)}{e_j}$, then they can not be linear. This is valuable hint for clustering. We call $\frac{\sigma(X_i)}{e_i}$ the feature value of sequence X_i.

Based on this, we derive an algorithm for clustering large number of sequences. Given a set of sequences $S = \{X_i \mid i = 1, 2, \cdots, K\}$, algorithm GRM.2 is as follows.

Algorithm GRM.2: Clustering of massive sequences

- Apply Algorithm GRM.1 to test whether the given sequences are linear to each other or not. If the answer to the test is yes, all the sequences can go into one cluster and we can stop, otherwise, go to next step.

- After GRM.1, we have eigenvector $[e_1, e_2, \cdots, e_K]^t$. Create a feature value sequence $F = (\frac{\sigma(X_1)}{e_1}, \frac{\sigma(X_2)}{e_2}, \cdots, \frac{\sigma(X_K)}{e_K})$ and sort it in increasing order. After sorting, let $F =$

$(f_1, f_2, \cdots, f_K)$.

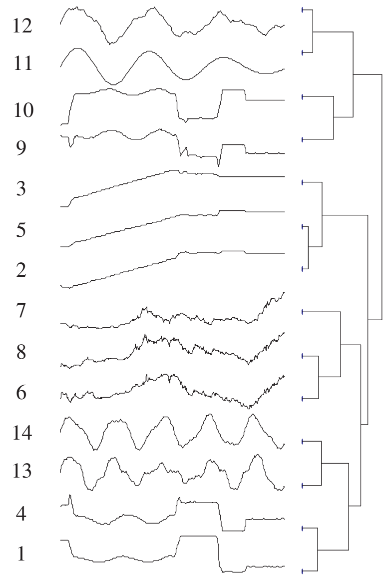
- Start from the first feature value f_1 in F. Suppose the corresponding sequence is X_i. We only check the linearity of X_i with the sequences whose feature values in F are close to f_1. Here "close" means $f_j/f_1 \leq \xi$. We take those sequences which have linearity with X_i with confidence $\geq C$ and put them into cluster CM_1. Delete all the sequences in this cluster from set S, and repeat the procedure to obtain the next cluster until S becomes empty.

Fig. 6.4 shows an example of clustering the Reality-Check sequences [8]. The data set consists of 14 sequences, from Space Shuttle telemetry, Exchange Rates and artificial data. Fig. 6.4 b) shows the sequence number and the corresponding feature values. After sorting the feature value, we can see that similar sequences are close to each other. For instance, sequence # 2,# 3,# 5 have close feature values as shown in Fig. 6.4 b) and the final clustering result confirms that they are similar to each other as shown in Fig. 6.4 a).

The most time-consuming part in GRM.1 and GRM.2 is the calculation of the maximum eigenvalue and corresponding eigenvector of scatter matrix S. Fast algorithm [19] can do calculate these with high efficiency.

6.3 Experiments

Our experiments focus on two aspects: 1) Is GR^2 a good measure for linearity? If two or more sequences have a high degree of linearity with each other, will GR^2 be close to 1



#	f
9	-1.1088
12	-0.6057
11	-0.6000
13	-0.5158
14	-0.4960
10	-0.4317
1	0.0654
4	0.1571
6	0.4382
8	1.1410
7	1.1608
3	1.4671
5	1.5452
2	1.5647

Figure 6.4: a) The clustering of the Reality-Check sequences by GRM.2. b) Sequence numbers and corresponding feature values in increasing order.

and vice versa? 2) Can GRM.2 be used to mine linear stocks in the market? What is the accuracy and performance of GRM.2 in clustering sequences?

We generated 5000 random sequences of real numbers. Each sequence $X = [x_1, x_2, \cdots, x_N]$ is a random walk: $x_{t+1} = x_t + z_t, t = 1, 2, \cdots$, where z_t is an independent, identically distributed (IID) random variable. Our experiments conducted on these sequences show that GR^2 is a good measure for linearity. Generally speaking, if $GR^2 \geq 0.90$, the sequences are very similar to each other; if $0.80 \leq GR^2 \leq 9.0$, the sequences have high degree of linearity, while $GR^2 < 0.80$ means the linearity is low.

We also downloaded all the NASDAQ stock prices from the Yahoo website starting from 05-08-2001 and ending at 05-08-2003. It has over 4000 stocks. We used the daily closing prices of each stock as sequences. These sequences are not of the same length for various reasons, such as cash dividend, stock split, etc. We made each sequence length 365 by truncating long sequences and removing short sequences. Finally, we have 3140 sequences all with the same length 365.

