Learning On-Line Handwritten Characters*

Qiong Gao†
Academy of Science, Beijing, China

Ming Li‡
University of Waterloo

Paul M.B. Vitányi§
CWI and University of Amsterdam

Abstract

We report on an experiment in learning to recognize isolated handwritten characters in one subject's handwriting, irrespective of size and orientation. Based on elastic matching, the optimal prediction rate is predicted for the parameters (length of sampling interval) considered most likely by a prior-free form of Bayesian inference, the so-called minimum description length (MDL) principle. There, the most likely hypothesis is the one which minimizes the sum of the length of the description of the hypothesis and the length of the description of the data relative to the hypothesis. This theory is solidly based on a provably ideal method of inference using Kolmogorov complexity. The soundness of the theoretical approach to the problem of learning handwritten characters is evidenced by the fact that the learning parameters achieving the optimal recognition rate as predicted by the theory coincide with the best parameters found experimentally.

1 Introduction

It is commonly accepted that all learning involves compression of experimental data in a compact 'theory' , 'hypothesis', or 'model' of the phenomenon under investigation. In [7, 8] the last two authors analysed the theory of such approaches related to shortest effective description length (Kolmogorov complexity). The question arises

*In part presented at IJCAI, 1989.

†Academy of Science, Beijing, China

‡Supported in part by ONR Grant N00014-85-K-0445 and ARO Grant DAAL03-86-K-0171 at Harvard University, by NSERC operating grant OGP-0036747 at York University, and by NSERC operating grant OGP-046506 at the University of Waterloo. Address: Computer Science Department, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. Email: mli@math.waterloo.edu

§Partially supported by NWO through NFI Project ALADDIN and NSERC International Scientific Exchange Award ISE0125663. Address: Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Email: paulv@cwi.nl

whether these theoretical insights can be directly applied to real world problems. Leslie Valiant suggested to try the problem of recognizing handwritten characters.

The first two authors, in [2], carried out the experiment in on-line learning to recognize isolated characters written in a particular person's handwriting. It is particularly satisfactory that in this case the general learning theory can without much ado be applied. We give a brief account of the general theory and treat how it is applied to learning to recognize on-line handwritten alphanumerical characters.

The genesis of this work is not rooted in traditional approaches to artificial intelligence (AI), but rather on new exciting general learning theories which have developed out from the computational complexity theory [20, 19], statistics and descriptional (Kolmogorov) complexity [11]. These new theories have received great attention in theoretical computer science and statistics, [20, 19, 11, 13, 12, 14, 1, 4]. One the other hand, the design of real learning systems seems to be dominated by *ad hoc* trial-and-error methods. Applications of these recent theoretical results to real world learning system design are scarce and far between. An exception of an elegant paper by Quinlan and Rivest, [10]. Our purpose here is to try and bring theory and practice together by testing some of the theories in real system designs. Specifically, we will apply Rissanen's Minimum Description Length Principle in the design of an on-line hand-written character learning system.

One of the important aspects of AI research is machine cognition of various aspects of natural human languages. An enormous effort has been invested in the problem of recognizing isolated handwritten characters or character strings, [17]. Recognizing isolated hand-written characters has applications, for example, in signature recognition and Chinese character input. The alphanumerical character learning experiment reported here is a pilot project which ultimately aims at providing a practicable method to learn Chinese characters. This problem knows many practical difficulties—both quantitatively and qualitatively.

There are several thousand independent Chinese characters. No current keyboard input method is natural enough for casual users. Some of these methods require the user to memorize a separate code for each of seven thousand characters. Some methods require the user to know *ping ying*—the phonological representation of mandarin Chinese in latin characters. But then the translation into computer representation of characters is not easy because there are too many homonyms. Similarly, sound recognition techniques do not help much either because almost every commonly used Chinese character has more than one commonly used homonym. For non-professional casual users, hand-written input which is mechanically scanned and processed, seems to be a quite reasonable way of entering data into the computer.

A variety of approaches and algorithms have been proposed to achieve a high recognition rate. The recognition process is usually divided into two steps:

1. feature extraction from the sample characters, and

2. classification of unknown characters.

