Word Segmentation of Handwritten Text Using Supervised Classification Techniques

Yi Sun [a,*], Timothy S. Butler [a], Alex Shafarenko [a], Rod Adams [a], Martin Loomes [a], Neil Davey [a]

[a] *Department of Computer Science*
Faculty of Engineering and Information Sciences
University of Hertfordshire
College Lane, Hatfield
Hertfordshire AL10 9AB

Abstract

Recent work on extracting features of gaps in handwritten text allows a classification of these gaps into *inter-word* and *intra-word* classes using suitable classification techniques. In this paper, we first analyse the features of the gaps using mutual information. We then investigate the underlying data distribution by using visualisation methods. These suggest that a complicated structure exists, which makes them difficult to be separated into two distinct classes. We apply 5 different supervised classification algorithms from the machine learning field on both the original dataset and a dataset with the best features selected using mutual information. Moreover, we improve the classification result with the aid of a set of feature variables of strokes preceding and following each gap. The classifiers are compared by employing McNemar's test. We find that SVMs and MLPs outperform the other classifiers and that preprocessing to select features works well. The best classification result attained suggests that the technique we employ is particularly suitable for digital ink manipulation at the level of words.

Key words: handwriting, supervised classification, mutual information, McNemar's test

* Corresponding author.
Email address: `Y.2.Sun@herts.ac.uk` (Yi Sun).

1 Introduction

In this paper, we address the problem of identifying word boundaries in handwritten text: a process known as word segmentation. We make use of a selection of contemporary classification algorithms, such as multi-layer perceptrons, support vector machines, and Gaussian mixture models.

Surprisingly, little attention has been paid to the word segmentation problem by the neural net work community. Nevertheless, recent work on extracting features of gaps between pieces of handwritten text, allows for the segmentation of words by classifying gaps into *inter-word* and *intra-word* classes directly [5]. In this paper we try to find a suitable classifier to automatically segment so-called *digital ink*: graphically enhanced fragments of pen trace representing handwritten words, shapes and symbols, of the sort that usually appear on paper when real ink is used for writing. Further details about the problem domain can be found in the next section. The previous work was done by using statistical methods to classify gaps into two classes based on one significant feature, named *river*, which is described in more detail in the following section. Each stroke involves an array of time-stamped sample points. However, as indicated in [14], exceptions are commonplace because of flourishes in writing styles with leading and trailing ligatures in handwriting. It is important to consider other possible features, as combinations of variables can provide significant information, which is not available in any of the individual variables separately. The task is therefore to propose an automatic classifier which can make as few errors as possible, based solely on the set of features.

In this work, we first attempt to understand the underlying data distribution by means of visualisation techniques. We then test 5 different supervised classification learning algorithms from the machine learning field to categorise gaps. We are also interested in selecting the most significant features. Since there is a proportion of gaps which can be classified with 99 percent accuracy in terms of the value of *river* directly, we also apply these classification techniques for those patterns which cannot be judged easily by the feature *river*.

We expound the problem domain in the next section. In Section 3, we introduce the datasets used in this paper. We explain how we select a subset of features in terms of mutual information. In addition, *hard* datasets are obtained by removing those gaps that are easily classified by the *river* feature. Principal component analysis [2] and the Generative Topographic Mapping [3] are employed to visualise the datasets in Section 4. Section 5 briefly introduces the classifiers used in our experiments. All experimental results are given in Section 6. We also analyse the classification results by applying McNemar's test. The paper ends in Section 7 with a discussion.

2 Problem Domain

Despite the widespread use of office computers, handwriting has been and remains an important mode of capturing and annotating textual information. Computer-assisted handwriting is an increasingly important part of the general interface between the electronic media and the business world. Indeed, apart from the niche market of Personal Digital Assistants (PDA, including mainly smart phones and palmtop PCs), where the use of pen input devices is motivated primarily by their greater compactness, the mainstream computing technology now includes so called Tablet PCs. A tablet PC is a portable computer with a sensitive screen and a digital stylus, which is used as the main, or even the only, input device. The operating system of a tablet PC is augmented with components that can handle *digital ink*. It is important to understand the difference between the digital ink and character-recognition interfaces. While the latter is merely a form of machine intelligence capable of recognising letters of an alphabet so that a keyboard can be replaced by an equivalent, but more compact, tablet and pen, the digital ink represents a separate form of input. It persists in documents as long as desirable for the author or/and readers. More importantly though, it is *processed* in its native form, i.e. as a graphical object. Words may be inserted, deleted or replaced at will without first being converted into a semantically focused form, such as an ASCII string. Such a conversion may happen eventually, when the final copy is produced.

There is therefore a fine balance for digital ink applications, namely one between the graphical form and semantic substance. One would like to benefit from the immediacy of pen input, its highly informal nature and potentially unlimited alphabet of letters, features and symbols, while at the same time having the computer penetrate the *structure* of the ink to the extent that it is necessary to be able to edit distinct parts of it. The depth of such penetration needs to be no more than superficial, down to a level of large self-contained units, such as lines and words, where the structuring is fairly well (albeit informally) defined. On the other hand, if no analysis is done of the ink input, then it is not really treated as handwriting, but as a general freehand graphical input. Consequently computer assistance (in the form of automatic placement, formatting and linkage with the rest of the document environment) would be very limited.

In this paper we focus on one level of the semantic penetration of pen input: the level of words. By 'word' we mean a group of pen strokes that have lexical significance, i.e. one that represents a word in a human language or a distinct symbol that can be used as a word. We wish to automatically segment digital ink represented as a *sampled pen trace* into word fragments purely on the basis of spatiotemporal relations between consecutive strokes, ignoring any meaning that may be represented by each such stroke. This has been a known problem in handwriting recognition research as well, although in this area of technology, word segmentation is seen merely as a precursor to full character recognition. In their recent comprehensive survey of handwriting recognition research, Plamondon and Srihari state that "prior to any recognition the acquired data is generally preprocessed to... segment the signal into meaningful units" [14].

The history of word segmentation research is delimited by the survey [14] and the one 10 years earlier [18], which is also referenced in [14]. The significant achievements reported in [18] for this area are confined to straightforward geometric segmentation using convex shells [10] with some consideration given to stroke timing. It is noteworthy that these early proposals have not been developed any further as is evidenced by [14]. One can only speculate about the reason why no further progress has been reported. Our experience shows that simple segmentation methods are prone to error due to an individual writer's idiosyncrasies as well as the fact that these methods fail to capture more subtle structural and temporal signals which would strengthen the basis for segmentation. More recent work is attempting to improve structure recognition by introducing hierarchical agglomerative clustering, see [16,11] in a broader context of automatic structural analysis of handwritten document. These in our opinion are interesting approaches, though they are susceptible to writing idiosyncrasies while being insensitive to any recurrent features of the language (or symbolic system) used by the writer.

The variability of one's writing style as well as the inherent diversity of writers would strongly advocate an adaptive solution. The solution would not be confined to any specific *ad hoc* metric of the pen trace as the basis of segmentation, but would accommodate a reasonably large set of these metrics, taking into account both prime features (such as the size and duration of inter-stroke gaps) as well as any secondary ones which may be significant. Such features are still proposed on the basis of their plausibility, without much formal basis or a priori evidence. However, we have been guided by [15] where a thorough geometric and temporal classification was provided for a pen gesture recogniser. To give an idea of the sort of features that were being used there, we illustrate some of them in Fig. 1. It presents a single pen stroke with its bounding box. The features x and y as shown give the dimensions of the bounding box and the angle α is linked with its aspect ratio. The distance s is between the end points of the stroke, and β is the angle between the line connecting those points and the vertical. Finally, if θ_i is the angle between two consecutive pen segments of the stroke, i and $i+1$, then one can use the feature

$$\sigma = \sum_{i=1}^{n-1} \theta_i$$

as a measure of curvature. The proposed features were not all purely geometric; there were a few related to the time interval of the stroke and the speed of the pen tip. Note that most of these features are inapplicable to inter-stroke gaps, but some still make sense, e.g., x, y β, etc. We have introduced a gap feature which has proven especially useful for our purposes. We call it *river width* or *river* for short, following Fox and Tappert [10]. The river of a gap is the shortest distance between two consecutive strokes, i.e. the length of the shortest chord drawn between pen position samples from neighbouring strokes, as shown in Fig. 2. Two rivers are indicated there by double-headed arrows.

We have expanded the set proposed in [10] by our own form factors, see [5], for each pen stroke. The pen trace has thus been abstracted to a sequence of strokes and gaps, where each gap is represented by 14 feature variables, while each stroke by 25. In this work, we are interested in classifying gaps. A human reader has annotated the gaps in our experimental traces as either intra-word or inter-word by recognising the words in the language. Thus the task is to

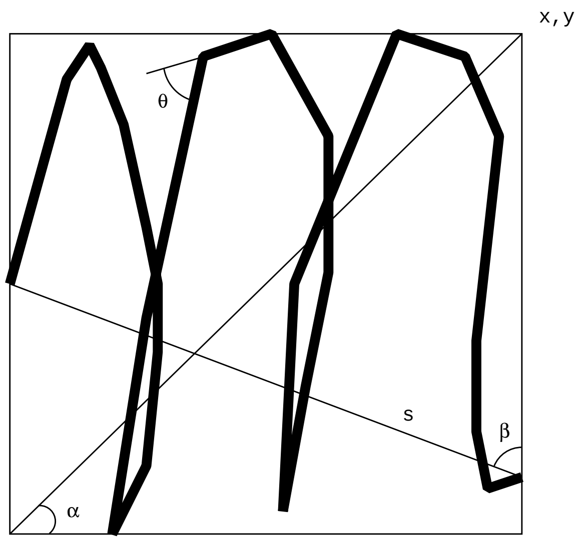

Fig. 1. An illustration: the sort of features of a single pen stroke with its bounding box.