The objective is to mine clusters of stocks with similar trends. We have two choices. The first is GRM.2 and the second is to use the Simple Regression Model and calculate linearity of each pair by brute-force. For convenience, we name this method BFR (Brute-Force R^2). As we discussed in Chapter 2, the Simple Regression Model can also compare two sequences without be affected by scaling and shifting, because R^2 is invariant to both scaling and shifting. Intuitively, BFR is more accurate but slower, while GRM.2 is faster at the expense of accuracy. Our experiments compare the accuracy and performance of BFR and GRM.2. We do not have to compare GRM.2 with other methods in the field of sequence analysis because other methods do not match multiple sequences at a time and most of them are sensitive to scaling and shifting. We require the confidence of linearity of sequences in the same cluster to be greater than 85%. Recall that linearity is defined by GR^2 in GRM and R^2 in the Simple Regression Model.

Given a set of sequences $S = \{Xi \mid i = 1, 2, \cdots, K\}$, BFR works as follows:

1) Take an arbitrary sequence from S, say X_i. Find all sequences from S which have $R^2 \geq$ 85% with X_i. Collect them in a temporary set. Do *post-selection* to ensure all sequences have confidence of linearity $\geq$ 85% with each other by deleting some sequences from the set if necessary.

2) Save this temporary set as a cluster and delete all sequences in this set from S.

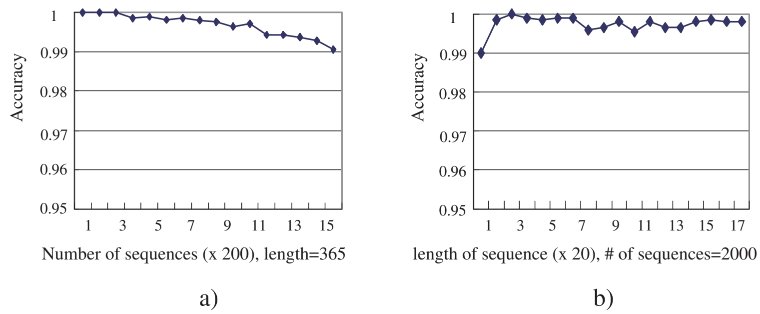


Figure 6.5: a) Relative accuracy of GRM.2 when the number of sequences varies. b) Relative accuracy of GRM.2 when the length of sequences varies.

3) Repeat 1) until S become empty.

After we find clusters $CM_1, CM_2, \cdots, CM_l$ using GRM.2 and clusters $CR_1, CR_2, \cdots, CR_h$ using BFR, we can calculate the accuracy of GRM.2. If the accuracy of the BFR is considered as 1, then the relative accuracy of GRM.2 is as follows.

$$Accuracy(CM, CR) = [\sum_i max_j(Sim(CM_i, CR_j))]/l,$$

where $Sim(CM_i, CR_j) = 2\frac{|CM_i \cap CR_j|}{|CM_i| + |CR_j|}$.

Our programs are written using MATLAB 6.0. Experiments were performed on a PC with 800MHz CPU and 384MB of RAM.

Fig. 6.5 a) shows the accuracy of GRM.2 relative to BFR while the number of sequences varies when the length of sequence is fixed at 365. The accuracy remains above 0.99 (99%) when the number of sequences increases over 3000. Fig. 6.5 b) shows that the relative accuracy of GRM.2 stays above 0.99 when the length of sequences varies with the number of sequences fixed at 2000.

Fig. 6.6 a) and b) show the running time of GRM.2 and BFR. We can see that GRM.2

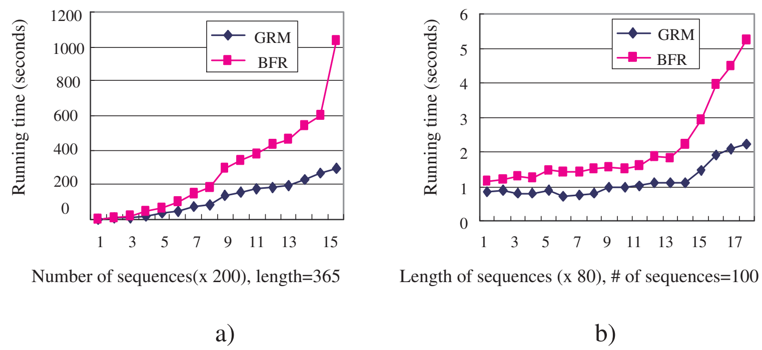


Figure 6.6: Performance comparison between GRM.2 and BFR. a) Running time comparison when the number of sequences varies. b) Running time comparison when the length of sequences varies.

is significantly faster than BFR, no matter how the number of sequences vary while their length is fixed and vice versa. The greater speed is at the cost of less than 1% accuracy of clustering. Note that any clustering algorithm cannot be 100% correct because the classification of some points is always ambiguous.