The latter often uses either deterministic or statistical inference based on the sample data, and various known mathematical and statistical approaches can be used. Our contribution is on that level. This leaves the technical problem of feature extraction, whose purpose is to capture the essence from the raw data. The state of the art is more like art than science. Here we use existing methods which we explain now.

1.1 Feature Extraction

The most common significant features extracted are center of gravity, moments, distribution of points, character loci, planar curve transformation, coordinates, slopes and curvatures at certain points along the character curve. The obvious difficulty of the recognition task is the variability involved in handwritten characters. Not only does the shape of the characters depend on the writing style which varies from person to person, but even for the same person trying to write consistently writing style changes with time.

One way to deal with this problem is the idea of 'elastic matching' [6, 18]. Roughly speaking, the elastic matching method takes the coordinates or slopes of certain points approximately equally spaced along the curve of the character drawing as feature to establish the character prototypes. To classify an unknown character drawing, the machine compares the drawing with all the prototypes in its knowledge base according to some distance function and the entered character is classified as the character represented by the closest prototype. When an unknown character is compared to a prototype, the comparison of the features is not only made strictly between the corresponding points with the prototype but also with the points adjacent to the corresponding point in the prototype. The method we use for feature extraction is a new modification of the standard elastic matching method.

1.2 Classification

Each implemented algorithm for character recognition embodies a model or a family of models for character recognition. One problem of the extant research is the lack of a common basis for evaluation and comparison among various techniques. This is especially true for on-line character recognition due to the lack of common standard and limited raw data source. A model from a family of models induced by a particular method of feature extraction is usually specified by a set of parameters. Varying the parameters gives a class of models with similar characteristics.

Consider the above mentioned elastic matching. It uses certain points along the character curve as features. The interval size used to extract these points along the curve is a parameter. How to determine the value of this parameter which gives optimal recognition?

Practically speaking, we can set the interval size to different values and experiment on a given sample set of data to see which value gives the best performance. However, since the experiment is based on one particular set of data, we do not know if this interval size value gives a similar optimal performance for all possible observations from the same data source. A theory is needed to guide the parameter selection in order to obtain the best model from the given class of models.

1.3 Hypothesis Selection

Suppose we have models $M_1, M_2, \ldots$. Let H_i be the hypothesis 'M_i gives the best recognition rate'. Our problem in selecting the best model consists in finding the most likely hypothesis. Rissanen's *Minimum Description Length Principle* (referred as MDLP hereafter) serves this model selection purpose naturally. MDLP finds its root in the well-known Bayesian inference and not so well-known Kolmogorov complexity.

Below we give an impeccable mathematical formula known as 'Bayes' rule'. According to Bayes' rule, a specific hypothesis is preferred if the probability of that hypothesis takes maximum value for a given set of data and a given prior probability distribution over the set of hypotheses. This happens for the hypothesis under which the product of the conditional probability of the data for the given hypothesis plus the prior probability of the hypothesis is maximal. When Bayes' rule is expressed in negative logarithmic form, this maximal probability is achieved by for the hypothesis under which the sum of the following two terms is minimized: the description length of the error of the data for the given hypothesis and the description length of the model (the hypothesis). Therefore, finding a maximum value of the conditional probability of a given set of hypotheses and data becomes minimizing the combined complexity or description length of the error and the model for a given set of candidate models.

To quantify this idea, the two description lengths are expressed in terms of the coding length of the model (set of prototypes) and the coding length of the error (combined length of all data failed to be described by the model). The trade-off between simplicity and complexity of both quantities is as follows.

1. If a model is too simple, in the sense of having too short an encoding, it may fail to capture essence of the mechanism generating the data, resulting in increased error coding lengths.

2. If a model is too complicated, in the sense of having too long code length (like when it consists of a table of all data), it may contain a lot of redundance from the data and become too sensitive to minor irregularities to give accurate predictions of the future data.

The MDLP states that among the given set of models, the one with the minimum combined description lengths of both the model and the error for given set of data is the best approximation of the mechanism behind data and can be used to predict the future data with best accuracy.