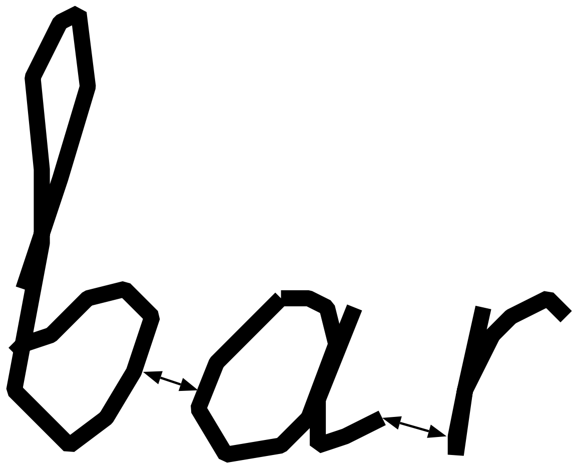

Fig. 2. An illustration: two rivers of gaps are shown by double-headed arrows.

search for a classification method which can produce the same annotations with as few errors as possible.

3 The Description of The Datasets

3.1 Gaps Datasets

The *original* gap dataset includes 2482 data points labeled as *inter-word* and 4980 as *intra-word*. In the experiments, 2/3 of the data points from the dataset are used for training, while 1/3 are used for testing. We do experiments with all 14 gap features and *reduced* features using the 8 most significant to the classification, found by analysing mutual information, as discussed in Section 3.2. The two datasets are referred to as *G-14* and *G-8*, respectively.

3.2 Feature Extraction by Using Mutual Information

The features associated with gaps are reduced by employing mutual information. The mutual information of two variables is a measure of the common information shared between them [12]. In this work, the two variables are the class variable c and the feature variable x, which could be any of the 14 features. The larger the value of the mutual information, the more common information is shared. If two variables are independent, their mutual information is zero. An advanced treatment of feature extraction using mutual information maximization can be found in [7].

In general, a data point may be assigned to one of C classes, which we shall denote by c_i, where $i = 1, \ldots, C$. (In this paper, $C = 2$). Mutual information, denoted by MI, is given by [6]

$$MI = H(c) - H(c|x), \tag{1}$$

where $H(c)$ is the entropy of the classes prior probability $P(c_i)$ given by

$$H(c) = -\sum_{i=1}^{C} P(c_i) \log P(c_i), \tag{2}$$

and $H(c|x)$ is conditional entropy having the form, as follows

$$H(c|x) = -\sum_{i=1}^{C} P(c_i, x) \log P(c_i|x), \tag{3}$$

where $P(c_i, x)$ are the joint probability distributions, and $P(c_i|x)$ are posterior probabilities. Equation (3) can be further written as

$$H(c|x) = -\sum_{i=1}^{C} P(c_i) \int p(x|c_i) \log P(c_i|x)\ dx. \tag{4}$$

Note that $\int p(x|c_i) \log P(c_i|x) dx$ is the expectation of $\log P(c_i|x)$ given the probability density $p(x|c_i)$.

Empirically the conditional entropy $H(c|x)$, which is based on the probability density function of the variable x, can be approximated as follows, when considering the two classes relevant to this paper:

$$H(c|x) \approx -\frac{1}{N_1} P(c_1) \sum_{k=1}^{N_1} \log P(c_1|x^k) - \frac{1}{N_2} P(c_2) \sum_{l=1}^{N_2} \log P(c_2|x^l), \tag{5}$$

where x^k and x^l denote the feature values given that the data points are generated from two densities $p(x|c_1)$ and $p(x|c_2)$, respectively. N_1 and N_2 are number of samples from the two distributions, respectively.

To compute (5), a sufficient number of data points: N_1 plus N_2, are required, sampled from the two estimated distributions. We employ two Gaussian mixture models to model the distributions of the gaps data collected from the class *inter-word* denoted by c_1 and *intra-word* denoted by c_2. The *expectation-maximisation* (EM) algorithm [8] is used for finding parameters of each model. A mixture distribution having M components (in this work, $M = 5$) can be calculated using:

$$p(x|c_i) = \sum_{j=1}^{M} p(x|j, c_i)P(j), \tag{6}$$

where $P(j)$ are mixing coefficients and satisfy the properties

$$\sum_{j=1}^{M} P(j) = 1, \qquad 0 \leq P(j) \leq 1, \tag{7}$$

which guarantee that $p(x|c_i)$ is a valid density function.and

$$p(x|j, c_i) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left\{-\frac{1}{2\sigma_j^2}(x - \mu_j)^2\right\} , \tag{8}$$

where μ_j and σ_j are mean and variance of each component j respectively. More details about Gaussian mixture models can be found in [2]. Then $500,000$ data points were sampled from these two distributions. Finally, the posterior probability can be computed using Bayes' theorem

$$P(c_i|x) = \frac{P(c_i)p(x|c_i)}{\sum_{i=1}^{C} P(c_i)p(x|c_i)}. \tag{9}$$

Fig. 3 shows the mutual information of each feature with the class variable, sorted by their values. As shown, there is a reasonable jump from the ninth value to the eighth. By ignoring the weaker features, namely those indexed from 9 to 14, we obtain a dataset, called *G-8*, with just 8 features all with MI values greater than 0.3.

3.3 The Hard Dataset with Thresholds

As seen in Fig. 3, there is one feature which is the most significant to classification, named *river*. As described in Section 2 and illustrated in Fig. 2, it measures the shortest distance between samples in adjacent strokes, and gaps between words usually have a larger value than gaps within words. We will give more detail about other features in section 3.4. By selecting threshold values for the *river* feature it is possible to directly classify a substantial subset of the gap dataset to an accuracy of 99%, as shown in Fig. 4. In this figure, the *river* values increase from left to right. Threshold 1 specifies a *river* value, on the left of which one can

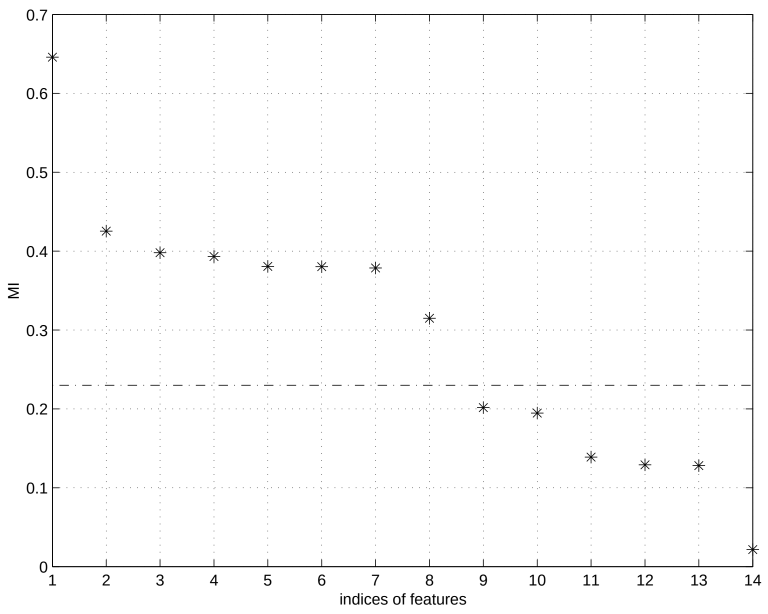

Fig. 3. Mutual information of class variable and each feature of gaps: each value is shown as a star sign. The background are dashed lines that are major grid lines to the current axes. The horizontal dash-dot line denotes the cut off value.

ensure that the probability that the gap belongs to class *intra-word* is not less than 99 percent; while threshold 2 specifies another value of *river*, on the right of which the probability that the gap belongs to class *inter-word* is not less than 99 percent. In this way, a sub-dataset called *hard*, whose values of the *river* feature are within these two boundaries, is obtained. This subset consists of 3361 gaps that cannot easily be classified by the *river* feature. On the other hand, a sub-dataset called *Evident-G-14*, which includes the gaps to the left of boundary 1 and to the right of boundary 2, is attained.

Both the *original* 14 features dataset and the reduced 8 features dataset can produce a *hard* subset. They are referred to as *Hard-G-14* and *Hard-G-8*, respectively. Our primary reason for using these datasets is that they are much smaller and can be separately classified much more quickly than the full datasets.

3.4 The Hard Stroke-gap-stroke Dataset

One might expect that a further improvement in classification could be achieved by utilising more information from the characteristics of the preceding and following strokes of a gap. To this end, a new set of vectors was created by concatenating a gap with its preceding stroke and its following stroke, i.e. stroke-gap-stroke. This dataset contains the same number of data points as the original gap dataset, namely 7462 points. However, this set now has a total of 65 features and is called *SGS-65*. As before, one can apply mutual information to select a subset of features for this dataset.

Looking at Fig. 5 and comparing with Fig. 3, it can be seen that the 10 most significant

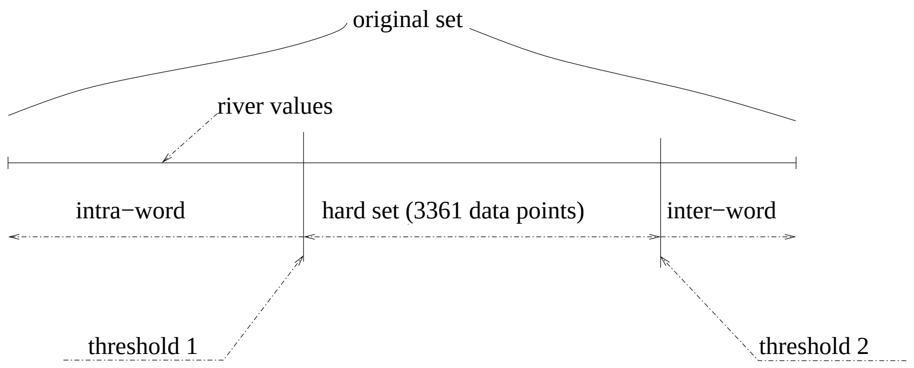

Fig. 4. A diagram: explaining how the *hard* dataset is generated with two thresholds.