Fig. 6.7 shows a cluster of stocks we found using GRM.2. The stocks of four companies, A (Agile Software Corporation), M (Microsoft Corporation), P (Parametric Technology Corporation) and T (TMP Worldwide Inc.), have similar trends. They have the following approximate linear relation: $\frac{A-10.96}{0.0138} = \frac{M-29.36}{0.0140} = \frac{P-6.37}{0.0115} = \frac{T-33.62}{0.0563}$

From the stocks shown in Fig. 6.7, we can see that A, M and P fit each other very well with some shifting. T fits others with both scaling and shifting. This is consistent with this our observation, since 0.0138, 0.014, 0.0115 are values close to each other, which means the scaling factor between each pair of M, P and T is close to 1, while 0.0563 is about 4 times of 0.014, therefore the scaling factor between T and M is about 4. Fig. 6.8 shows the

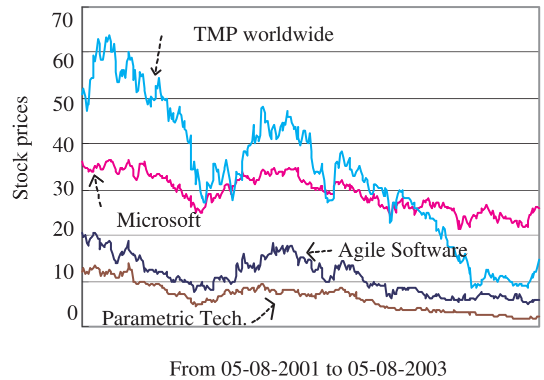


Figure 6.7: A cluster of stocks mined out by GRM.2.

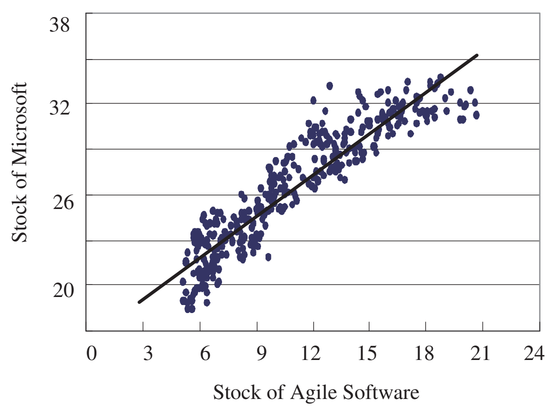


Figure 6.8: The regression of Microsoft stock on Agile Software stock.

regression of Microsoft stock on Agile Software stock. The distribution of the points in the 2-dimensional space confirms that the two stocks have a strong linear relation.

6.4 Chapter Summary

GRM is a generalization of the traditional Simple Regression Model. GRM gives a measure GR^2, which is a new measure of linearity among multiple sequences. The meaning of

GR^2 for linearity is not relative. Based on GR^2, algorithm GRM.1 can test the linearity of multiple sequences at the same time and GRM.2 can cluster massive sequences with high accuracy and high efficiency.

6.5 Appendix

Determine the Regression Line in *K*-dimensional space

Suppose $p_1, p_2, \cdots, p_N$ are the N points in K-dimensional space. First we can assume the regression line is l, which can be expressed as:

$$l = m + ae, \tag{6.12}$$

where m is a point in the K-dimensiona space, a is an arbitrary scalar, and e is a unit vector in the direction of l.