The objective of this work is to implement a small system which learns to recognize handwritten alphanumerials based on both elastic matching and statistical inference. Rissanen's MDLP is used to guide the model selection, specifically the selection of the interval of feature extraction. The result is then tested experimentally to validate the application of the theory. We first explain the idea of Bayesian reasoning and give Solomonoff's ideal noncomputable but provably good approach to learning. Then, we will dilute the approach to obtain a feasible modification of it in the form of Rissanen's MDLP. Finally, we apply this principle to obtain our learning system.

2 Bayesianism

Mathematics has come up with an induction principle which has an impeccable derivation, yet allows us to estimate the relative likelihood of different possible hypotheses—which is impossible with the commonly used Pearson-Neyman testing. With the latter tests we accept or reject a zero hypothesis with a given confidence. If we reject the zero hypothesis, then this does not mean that we do accept the alternative hypothesis. We cannot even use the same data to test the alternative hypthesis. (Or a subhypothesis of the alternative hypothesis—because note that all hypotheses different from the zero hypothesis must be taken together to form the alternative hypothesis.) In fact, this type of testing does not establish the relative likelyhood between competing hypotheses at all.

Definition 1 Consider a discrete sample space Ω. Let $D, H_1, H_2, \ldots$ be a countable set of events (subsets) of Ω. $\mathbf{H} = \{H_1, H_2, \ldots\}$ is called *hypotheses space* . The hypotheses H_i are exhaustive (at least one is true). From the definition of conditional probability, that is, $P(A|B) = P(A \cap B)/P(B)$, it is easy to derive **Bayes' formula** (rewrite $P(A \cap B)$ in two different ways):

$$P(H_i|D) = \frac{P(D|H_i)P(H_i)}{P(D)}. \tag{1}$$

If the hypotheses are mutually exclusive ($H_i \cap H_j = \emptyset$ for all i, j), then

$$P(D) = \sum_i P(D|H_i)P(H_i).$$

Despite the fact that Bayes' rule is just a rewriting of the definition of conditional probability and nothing more, it is its interpretation and applications that are most profound and caused much bitter controversy during the past two centuries. In Equation 1, the H_i's represent the possible alternative hypotheses concerning the phenomenon we wish to discover. The term D represents the empirically or otherwise known data concerning this phenomenon. The term $P(D)$, the probability of data D, may be considered as a normalizing factor so that $\sum_i P(H_i|D) = 1$. The term $P(H_i)$ is called the *a priori* probability or *initial* probability of hypothesis H_i, that is, it is the probability of H_i being true before we see any data. The term $P(H_i|D)$ is called *a posteriori* or *inferred* probability

The most interesting term is the prior probability $P(H_i)$. In the context of machine learning, $P(H_i)$ is often considered as the learner's *initial degree of belief* in hypothesis H_i. In essence Bayes' rule is a *mapping* from *a priori* probability $P(H_i)$ to *a posteriori* probability $P(H_i|D)$ determined by data D. In general, the problem is not so much that in the limit the inferred hypothesis would not concentrate on the true hypothesis, but that the inferred probability gives as much information as possible about the possible hypotheses from only a limited number of data. In fact, the continuous bitter debate between the Bayesian and non-Bayesian opinions centered on the prior probability. The controversy is caused by the fact that Bayesian theory does not say how to initially derive the prior probabilities for the hypotheses. Rather, Bayes' rule only tells how they are to be *updated.* In the real world problems, the prior proabilities may be unknown, uncomputable, or even conceivably non-existent. (What is the prior probability of use of a word in written English? There are many different sources of different social backgrounds living in different ages.) This problem would be solved if we can find a *single* probability distribution to use as the prior distribution in each different case, with approximately the same result as if we had used the real distribution. Surprisingly, this turns out to be possible up to some mild restrictions.

2.1 Kolmogorov Complexity

The Kolmogorov complexity, [5, 21, 9], of x is the length of the shortest effective description of x. Formally, this can be defined as follows. Let $x, y, z \in \mathcal{N}$, where $\mathcal{N}$ denotes the natural numbers and we identify $\mathcal{N}$ and $\{0,1\}^*$ according to the correspondence $(0, \epsilon), (1, 0), (2, 1), (3, 00), (4, 01), \ldots$. Hence, the length $|x|$ of x is the number of bits in the binary string x.