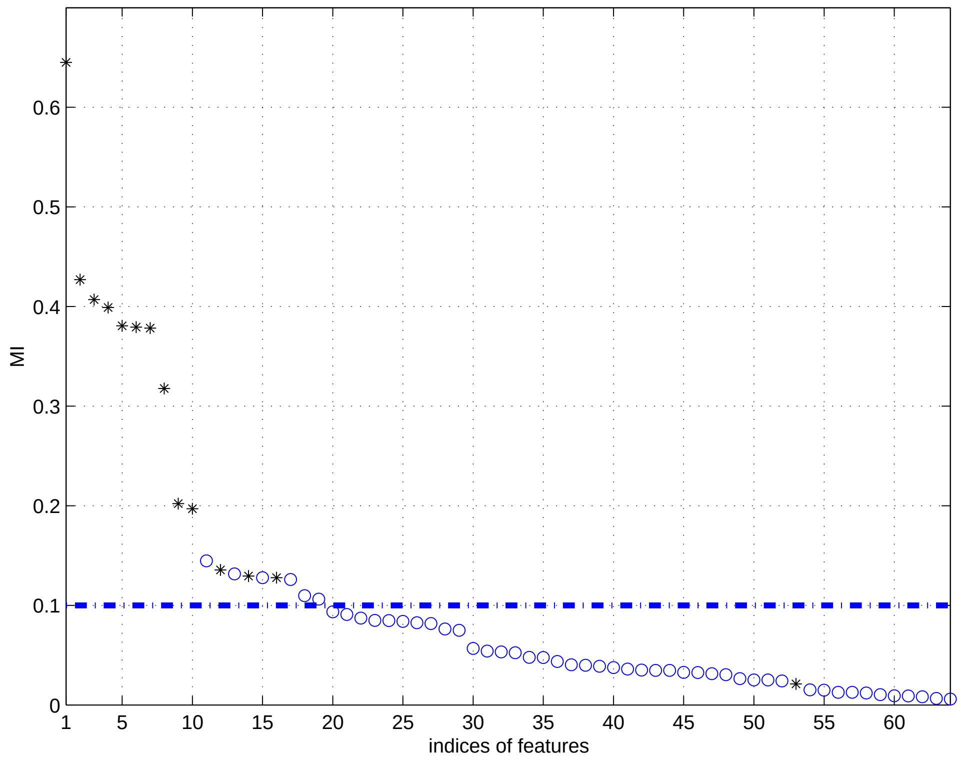

Fig. 5. Mutual information of the class variable and each feature: each value is shown as a star sign if the corresponding feature is one of gap variables, otherwise it is presented as a circle if the feature is one of stroke variables. The horizontal dash-dot line denotes the cut off value.

features for classification are in fact all gap feature variables. Since we want to know whether the combination of stroke and gap features can improve the categorisation, a lower cut off value of 0.1 is used so that a set of stroke features can be involved. This produces a reduced dataset, called *SGS-19*, with the same number of data points but only 19 features. Using the *river* feature, which is still the most significant feature in Fig. 5, one can again produce a *hard* subset from this reduced feature dataset, called *Hard-SGS-19*.

Table 1 lists the 19 selected features using mutual information. These include all but one gap feature and 6 new stroke features. Interestingly only 1 feature is selected by mutual information from the stroke preceding the gap. Note that some gap features, such as No. 5 and 6, are the same as those used for describing a stroke. During the data collection procedure, pen data is recorded while the pen is within 5mm of the table surface. While pen pressure is under a certain threshold, the pen is considered to off the tablet surface (i.e. a gap). However, there will be some pen-trace information describing the movement of the pen over a gap. At the very minimum this will include the start and end of the gap as the pen moves out of and in to proximity. At the most it will describe the movement of the pen over the entire gap.

3.5 A Dataset List

For the sake of clarity, all datasets mentioned in this paper are listed in Table 2, where the last column indicates whether the corresponding dataset is used in either the classification or visualisation section.

4 Visualisation

Before classifying gaps into *inter-word* and *intra-word* classes, we first look at the underlying data distribution by means of data visualisation technology, which projects data into a low-dimensional space, usually of two dimensions.

4.1 Visualisation Using PCA

We first visualise the Evident-G-14 and Hard-G-14 datasets using principal component analysis (PCA). Classical PCA [2] is a projection method which maps data to a low-dimensional space with a linear transformation. It is one of the most popular techniques for pre-processing and visualising data.

Fig. 6 and Fig. 7 are visualisation results using PCA. As can be seen, the *evident* gaps are separated into 2 distinct clusters, though there are a few points mixed together. On the other hand, the two different classes for *hard* gaps significantly overlap. Since PCA employs a simple linear transformation to map data into a 2 dimensional space, it is difficult to clearly reflect complex relationships between data points. It suggests that there are more complex non-linear structures in this dataset.

Table 1
19 significant features selected using mutual information.

No.	Feature	From	Description
1	**River**	gap	This is the minimum distance between two strokes. Distances are calculated between all the sample points in stroke N and those in stroke $N+1$.
2	**X-displacement**	gap	This is the **X**-displacement between the end of stroke N and the start of stroke $N+1$ (start and end of the gap).
3	**Distance between centre of gravity**	gap	The centre-of gravity of a stroke is the average all the sample points: $\mathbf{x}$ = avg. of all $\mathbf{x}$ coords, $\mathbf{y}$ = avg. of all $\mathbf{y}$ coords.
4	**Displacement**	gap	As **X**-displacement, but in 2 dimensions (**X** and **Y** coords).
5	**Length of the bounding box diagonal**	gap	A bounding box is found round the coords of the sample points for the path of the pen over the gap. Minimum and maximum **X** and **Y** coords are found by checking each sample point in the stroke. The length of the diagonal of this box is then found.
6	**Num. of samples in the gap**	gap	The number of sample points in the path of the pen over the gap.
7	**Duration of the gap**	gap	The time between the end of the preceding stroke and the beginning of the stroke..
8	**Distance between first and last point**	gap	The distance between **X**, **Y** of start point and **X**, **Y** of the end point.
9	**Cosine between first and last point**	gap	This is the cosine of the angle between the last **X**, **Y** sample point of stroke $N+1$ and first **X**, **Y** sample point of stroke N
10	**Distance between centre of gravity incl. pressure**	gap	This is as **Distance between centre of gravity**, but is in a 3-dimensional space, with pen pressure constituting the third dimension.
11	**Length of the bounding box diagonal**	following stroke	As **Length of the bounding box diagonal** above.
12	**Angle of the bounding box diagonal**	gap	This is the angle between minimum and maximum **X** and **Y** coords as found in **Length of the bounding box diagonal** above.
13	**Sine between first and last point**	following stroke	This is the sine of the angle between the first **X**, **Y** sample point of stroke N and last **X**, **Y** sample point of stroke $N+1$.
14	**Sine between first and last point**	gap	As (13) above, but for the gap.
15	**Num. of samples in stroke**	following stroke	As (6) above.
16	**Y-displacement**	gap	As (2) above, but in the **Y**-displacement.
17	**Duration of stroke**	following stroke	As (7) above.
18	**Total stroke length**	following stroke	The sum of the distances between adjacent sample points along a stroke.
19	**Total stroke length**	preceding stroke	As (18) above.

Table 2
Datasets in this paper.

Name	Number of features	Number of data points	Pattern	Used
G-14	14	7462	gaps	yes
G-8	8	7462	gaps	yes
Hard-G-14	14	3361	gaps	yes
Hard-G-8	8	3361	gaps	yes
Evident-G-14	14	4101	gaps	yes
SGS-65	65	7462	stroke-gap-stroke	no
SGS-19	19	7462	stroke-gap-stroke	no
Hard-SGS-19	19	3361	stroke-gap-stroke	yes

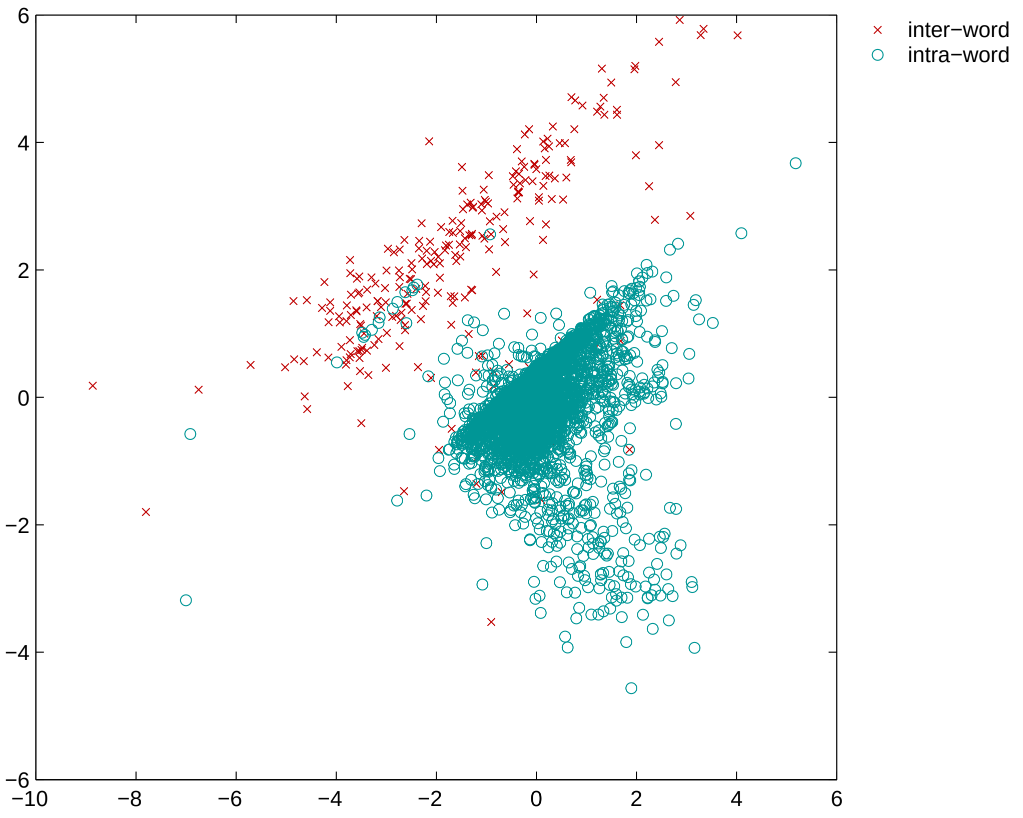

Fig. 6. Projection of Evident-G-14 using PCA.