When we project points $p_1, p_2, \cdots, p_N$ to line l, we have point $m + a_i e$ corresponding to $p_i (i = 1, \cdots, N)$. The squared error for point p_i is:

$$u_i^2 = \parallel (m + a_i e) - p_i \parallel^2 \tag{6.13}$$

Thus, the sum of all the squared-errors is:

$$\begin{aligned} \sum_{i=1}^{N} u_i^2 &= \sum_{i=1}^{N} \parallel (m + a_i e) - p_i \parallel^2 \\ &= \sum_{i=1}^{N} \parallel a_i e - (p_i - m) \parallel^2 \\ &= \sum_{i=1}^{N} a_i^2 \parallel e \parallel^2 - 2 \sum_{i=1}^{N} a_i e^t (p_i - m) + \sum_{i=1}^{N} \parallel p_i - m \parallel^2 \\ &= \sum_{i=1}^{N} a_i^2 - 2 \sum_{i=1}^{N} a_i e^t (p_i - m) + \sum_{i=1}^{N} \parallel p_i - m \parallel^2 \end{aligned}$$

Since e is unit vector, $\| e \|^2 = 1$.

In GRM, the sum of squared error must be minimized. Note that $\sum_{i=1}^{N} u_i^2$ is a function of m, a_i and e. Partially differentiating it with respect to a_i and setting the derivative to be zero, we can obtain:

$$a_i = e^t(p_i - m) \tag{6.14}$$

Now, we should determine vector e to minimize $\sum_{i=1}^{N} u_i^2$. Substituting (6.14), we have:

$$\begin{aligned}
\sum_{i=1}^{N} u_i^2 &= \sum_{i=1}^{N} a_i^2 - 2\sum_{i=1}^{N} a_i a_i + \sum_{i=1}^{N} \| p_i - m \|^2 \\
&= -\sum_{i=1}^{N} a_i^2 + \sum_{i=1}^{N} \| p_i - m \|^2 \\
&= -\sum_{i=1}^{N} [e^t(p_i - m)]^2 + \sum_{i=1}^{N} \| p_i - m \|^2 \\
&= -\sum_{i=1}^{N} [e^t(p_i - m)(p_i - m)^t e] + \sum_{i=1}^{N} \| p_i - m \|^2 \\
&= -e^t S e + \sum_{i=1}^{N} \| p_i - m \|^2,
\end{aligned}$$

where $S = \sum_{i=1}^{N}(p_i - m)(p_i - m)^t$, called *scatter matrix*[22].

The vector e that minimizes the above equation also maximizes $e^t S e$. We can use Lagrange multipliers to maximize $e^t S e$ subject to the constraint $\| e \|^2 = 1$. Let:

$$\mu = e^t S e - \lambda(e^t e - 1) \tag{6.15}$$

Differentiating μ with respect to e, we have:

$$\frac{\partial \mu}{\partial e} = 2Se - 2\lambda e \tag{6.16}$$

Therefore, to maximize $e^t Se$, e must be the eigenvector of the scatter matrix S:

$$Se = \lambda e \tag{6.17}$$

Furthermore, note that,

$$e^t Se = \lambda e^t e = \lambda \tag{6.18}$$

Since S generally has more than one eigenvector, we should select the eigenvector e which corresponds to the largest eigenvalue λ.

Finally, we need m to complete the solution. $\sum_{i=1}^{N} \parallel p_i - m \parallel^2$ should be minimized (since it is always non-negative) by making m the average of $p_1, p_2, \cdots, p_N$. Readers are referred to [22] for proof.

With m as the average of the N points and e derived from (6.17), the regression line l is determined. Let $e = [e_1, e_2, \cdots, e_K]^t$ and $m = [\overline{x_1}, \overline{x_2}, \cdots, \overline{x_K}]^t$. l can be expressed as: $\frac{p(1)-\overline{x_1}}{e_1} = \frac{p(2)-\overline{x_2}}{e_2} = \cdots = \frac{p(K)-\overline{x_K}}{e_K}$, where $p(i)$ is the i-th element of point p.

Chapter 7

SUMMARY

This thesis focuses on the problem sequential pattern classification without explicit feature extraction. To avoid explicit and costly feature extraction, we present direct similarity measure ER^2 for use in multi-dimensional sequence matching. It is tested in the applications of on-line signature verification and on-line handwritten character recognition.

In the SVM framework, we present implicit feature extraction in off-line handwritten digit recognition, where the image is taken as sequences by simply ordering the pixels. For unsupervised classification, we have proposed Generalized Regression Model to tackle the clustering problem.