In the following we informally describe the needed notions. All formal and rigorous details can be found in [9, 7]. First we fix a universal computer or programming language (like LISP). The *conditional prefix Kolmogorov complexity* of x given y for free, denoted by $K(x|y)$ is the length of the *shortest* prefix-free binary program p that computes x from input y. The *unconditional prefix Kolmogorov complexity* is defined as $K(x) := K(x|\epsilon)$ where ϵ is the empty word. The Kolmogorov complexity is an objective recursively invariant notion by Church's Thesis and the ability of universal machines to simulate one another. Just as the prefix Kolmogorov complexity is the smallest effective description length, the *universal probability* of x conditional the event described by y, denoted by $\mathbf{m}(x|y)$, is the *largest* probability density distribution that can be approximated from below by a computable process. In particular, for every computable probability density function[1] P we have $P(x|y) = O(\mathbf{m}(x|y)$. It turns out that

$$-\log \mathbf{m}(x) = K(x) \pm O(1), \tag{2}$$

That means that $\mathbf{m}$ assigns high probability to simple objects and low probability to complex or random objects. Let us look at some examples for the unconditional version. For example, for $x = 00\ldots0$ (n 0's) we have $K(x) \leq \log n + 2\log\log n + O(1)$ since the program

$$\texttt{print } n\texttt{\_times a `0'}$$

prints x. Then, $\mathbf{m}(x) \geq c/(n \log^2 n)$ for some constant $c > 0$. But if we flip a coin to obtain a y of n bits, then with overwhelming probability $K(y) \geq n - O(1)$ and hence $\mathbf{m}(y) = O(1/2^n)$.

It has been shown by Solomonoff [16] that a variant of $\mathbf{m}$ has astonishing performance in predicting sequences where the probability of the next element is computable from the initial segment. We now come to the punch line: Bayes' rule using the universal prior distribution as suggested by Solomonoff already in [15] yields Occam's Razor principle that can be rigorously shown to work correctly, both in hypothesis identification and in prediction [7].

3 Minimum Description Length Principle

Scientists formulate their theories in two steps. Firstly, a scientist, based on scientific observations, formulates alternative hypotheses (there can be an infinity of alternatives), and secondly a definite hypothesis is selected. The second step is the subject of inference in statistics. Historically this was done by many different principles, like Fisher's Maximum Likelihood principle, various ways of using Bayesian formula (with different prior distributions). Among the most dominant ones is the 'common sense' idea of applying Occam's razor principle of choosing the simplest consistent theory.

However, no single principle is both theoretically sound and practically satisfiable in all situations. Fisher's principle ignores the prior probability distribution (of hypotheses). To apply Bayes' rule is difficult because we usually do not know the actual prior probability distribution. (What is the prior distribution of words in written English, where there are many sources of many ages and social classes?) No single principle turned out to be satisfiable in all situations. Philosophically speaking, Solomonoff's approach presents an ideal way of solving induction problems. However, due to the non-computability of the universal prior function, such a theory cannot be directly used. Some approximation is needed in the real world applications.

[1] This includes every known distribution with computable parameters.

Now we will closely follow Solomonoff's idea, but substitute a 'good' computable approximation to $\mathbf{m}(x)$. This results in Rissanen's **Minimum Description Length** principle [11]. Rissanen not only gives the principle, more importantly he also gives the detailed formulas on how to use this principle. This made it possible to use the MDL principle. The MDL principle can be intuitively stated as follows:

> **Minimum Description Length Principle.** *The best theory to explain a set of data is the one which minimizes the sum of*
> - *the length, in bits, of the description of the theory;*
> - *the length, in bits, of data when encoded with the help of the theory.*

We now develop this MDL principle from Bayes' rule using the universal distribution $\mathbf{M}(x)$, assuming that P is computable. From Bayes' Formula 1, we must choose the hypothesis H such that $P(H|D)$ is maximized. First we take the negative logarithm on both sides of Equation 1:

$$-\log P(H|D) = -\log P(D|H) - \log P(H) + \log P(D).$$

Here, $\log P(D)$ is a constant and hence ignored. Maximizing the $P(H|D)$'s over all possible H's is equivalent to *minimizing* $-\log P(H|D)$, or minimizing

$$-\log P(D|H) - \log P(H)$$

Now to get the minimum description length principle, we only need to explain above two terms in the sum properly. When P is computable, then we approximate P by $\mathbf{m}$. The prior probability $P(H)$ is set to

$$\mathbf{m}(H) = 2^{-K(H)},$$

where $K(H)$ is the prefix-complexity of H. That is, $-\log P(H)$ is about the *length* of a minimum *prefix code*, or program, of hypothesis H.