4.2 The Generative Topographic Mapping

The generative topographic mapping (GTM) method has been proposed as a method for visualisation high-dimensional data [3]. The idea of the GTM is straightforward. It is assumed for visualisation purposes that each observed data point is the result of a non-linear mapping from of a grid in a low-dimensional (typically bounded in a 2-dimensional Euclidean domain, e.g. $[-1, 1] \times [-1, 1]$) *latent space*. It is also assumed that the observed data is noisy, and this

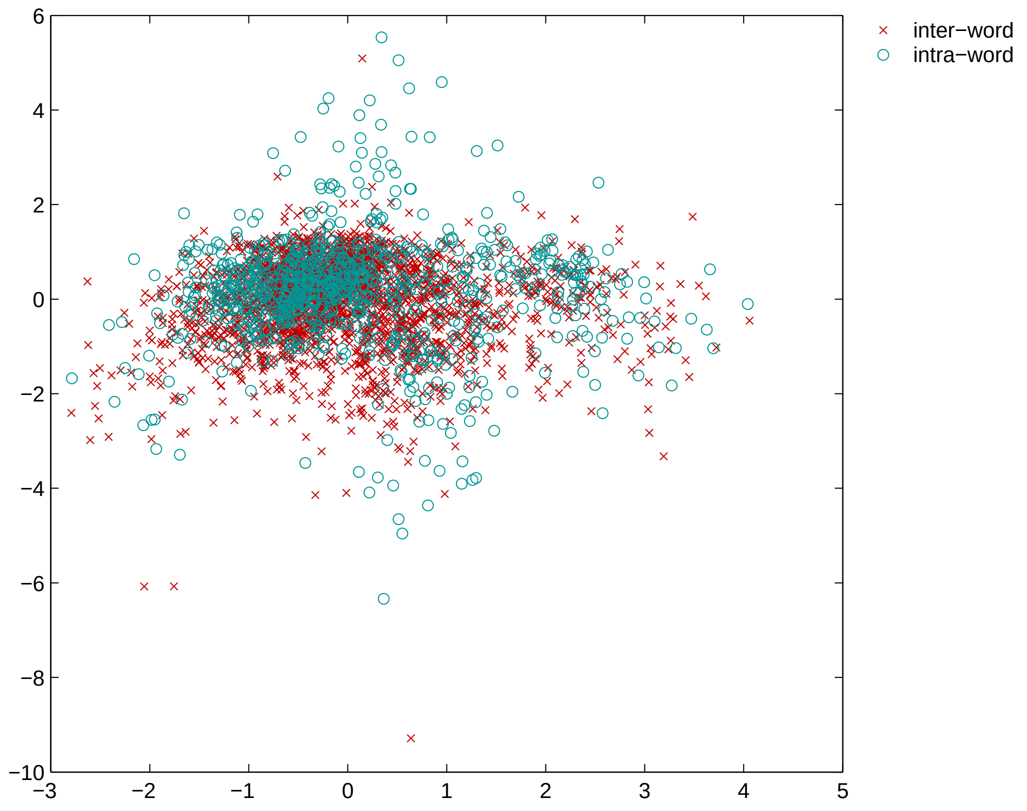

Fig. 7. Projection of Hard-G-14 using PCA.

is modelled by a spherical Gaussian probability density function centered on the transformed grid points (see Fig. 8). The EM algorithm is then used to estimate parameters of the GTM model.

At this point, the posterior probability that each of the grid nodes in the latent space is responsible for generating each of the data points is known. So the data points can be represented in the latent space as the mean position amongst the grid nodes weighted by these probabilities (see Fig. 9). Thus the aim of the GTM algorithm is to represent the high-dimensional data vectors $\{\mathbf{x}^n\}_{n=1,\ldots,N}$ in the latent space so important structural characteristics may be revealed.

Formally the GTM algorithm involves a non-linear mapping, using radial basis functions (RBF) from points $\mathbf{z}$ in an L-dimensional latent space $\mathcal{H}$ to the data space $\mathcal{D}$ with D-dimension $(L < D)$, is given by

$$\mathbf{y}(\mathbf{z};\mathbf{W}) = \mathbf{W}\boldsymbol{\phi}(\mathbf{z}), \tag{10}$$

where $\boldsymbol{\phi}(\mathbf{z})$ are M fixed radial basis functions $\boldsymbol{\phi}_m$ (including one bias term) and $\mathbf{W}$ is a $D \times M$ matrix of weight parameters.

The noise model $p(\mathbf{x}|\mathbf{z},\mathbf{W},\beta)$, which is the conditional density of the data given the latent variables, is a spherical Gaussian $\mathcal{N}(\mathbf{y}(\mathbf{z};\mathbf{W}),\beta^{-1}\mathbf{I})$ centred on $\mathbf{y}(\mathbf{z};\mathbf{W})$ with variance β^{-1}

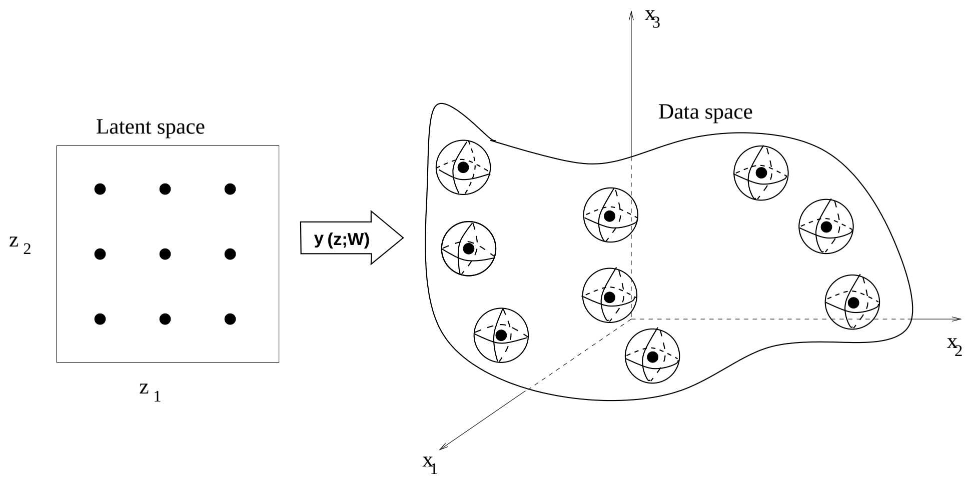

Fig. 8. GTM mapping and manifold: each node $\mathbf{z}_k$ located at a regular grid in the latent space is mapped to a corresponding point $\mathbf{y}(\mathbf{z}_k;\mathbf{W})$ in the data space, and forms the centre of a corresponding Gaussian distribution.

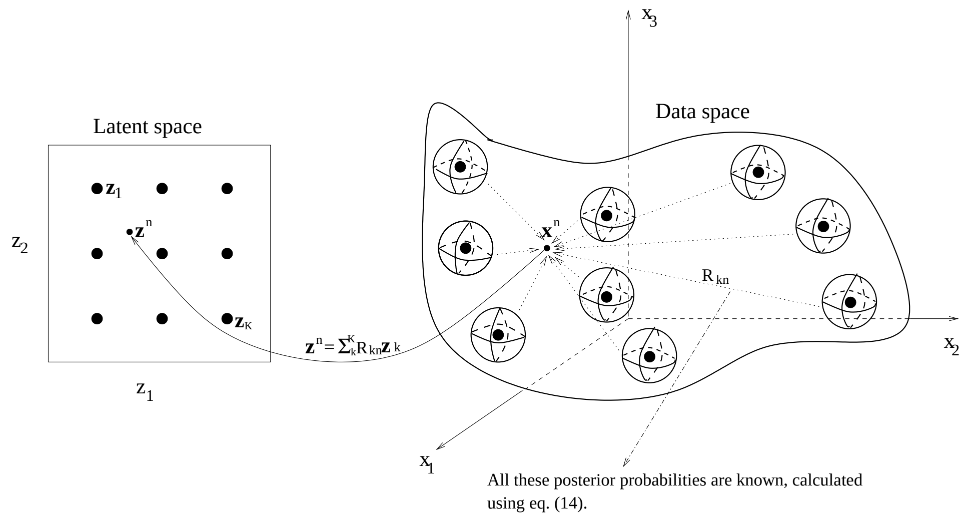
Latent space
z1
zn
zK
z2
z1
zn = ΣkK Rkn z k
xn
Rkn
Data space
x3
x2
x1
All these posterior probabilities are known, calculated using eq. (14).