In chapter 2, we presented a novel similarity measure ER^2 by extending R^2 from the Simple Regression Model. In Chapter 3, we show the application of ER^2 in on-line signature verification. Signature verification systems can use ER^2 coupled with optimal alignment to obtain an intuitive similarity score and high accuracy. Although the experimental results are encouraging, further evaluation on large and real databases is necessary. When using ER^2, no feature extraction is necessary. Just the X-, Y-coordinate sequences in their original representations are used.

In Chapter 4, we propose ER^2-driven sequence classification. 2D shape recognition problems, such as on-line handwriting, can benefit from ER^2. ER^2 speeds up conventional SVM by making early classification decision when the input pattern is found to be very similar to the samples of a certain class. For sequence classification based on SVM, RBF kernel was found to be more suitable than DTW kernel, although DTW is usually more robust than the Euclidean norm. The simulation results on adaptive on-line handwritten character recognition are encouraging. The writer-dependent character recognition accuracy was above 94% without explicit feature extraction.

In Chapter 5, off-line handwritten digit recognition is implemented by incorporating both PCA and RFE in the standard SVM. The accuracy of featureless recognition was 98.14% (see Table 5.8). With FR-SVM, no feature is explicitly defined. PCA reduces the dimensions and RFE selects the most discriminative features (the ”features” are original data with simple preprocessing). By choosing the proper number of principle components and the number of top ranked features for each pair of classes, a significant speedup in SVM evaluation can be achieved while maintaining accuracy. The speedup enables the featureless handwritten digit recognition from a practical point of view.

In Chapter 6, we presented GRM by generalizing the traditional Simple Regression Model. GRM gives the measure GR^2, which is a new measure for linearity of multiple sequences. GR^2 based algorithm can test the linearity of multiple sequences all at once and cluster massive sequences with high accuracy and efficiency.

We implemented supervised and unsupervised sequential pattern classification when explicit feature extraction is absent for a variety of applications, ranging from on-line signature verification to similar trends discovery, from on-line handwritten character recognition to off-line handwritten digit recognition. Explicit feature extraction for sequential patterns can be avoided when: i) a direct similarity measure such as ER^2 is available or ii) efficient learning classifier such as FR-SVM (Feature Reduced Support Vector Machine) is present.

7.1 Future Work

Future work includes conducting more experiments on on-line handwritten data, mainly the benchmark database UNIPEN. Since explicit feature extraction is avoided, the recognition can be language independent. We can test the recognition method on different languages, such as English, Chinese and Hindi. In the future, it might be also valuable to systematically compare the handwriting recognition with and without explicit feature extraction. Chapter 5 serves as an exploration in the feasibility of off-line handwriting digit recognition without explicit feature extraction.

In sequential pattern classification, explicit feature extraction has been shown to be not always necessary. How about in regular pattern classification? Will the explicit feature extraction also be avoidable? Future work will continue to explore this direction.

BIBLIOGRAPHY

[1] The first international signature verification competition. *http://www.cs.ust.hk/svc2004/*, 2004.

[2] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence database. *Th 4th International Conference on Foundations of Data Organizations and Algorithms*, pages 69–84, 1993.

[3] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence database. *The 4th International Conference on Foundations of Data Organizations and Algorithms (FODO)*, pages 69–84, 1993.

[4] R. Agrawal, K. Lin, H. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time series databases. *The 21st International Conference on Very Large Databases(VLDB)*, pages 490–501, 1995.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. *The 11th International Conference on Data Engineering*, pages 3–14, 1996.

[6] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with

support vector machines - a kernel approach. *The 8th International Workshop on Frontiers in Handwriting (IWFHR)*, pages 49–54, 2002.

[7] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. *Working Notes of the Knowledge Discovery in Databases Workshop*, pages 359–370, 1994.

[8] C. Blake and C. Merz. UCI repository of machine learning databases. 1998.

[9] B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, 1992.

[10] C. Burges and B. Schölkopf. Improving speed and accuracy of support vector learning machines. In *Advances in Kernel Methods - Support Vector Learning*, pages 375–381, Cambridge, MA, 1997. MIT Press.

[11] K. Chan and W. Fu. Efficient sequences matching by wavelets. *The 15th international Conference on Data Engineering*, 1999.

[12] C. Chang and C. Lin. LIBSVM: a library for support vector machines. *http://www.kernel-machines.org/*, 2001.

[13] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

[14] K. Chu and M. Wong. Fast time series searching with scaling and shifting. *The 18th ACM Symposium on Principles of Database Systems(PODS)*, pages 237–248, 1999.