Fig. 9. An illustration: the latent space representation $\mathbf{z}^n$ of the point $\mathbf{x}^n$ is taken to be the mean of the posterior distribution on $\mathcal{H}$.

given by

$$p(\mathbf{x}|\mathbf{z},\mathbf{W},\beta) = (\frac{\beta}{2\pi})^{D/2} \exp\left\{-\frac{\beta}{2}\|\mathbf{x}-\mathbf{y}(\mathbf{z};\mathbf{W})\|^2\right\}. \tag{11}$$

The prior distribution $P(\mathbf{z})$ of the latent variable $\mathbf{z}$ is a sum of delta functions centred on K

nodes $\{\mathbf{z}_k\}_{k=1,\dots,K}$ of a regular grid in the latent space,

$$P(\mathbf{z}) = \frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} \delta(\mathbf{z} - \mathbf{z}_k). \tag{12}$$

In this way, a non-Gaussian, maximum entropy (uniform) latent prior is imposed over the latent space. The distribution of data in the data space $\mathcal{D}$, for given values of $\mathbf{W}$ and β, is then obtained by

$$P(\mathbf{x}|\mathbf{W}, \beta) = \frac{1}{\mathrm{K}} \sum_{k=1}^{\mathrm{K}} p(\mathbf{x}|\mathbf{z}_k, \mathbf{W}, \beta). \tag{13}$$

The EM algorithm is used to estimate $\mathbf{W}$ and β.

For visualisation purpose, we apply Bayes' theorem to invert the transformation $\mathbf{y}$. Let R_{kn} denote the posterior probability that the $k-$th Gaussian generates $\mathbf{x}^n$ [3], given by

$$R_{kn} = \frac{p(\mathbf{x}^n|\mathbf{z}_k, \mathbf{W}, \beta)P(\mathbf{z}_k)}{\sum_{k'=1}^{K} p(\mathbf{x}^n|\mathbf{z}'_k, \mathbf{W}, \beta)P(\mathbf{z}'_k)}. \tag{14}$$

So the data points can be represented in latent space as the mean position weighted by these probabilities, i.e., $\mathbf{z}^n = \sum_k^K R_{kn}\mathbf{z}_k$.

4.3 Visualisation Using the GTM

In this study, the grid size is $K = 15 \times 15 = 225$, the number of Gaussian functions is $M = 17$, and all Gaussian functions have the same width $\sigma = 1.0$.

In Fig. 10, three distinct *intra-word* clusters can be seen on the bottom right-hand corner, the top left-hand corner and the top middle part, while two easily distinguishable *inter-word* groups can be viewed on the top right-hand corner and bottom middle part. In comparing with Fig. 7, more groups of *intra-word* class are revealed in Fig. 10, and more gaps belonging to *inter-word* are spread out along the diagonal of the plot, which makes them more readily separated from *intra-word* gaps, though there is an overlap in the middle of the plot.

In Fig. 11, the GTM technique is applied to Hard-SGS-19 dataset. As can be seen, more *intra-word* gaps are grouped together, while *inter-word* gaps cover the upper left triangle area of the plot. 4 clusters labelled 1-4 are revealed in the plot. Thus the classification result may be improved with the support of the information in the set of stroke variables.

5 Supervised Classifiers

In this section, we briefly introduce the supervised classifiers used in our experiments. Readers who are interested in those classification techniques can follow the references to learn more.

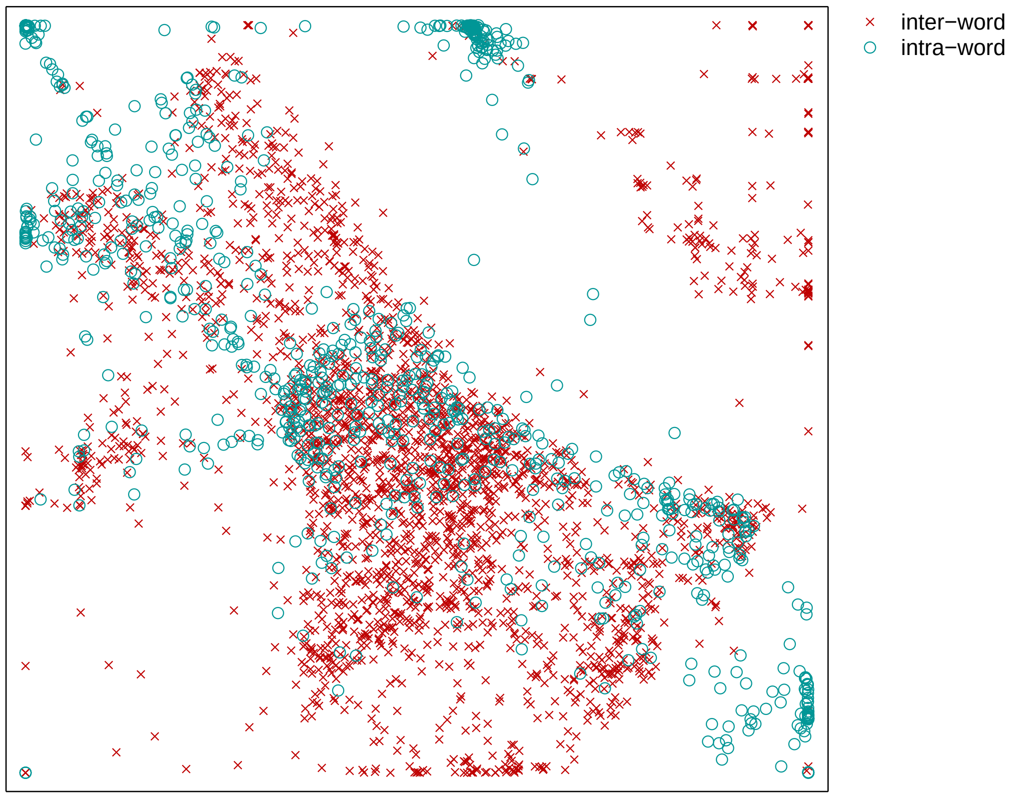

Fig. 10. Visualisation by using the GTM for Hard-G-14 dataset.

Suppose we have a training dataset $\{\mathbf{x}^n, t^n\}, n = 1, \ldots, N$, where $\mathbf{x}^n \in \mathcal{R}^D$ is the input, and t^n is the corresponding target. In classification the task is to assign each new input $\mathbf{x}$ to one of the classes, c_i, where $i = 1, \ldots, C$. We shall denote y as the corresponding predictor of each input.

5.1 Logistic Discrimination Analysis (LDA)

First of all, we consider a simple type of classifier, using a non-linear function $g(\cdot)$ on the weighted sum a of the components of an input, and which can be written as follows [2]:

$$y = g(a) \ , \tag{15}$$

where

$$a = \mathbf{w}^T \mathbf{x} + w_0 \ , \tag{16}$$

and $g(\cdot)$ is an activation function, $\mathbf{w}$ is the weight vector determining the orientation of the separating hyperplane, and w_0 is the bias determining the position of the hyperplane in the data space.

For the two-class problem, one choice for the activation function is the logistic sigmoid acti-

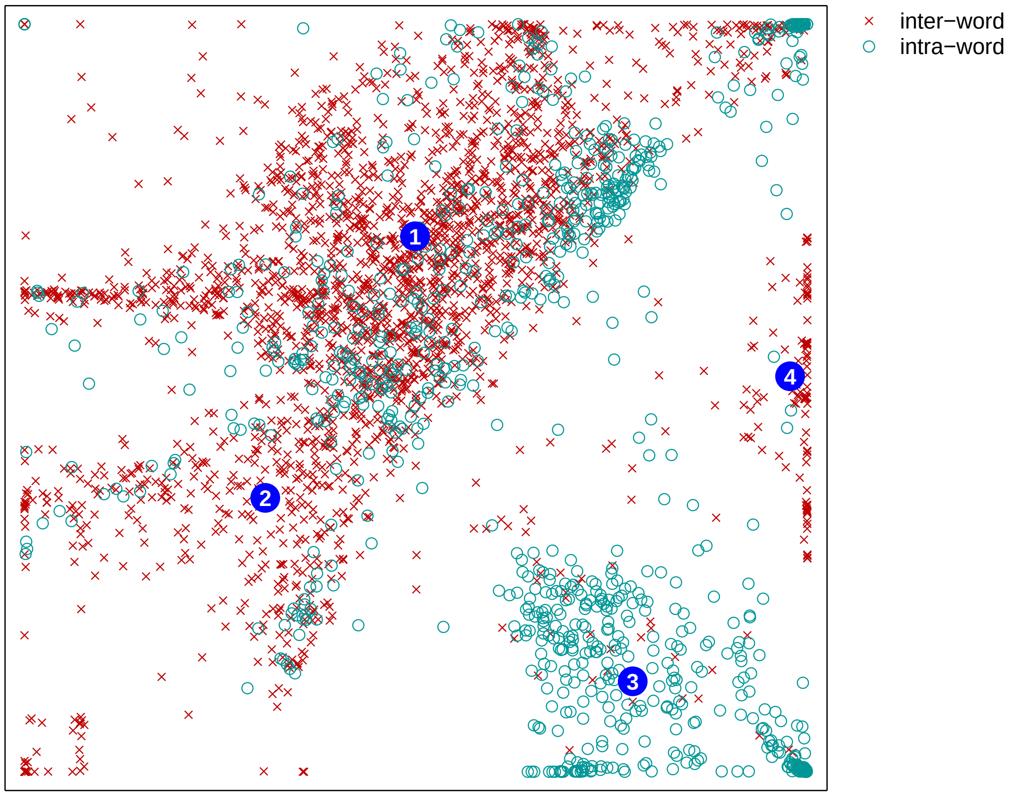

Fig. 11. Visualisation by using the GTM for Hard-SGS-19 dataset.

vation function given by

$$g(a) = \frac{1}{1 + \exp(-a)} \, . \tag{17}$$

The logistic sigmoid function is monotonic, and its interval is $(0, 1)$. It allows the outputs of the discriminant to be interpreted as posterior probabilities.

The logistic discrimination analysis is equivalent to a single layer neural network.

5.2 K-nearest Neighbour Classification (KNN)

Given a test pattern, the algorithm finds the K nearest neighbours among the training patterns based on a similarity measurement, and assigns the correct category by majority vote [13]. In our experiments, we used Euclidean distance to measure the similarity.

5.3 Guassian Mixture Model (GMM)

Classification can be performed by modelling the class-conditional probability $p(\mathbf{x}|c_i)$ for each class first, then by calculating corresponding posterior probabilities using Bayes' theorem.

Since it is usually insufficient to model the conditional density by a single Guassian distribution, we apply a Gaussian mixture model for each class-conditional probability density. In a Gaussian mixture model, the probability density function of each class is independently modelled as a linear combination of Gaussian basis function. The number of basis functions, their position and variance and their mixing coefficients are all parameters of the model. The EM algorithm is used to estimate these parameters for an optimal fit to the training data. An illustration is shown in Fig. 12[1].

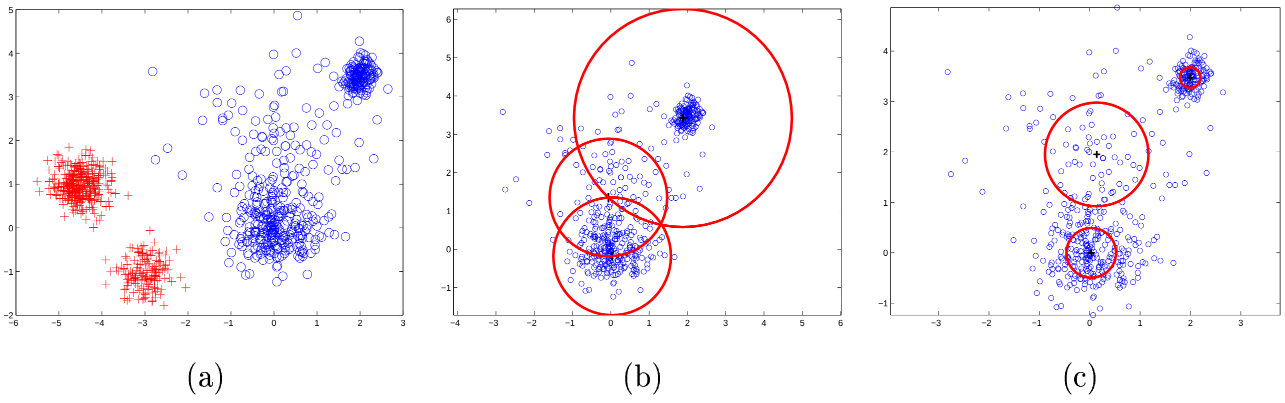

Fig. 12. An example of applying the GMM for fitting data points: (a)1000 data points drawn from two classes, which are presented as plus signs and circles, respectively; (b)the initial configuration of 3 Gaussians of a mixture model which has been initialised using data from one class; (c) final configuration of the Gaussians after 10 iterations of the EM algorithm using the dataset from (b).

In a Gaussian mixture model, $p(\mathbf{x}|c_i)$ is of a linear combination of component densities $p(\mathbf{x}|j, c_i)$, and be written as follows:

$$p(\mathbf{x}|c_i) = \sum_{j}^{M} p(\mathbf{x}|j, c_i)P(j) \ , \tag{18}$$

where for each component j, we have a Gaussian distribution function

$$p(\mathbf{x}|j, c_i) = \frac{1}{\sqrt{(2\pi)^D|\mathbf{\Sigma}_j|}} \times \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\} \ , \tag{19}$$

in which case $\boldsymbol{\mu}_j$ and $\mathbf{\Sigma}_j$ are mean and covariance matrix of each component j respectively. $P(j)$ in equation (18) satisfies (7).

The error function is defined as the negative log-likelihood for the dataset given by

$$E = -\ln \mathcal{L} = -\sum_{n=1}^{N} \ln p(\mathbf{x}^n|c_i) = -\sum_{n=1}^{N} \ln\left\{\sum_{j}^{M} p(\mathbf{x}^n|j, c_i)P(j)\right\}. \tag{20}$$

One practical method to estimate parameters $P(j)$, $\boldsymbol{\mu}_j$, and $\mathbf{\Sigma}_j$ of a mixture model is the EM algorithm, as shown here.

[1] It was implemented using the NETLAB toolbox, which is available from the URL `http://www.ncrg.aston.ac.uk/netlab/`.

In our experiment, we first estimated parameters of each class-condition density from the training dataset. Then we calculate class-conditional density for test inputs by employing equations (19) and (18). Output posterior is obtained by applying equation (9).

5.4 *Multi-layer Perceptron (MLP)*

The MLP is a traditional architecture in neural networks. For classification tasks, it has been proved that networks with sigmoidal activation function and two layers of weights can approximate any decision boundary to arbitrary accuracy [4]. For a given input $(x_1, \ldots, x_D)^T$, the output of the jth hidden unit is given by

$$z_j = h(\sum_{d=1}^{D} w_{jd} x_d + w_{j0}) \ , \tag{21}$$

where w_{jd} are the weights between the input and the hidden layer, and w_{j0} denotes the bias for hidden unit j. Then the output of the network is obtained by

$$y_i = g(\sum_{j=1}^{J} w_{ij} z_j + w_{i0}) \ , \tag{22}$$

where w_{ij} are the weights between the hidden layer and the output, and w_{i0} denotes the bias for the output. We use a *tanh* activation function for $h(\cdot)$ and a *logistic sigmoid* function for $g(\cdot)$.

A suitable error function in classification is the *cross-entropy* function [2]. For two classes, it is given by

$$E = -\sum_{n=1}^{N} t^n \ln y^n + (1 - t^n) \ln(1 - y^n) \ . \tag{23}$$

5.5 *Support Vector Machine (SVM)*

The SVM is a recently developed technique in the machine learning field. The basic idea of the SVM is to find the decision hyperplane that has maximum *margin*: the distance of the closest point to the hyperplane (as shown in Fig. 13). The general form of the decision function for SVM is given by [17]

$$y(\mathbf{x}) = \sum_{n}^{N} \alpha^n t^n k(\mathbf{x}, \mathbf{x}^n) + b \tag{24}$$

with constraints $\alpha^n y^n = 0$ and $0 \leq \alpha^n \leq A$, where b is a threshold, α_i are Lagrange multipliers introduced in a *constrained optimisaton problem*, and A is a constant to determine the trade-off between minimising the training error and maximizing the margin. In equation (24), $k(\mathbf{x}, \mathbf{x}^n)$ is a kernel function, which defines a similarity measure for $\mathbf{x}$ and $\mathbf{x}^n$. The effect of using a kernel function is to implicitly map the data points into a higher-dimensional feature space, and to take the inner-product in that feature space. The potential benefit of using a kernel function is that the data is more likely to be linearly separable in the feature space, and also the actual mapping to the higher-dimensional space is never needed. During the training, only a few α^n are non-zero. Patterns with α^n having non-zero values are called support vectors. In our experiments, we used a Gaussian kernel function.

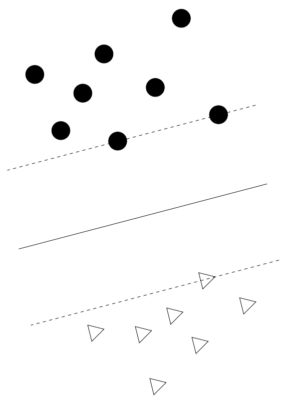

Fig. 13. A binary classification toy problem: separate dots from triangles. The solid line shows the optimal hyperplane. The dashed lines parallel to the solid one show how much one can move the decision hyperplane without misclassification of the data. The patterns on the dotted lines are the support vectors.

6 Experimental Results with Different Classifiers

6.1 Experiments

Experiments were performed on the datasets G-14, G-8, Hard-G-14, Hard-G-8 and Hard-SGS-19. The user-chosen parameters for each classifier were selected by cross-validation, where the training set was divided into 10 partitions. 9 partitions were used to train the model and the other one was used as a validation set. The SVM experiments were completed using LIBSVM, which is available from the URL
`http://www.csie.ntu.edu.tw/čjlin/libsvm`. The others were implemented using the NETLAB toolbox.

In Table 3, we present all the user-chosen parameters attained by using cross-validation.

Table 3
User-chosen parameters from cross-validation. K denotes the number of neighbours in the KNN; $nc1$ and $nc2$ are the number of Gaussian models in each mixture in the GMM; j signifies the number of hidden units in the MLP; A is upper bound of coefficients α_i in the SVM; and σ is width of radial basis in the SVM function.

	KNN (K)	GMM ($nc1$, $nc2$)	MLP (j)	SVM (A, σ^2)
Hard-G-8	9	6, 6	8	25, 0.16
Hard-G-14	9	6, 4	5	20, 0.1
Hard-SGS-19	9	9, 6	5	20, 0.1
G-8	5	8, 9	15	25, 0.25
G-14	5	9, 9	5	5, 0.16

6.2 Classification Results

Classification results for each of the full gap test datasets, with different supervised classifiers, are displayed in Fig. 14, while for each *hard* gap test dataset see Fig. 15. In Fig. 16, the classification result for the Hard-SGS-19 test set is presented along with those for *hard* gap datasets. In all three figures, the accuracy is defined as the percentage of correctly classified patterns over the number of total patterns in the test set. The presented results for the GMM and MLP are averages of 10 repetitions with different random initial conditions.