[15] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

[16] G. Das and D. Gunopulos. Time series measures. In *The tutorial notes of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 243–307, 2000.

[17] G. Das, K. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *Knowledge Discovery and Data Mining*, pages 16–22, 1998.

[18] S. Davis and S. Russel. NP-completeness of searches for smallest possible feature sets. *The AAAI FAll Symposion on Relevance*, pages 37–39, 1994.

[19] I. Dhillon. A New $O(n^2)$ Algorithm for the Symetric Tridiagonal igenvalue/Eigenvector Problem. PhD thesis, University of California, Berkerley, 1997.

[20] T. Downs, K. Gates, and A. Masters. Exact simplification of support vector solutions. *Journal of Machine Learning Research*, 2:293–297, 2001.

[21] K. Duan, S. Keerthi, and A. Poo. Evaluation of simple performace measures for tuning hyperparameters. *Neurocomputing*, 15:487–507, 2003.

[22] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2000.

[23] R. Duin, D. Ridder, and D. Tax. Experiments with a featureless approach to pattern recognition. *Pattern Recognition Letters*, 18:1159–1166, 1997.

[24] R. Duin, D. Ridder, and D. Tax. Featureless pattern classification. *Kybernetika*, 34(4):399–404, 1998.

[25] C. Faloutsos, M.Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *ACM SIGMOD Conference*, pages 419–429, 1994.

[26] J. Favata and G. Srikantan. A multiple feature/resolution approach to handprinted digit and character recognition. *International Journal of Imaging Systems and Technology*, 7:304–311, 1996.

[27] D. Goldin and P. Kanellakis. On similarity queries for time series data: Constraint specification and implementation. *The 1st International Conference on the Principles and Practice of Constraint Programming*, pages 137–153, 1995.

[28] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[29] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project

of on-line data exchange and recognizer benchmarks. In *The 12th International Conference on Pattern Recognition*, pages 29–33, 1994.

[30] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002.

[31] S. Hettich and S. Bay. The UCI KDD archive. *http://kdd.ics.uci.edu*, 1999.

[32] C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13:415–425, 2002.

[33] A. Jain, F. Griess, and S. Connell. On-line signature verification. *Pattern Recognition*, 35(12):2963–2972, 2002.

[34] E. Keogh. Exact indexing of dynamic time warping. *The 28th International Conference on Very Large Data Bases(VLDB)*, pages 406–417, 2002.

[35] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 102–111, 2002.

[36] E. Keogh and M. Pazzani. Scaling up dynamic time warping for datamining applications. *The 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 1–11, 1999.

[37] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *The 28th International Conference on Very Large Data Bases(VLDB)*, pages 406–417, 2002.

[38] E. Keogh and P. Smyth. A probabilistic approach to fast pattern matching in time series databases. In *The 3rd International Conference on Knowledge Discovery and Data Mining*, pages 24–30, 1997.

[39] S. Kim and S. Park. An index-based approach for similarity search supporting time warping in large sequence databases. *The 17th International Conference on Data Engineering(ICDE)*, pages 607–614, 2001.

[40] S. Kim, S. Park, and W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. *The 16th International Conference on Data Engineering(ICDE)*, pages 607–614, 2001.

[41] U. Kreßel. Pairwise classification and support vector machines. In *Advances in Kernel Methods - Support Vector Learning*, pages 255–268, Cambridge, MA, 1999. MIT Press.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, Nov 1998.

[43] L. Lee, T. Berger, and E. Aviczer. Reliable on-line human signature verification sys-

tems. *IEEE Transations on Pattern Analysis and Machine Intelligence*, pages 643–647, 1996.

[44] M. S. Lee, S. S Keerthi, C. J. Ong, and D. Decoste. An efficient method for computing leave-one-out error in support vector machines with gaussian kernels. *IEEE Transactions on Neural Networks*, 15:750–757, 2004.

[45] Y. Lee and O. Mangasarian. RSVM: Reduced support vector machines. In *The First SIAM International Conference on Data Mining*, 2001.

[46] H. Lei and V. Govindaraju. Er-squared: an intuitive similarity measure for on-line signature verification. In *The 9th International Workshop on Frontiers in Handwriting Recognition*, pages 191–195, 2004.

[47] H. Lei and V. Govindaraju. GRM: Generalized regression model for clustering linear sequences. *The 4th SIAM Conference on Data Mining*, 2004.