6.2.1 Classification of the Gap Datasets

Fig. 14 gives the classification accuracies for the gaps dataset with both the full 14 features and the reduced set of 8 features. It can be seen that all but one of the classifiers give only slightly better performance with the G-14 dataset. This confirms that the six attributes removed from the G-8 do not contribute much to the classification. The GMM algorithm performed relatively poorly on both datasets, and interestingly produced a better result for the G-8 dataset.

The results also show that the MLP and SVM provide a more accurate classification than the LDA, KNN and GMM classifiers.

6.2.2 Classification of the Hard Gap Dataset

Fig. 15, gives the classification accuracies for the two hard gap datasets. The relative performance of each classifier on the two different datasets is now more complicated. Only the LDA and SVM classifiers did better on the higher-dimensional dataset (Hard-G-14). All the other classifiers did better on the lower-dimensional dataset (Hard-G-8). The differences are still not great except for the LDA and GMM.

Again, the MLP and SVM perform better than the other classifiers. In fact, the visualisation

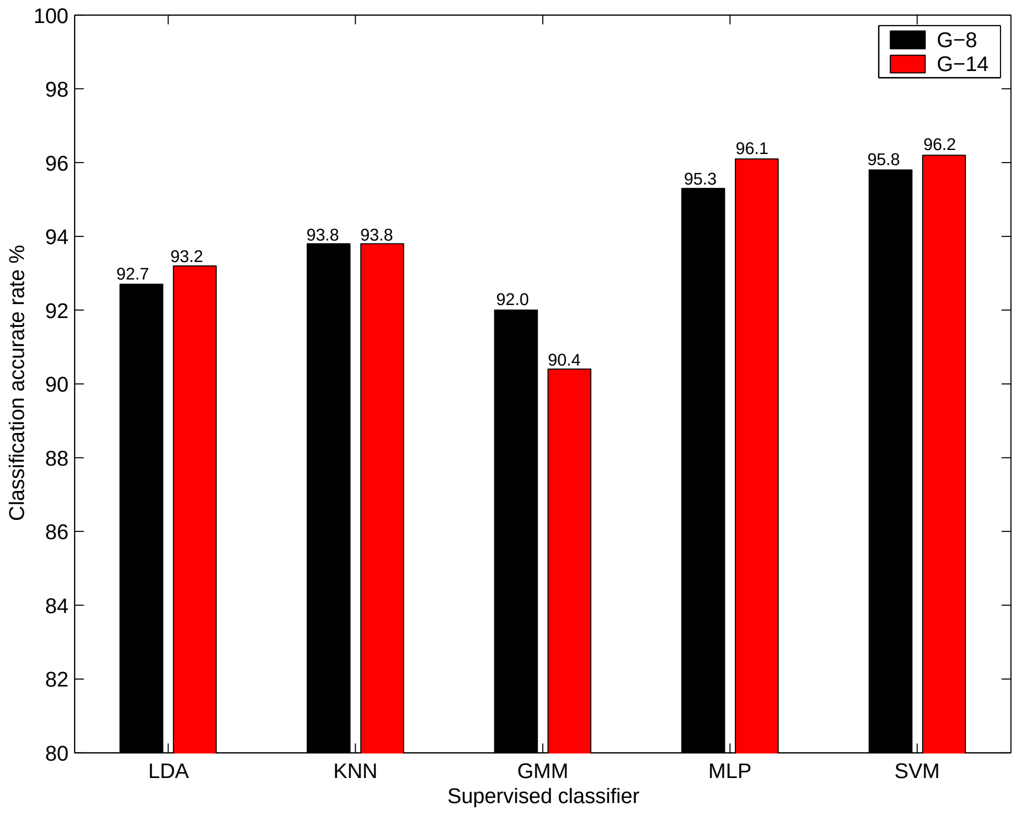

Fig. 14. Bar graph: classification results for each original gaps test dataset with full 14 and selected 8 feature variables. The corresponding accuracy is shown on the top of each bar.

described in Section 4 has already indicated why this may be the case. The visualisation suggests that advanced non-linear classification methods should be applied for these *hard* datasets because of the difficulty of separating the two classes.

Comparing the results between Fig. 14 and Fig. 15, it is clear that the classification rate is higher for the full datasets than the *hard* datasets. This is bound to be the case since the easier to classify points have already been removed from the *hard* datasets. However, one can amalgamate the *hard* dataset results and the *evident* dataset results and still achieve comparable results with the full datasets. The values given in Fig. 15 are the accuracy rate for just the *hard* gaps. Since the rest of original dataset has already been classified with 99% accuracy, the classification for the whole dataset can be calculated. For instance considering the results for the Hard-G-14, the SVM classifier gives a full classification rate for the whole dataset as follows. The whole dataset is 7462 points. 3361 of these are *hard* and are classified by the SVM at 92.5%. The removed gaps are classified by thresholding *river* feature at 99%. Thus an overall classification performance is given by:

$$\frac{3361}{7462} \times 92.5\% + \frac{7462 - 3361}{7462} \times 99\% = 96.1\%.$$

This is almost identical to the result given by working on G-14 directly. However working with the *hard* datasets involves much smaller training times.

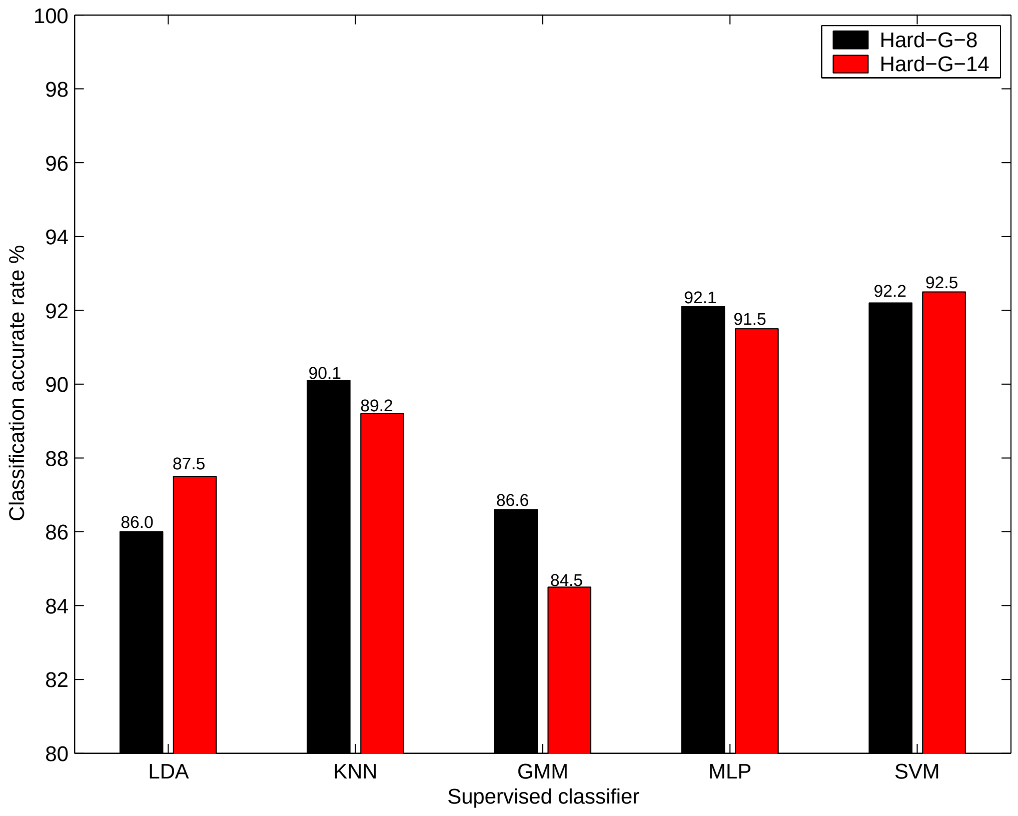

Fig. 15. Bar graph: classification results for each *hard* gap test dataset with 8, 14 feature variables. The corresponding accuracy is shown on the top of each bar.

6.2.3 Classification of the Hard-SGS-19 Dataset

This section adds the results for the stroke-gap-stroke dataset. It can be seen in Fig. 16 that for all but one of the classifiers the results for the SGS data are noticeable better than the gaps only data. In particular, the GMM performs much better on the Hard-SGS-19 dataset.

Furthermore, when considering the *hard* dataset with 19 feature variables, the SVM classifier gives a classification rate 93.7% which we can again amalgamate with the *evident* data points to calculate a final classification rate of 96.6% for the whole dataset. This is our best classification result.

6.3 Statistical Test for Comparing Supervised Classification Learning Algorithms

In this section we investigate the statistical significance of our results for the different classifiers. Looking at Fig. 14 and Fig. 16, it can be seen that there appears to be no big difference between the MLP and SVM algorithms. As addressed in [9], McNemar's test can be used for determining whether one learning algorithm is better than another on a specific task. We therefore use McNemar's test to compare these two algorithms. As a comparison, we provide results of McNemar's test on the KNN and SVM since they do appear to be different.