[48] H. Lei and V. Govindaraju. A study on the consistency of features for on-line signature verification. *Joint IAPR International Workshops on Syntactical and Structural Pattern Recognition (SSPR 2004) and Statistical Pattern Recognition (SPR2004)*, 2004.

[49] H. Lei and V. Govindaraju. Half-against-half multi-class support vector machines. *The 6th International Workshop on Multiple Classifier Systems*, pages 156–164, 2005.

[50] K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14:1449–1559, 2003.

[51] J. Ma, Y. Zhao, and S. Ahalt. OSU SVM classifier matlab toolbox. *http://www.kernel-machines.org/*, 2002.

[52] R. Martens and L. Claesen. On-line signature verification by dynamic time warping. *The 13th International Conference on Pattern Recognition*, pages 38–42, 1996.

[53] F. Mosteller and J. Tukey. *Data Analysis and Regression: A Second Course in Statistics*. Addison-Wesley, 1977.

[54] V. Mottl, S. Dvoenko, O. Seredin, and C. Muchnik. Featureless pattern recognition in an imaginary Hilbert space and its application to protein fold classification. *Machine Learning and Data Mining in Pattern Recognition : Second International Workshop*, 2001.

[55] M. Munich and P. Perona. Visual identification by signature tracking. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 2003.

[56] V. Nalwa. Automatic on-line signature verification. *Proceedings of the IEEE*, 85(2):213–239, 1997.

[57] T. Oates, L. Firoiu, and P. Cohen. Clustering time series with hidden markov models

and dynamic time warping. In *The IJCAI-99 Workshop on Neural, Symbolic and Reinforcement Learning Methods for Sequence Learning*, pages 17–21, 1999.

[58] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In *IEEE Workshop on Neural Networks for Signal Processing*, 1997.

[59] R. Plamondon and S. Srihari. On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[60] J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*, pages 185–208, Cambridge, MA, 1999. MIT Press.

[61] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553, 2000.

[62] D. Rafiei and A. Mendelzon. Efficient retrieval of similar time sequences using DFT. *The 5th International Conference on Foundations of Data Organizations and Algorithms (FODO)*, pages 69–84, 1998.

[63] T. Rhee, S. Cho, and J. Kim. On-line signature verification using model-guided seg-

mentation and discriminative feature selection for skilled forgeries. *The 6th International Conference on Document Analysis and Recognition*, 2001.

[64] R. Rifin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[65] G. Seni, R. Srihari, and N. Nasrabadi. Large vocabulary recognition of on-line handwritten cursive words. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):757–762, 1996.

[66] H. Shimodaira, K. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 921–928, Cambridge, MA, 2002. MIT Press.

[67] P. Smyth. Clustering sequences with hidden markov models. In *Advances in Neural Information Processing Systems*, volume 9, page 648, 1997.

[68] S. Srihari. High performance reading machines. *Proceedings of IEEE*, 80(7):1120–1132, 1992.

[69] Z. Struzik and A. Siebes. The haar wavelet transform in the sequences similarity paradigm. *The 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, 1999.

[70] C. Tappert, C. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, 1990.

[71] T.Bozkaya, N. Yazdani, and Z. Ozsoyoglu. Matching and indexing sequences of different lengths. In *The 6th Conference on Information and Knowledge Management(CIKM)*, pages 128–135, 1997.

[72] V. Vapnik. *Estimation of Dependencies Based on Empirical Data*. Springer-Verlag, New York, 1982.

[73] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[74] V. Vapnik and O. Chapelle. Bounds on error expectation for support vector machines. *Neural Computation*, 12:2013–2036, 2000.

[75] V. Vuori. *Adaptive methods for on-line recognition of isolated handwritten characters*. PhD thesis, Helsinki University of Technology, 2002.

[76] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for SVMs. *Advances in Neural Information Processing Systems*, 2001.

[77] J. Wooldridge. *Introductory Econometrics: a modern approach*. South-Western College Publishing, second edition, 1999.

[78] B. Yi and C. Faloustsos. Fast time sequence indexing for arbitrary Lp norms. *The 26th International Conference on Very Large Databases(VLDB)*, pages 385–394, 2000.

[79] B. Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. *The 14th International Conference on Data Engineering(ICDE)*, pages 23–27, 1998.

[80] H. Yu. *Data Mining via Support Vector Machines: Scalability, Applicability, and Interpretability*. PhD thesis, University of Illinois at Urbana-Champaign, May 2004.