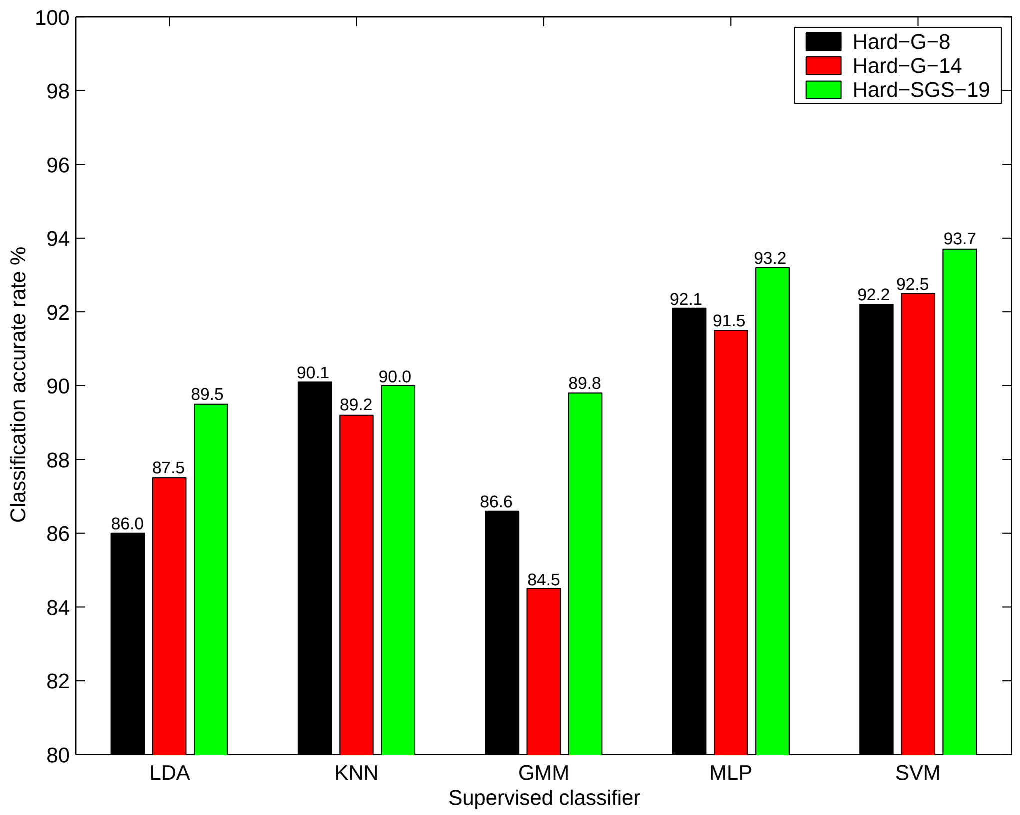

Fig. 16. Bar graph: classification results for each *hard* test dataset with 8, 14, 19 feature variables. The corresponding accuracy is shown on the top of each bar.

The null hypothesis to be tested is that the two algorithms will produce the same error rate on examples from the test set. First the contingency table for the two algorithms I and II, illustrated in Table 4 [9] is calculated. Here n_{00} is number of samples misclassified by both algorithms; n_{01} number of samples misclassified by algorithm I but not II; n_{10} number of samples misclassified by algorithm II but not I; n_{11} are correctly classified by both algorithms.

Table 4
2×2 contingency table

n_{00}	n_{01}
n_{10}	n_{11}

Under the null hypothesis the performance of two different learning algorithms should have the same error rates, i.e., $n_{10} = n_{01}$. McNemar's test is based on a chi-squared test (with 1 degree of freedom) for goodness of fit to the null hypothesis [9]. Chi-Squared (χ^2) is approximated as follows:

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}}. \tag{25}$$

From the *Chi-Squared distribution* table, we have $P(\chi^2 \leqslant 3.84) = 0.95$. So we may reject the null hypothesis and conclude that the algorithms have different performance when χ^2 is greater than 3.84.

Table 5
Results of McNemar's test for comparing the MLP with the SVM and the KNN with the SVM algorithms.

	mlp-svm		knn-svm	
dataset	χ^2	P-value	χ^2	P-value
Hard-G-8	0.77	0.38	7.22	0.0072
Hard-G-14	1.47	0.23	13.28	0.0003
G-8	1.80	0.18	22.52	0.0001
G-14	0.62	0.43	31.65	0.0001

Table 5 displays results comparing both the MLP and the KNN algorithm with the SVM algorithm. The χ^2 for the MLP and SVM is an average calculated over the 10 runs. Looking at the third column, since all the P-values are greater than 0.05, we cannot reject the null hypothesis. This suggests that applying the MLP and SVM learning algorithms to construct classifiers for this application gives equivalent performance. However, looking at the last column the P-values are all smaller than 0.05, so the null hypothesis can be rejected and we can conclude with high probability that the two algorithms do not give equivalent performance.

7 Discussion

Our aim in this piece of work has been to represent the gaps in handwritten text as feature vectors and then to attempt to classify the gaps as either intra or inter word. When the features are analysed we find that one of them, named *river*, has a high level of mutual information with the two classes of gap, suggesting that it will be a good indicator of the class of an unseen example. As shown in Section 3.3 we are able to correctly classify nearly a half of the data to a 99% accuracy from the value of the river feature alone. Specifically those gaps with a *river* value above a particular threshold value are extremely likely to be inter word gaps, and those below a lower threshold are likely to be intra word gaps; such vectors are denoted as *evident* data. However we are left with a large set of still ambiguous gaps. These remaining gaps, which do not have extreme values of the *river* feature, are denoted as the *hard* dataset. A further reduction in the data was achieved by selecting a subset of the features with relatively large mutual information with the gap classes, and we can reduce the original 14 features to the best 8.

Visualisation of the data using a linear PCA projection showed that the clear linear separation for the *evident* data and also the difficulty of separating the *hard* data. However, the visualisation of the more powerful GTM method did show some structural separation of the *hard* data, suggesting that a supervised non-linear classifier could have some success on this more complex dataset.

The results of the supervised classifiers showed that there was little difference in performance on the full set of 14 features as opposed to the reduced set of 8. The most significant result was that the MLP and SVM algorithms performed better than the other three methods used. In fact they managed to correctly classify unseen examples of gaps from the hard dataset at a level of over 92%, with an equivalent classification rate for the whole data at 96.1%.

In order to still further improve the performance of the classifier we took feature representations of the stroke preceding and following each gap. Once again we analysed the mutual information of all the 65 resulting features with the two classes of gap. By taking the 19 best features we were able to include six stroke features. The classification results for the best classifiers, again the MLP and SVM, showed a notable reduction in the error rate - roughly one sixth of misclassifications were removed. Our best classifier, the SVM on the Hard-SGS-19 dataset, in combination with predicting on the basis of the *river* feature on the *evident* dataset gave an overall classification of unseen gaps at 96.7% correct.

Such a high level of word entity identification is likely to be sufficient to support digital ink applications in which character recognition is not used. Indeed the fact that word structure can be identified with less than one error in 30 words and irrespective of overall legibility of text, makes this technique especially suitable for digital ink manipulation at a whole word level. The cost of error (when errors are infrequent) is small: an occasional misclassification would only split a word in two resulting in a minor problem, e.g. a line break mid-word. For a small text area characteristic of Tablet PC and PDA applications, this would happen once or twice in a screen, which would be completely acceptable. On the other hand, since writing on digital media is generally less easy than it is using pen and paper, some support for editing hand-written text, if only at a level of whole word manipulation, is crucial to ensure that stylus-based note-taking and document-processing are accepted by the mainstream user.

References

[1] R. Bellman, *Adaptive Control Processes: A Guided Tour.*, New Jersey: Princeton University Press, 1961.

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*, New York: Oxford University Press, 1995.

[3] C. M. Bishop, M. Svensén and C.K.I. Williams "GTM: the generative topographic mapping", *Neural Computation*, Vol. 10, No. 1, pp215–235, 1998.

[4] E. K. Blum and L. K. Li, "Approximation theory and feedforward networks,", *Neural Networks*, Vol. 4, No. 4, pp511–515, 1991.

[5] T. Butler, *Human Interaction with Digital Ink: Legibility Measurement and Structural Analysis*, Ph.D. thesis, Department of Computer Science, University of Hertfordshire, Hertfordshire, UK.

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, Inc., 1991.

[7] K. Torkkola, "Feature extraction by non-parametric mutual information maximization", *Journal of machine learning Reseach*, Vol. 3, pp1415-1438, 2003.

[8] A. P. Dempster, N. M. Laird and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm”, *J. Roy. Stat. Soc. B*, Vol. 39, pp1–38, 1977.

[9] T. G. Dietterich, “Approximate statistical tests for comparing supervised classification learning algorithms”, *Neural Computation*, Vol. 10, No. 7, pp1895-1923, 1998.

[10] A. S. Fox and C. C. Tappert, “On-line external word segmentation for handwriting recognition”, *Proceedings of the Third International Symposium on Handwriting and Computer Applications*, 1987, pp53–55.

[11] Y. Li, Z. Guan, H. G. Wang, G. Z. Dai and X. S. Ren, “Structuraizing freeform notes by implicit sketch understanding”, *AAAI Spring Symposium on Sketch Understanding*, 2002.

[12] D. Michie, D. J. Spiegelhalter and C. C. Taylor, (Eds), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, 1994.

[13] I. T. Nabney, *Netlab Algorithm for Pattern Recognition*, Springer, 2001.

[14] R. Plamondon and S. N. Srihari, “On-line and off-line handwriting recognition: a comprehensive survey”, *IEEE Transactions on Pattern Analysis and machine Intelligence*, Vol. 22, No. 1, pp63-84, 2000.

[15] D. Rubine, “Specifying gestures by example”, *Computer Graphics*, Vol. 25, No. 4, pp329-337, 1991.

[16] E. Saund, J. Mahoney, D. Fleet, D. Larner and E. Lank, “Perceptual organization as a foundation for intelligent sketch editing”, *AAAI Spring Symposium on Sketch Understanding*, 2002.

[17] B. Scholköpf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, 2002.

[18] C. C. Tappert, C. Y. Suen and T. Wakahara, “The State of the art in on-line handwriting recognition”, *IEEE Transactions on Pattern Analysis and machine Intelligence*, Vol. 12, No. 8, pp787-808, 1990.

[19] P. Tiňo and I. Nabney, “Hierarchical GTM: constructing localized nonlinear projection manifolds in a principled way”, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 24, 2002.