A similar argument applies to the term $-\log P(D|H)$: That is, $\mathbf{m}(D|H) = 2^{-K(D|H)}$ is a reasonable approximation of $P(D|H)$ provided D is a *typical* outcome for H [9, 7]. For example, if H is the hypothesis that we deal with a fair coin and the data sample D is a hundred outcomes 'heads' in a row, then D isn't typical for H. But if D is a truely random individual sequence with respect to H (a notion that has a precise formal and quantitative meaning), then D is typical for H. The probability of atypical sequences is very small and goes to zero when the data sample grows unboundedly. The term $-\log P(D|H)$, also known as the *self-information* in information theory and the negative log likelihood in statistics, can now be regarded as the number of bits it takes to redescribe or encode D with an ideal code relative to H. In different applications, the hypothesis H can mean many different things, such as decision trees, finite automata, Boolean formulas, or a polynomial.

Example 1 In general statistical applications, one assumes that H is some model $H(\theta)$ with a set of parameters $\theta = \{\theta_1, \ldots, \theta_k\}$ of precision c, where the number k may vary and influence the descriptional complexity of $H(\theta)$. For example, if we want to determine the distribution of the length of beans, then H is a normal distribution $N(\mu, \sigma)$ with parameters median μ and variation σ. So essentially we have to determine the correct hypothesis described by identifying the type of distribution (normal) and the correct parameter vector (μ, σ).

In such cases, we minimize

$$-\log P(D|\theta) - \log P(\theta).$$

Example 2 Let's consider the fitting of a 'best' polynomial on n given sample points in the 2-dimensional plane. This question is notoriously underdefined, since both a 1st degree polynomial with χ^2 best fit, and a $(n-1)$th degree polynomial with perfect fit are arguably the right solutions. But with the MDL principle we can find an objective 'best' polynomial among the polynomials of all degrees.

For each fixed k, $k = 0, \ldots, n-1$, let f_k be the best polynomial of degree k, fitted on points (x_i, y_i) $(1 \le i \le n)$, which minimizes the error

$$error(f_k) = \sum_{i=1}^{n} (f_k(x_i) - y_i)^2.$$

Assume each coefficient takes c bits. So f_k is encoded in $c(k+1)$ bits. Let us interpret the y_i's as measurements for argument x_i of some true polynomial to be determined. Assume that the measurement process involves errors. Such errors are accounted for by the commonly used Gaussian (normal) distribution of the error on y_i's. Thus, given that f is the true polynomial,

$$\Pr(y_1, \ldots, y_n | f, x_1, \ldots, x_n) = \prod \exp(-O((f(x_i) - y_i)^2)).$$

The negative logarithm of above is $c' \cdot error(f)$ for some computable c'. The MDL principle tells us to choose $f = f_m$, with $m \in \{0, \ldots, n-1\}$, which minimizes $c(m+1) + c' \cdot error(f_m)$.

In the original Solomonoff approach a hypothesis H is a Turing machine. In general we must avoid such too general approach to keep things computable. In different applications, H can mean many different things. For example, if we infer decision trees, then H is a decision tree. In case of learning Boolean formulas, then H may be a Boolean formula. If we are fitting a polynomial curve to a set of data, then H may be a polynomial of some degree; In our case, H will be the model for a particular character. Each such H can be encoded by a binary string from a prefix-free set, where a set of codes is *prefix-free* if no code in the set is a prefix of another.

Above we showed that Solomonoff suggested to assign to H the prior probability $\mathbf{m}(H) = 2^{-K(H)}$, where $K(H)$ is the length of shortest effective prefix-free description of H. Then, $-\log P(H)$ is about the minimum *length* of an effective *prefix code* of the hypothesis H. More precisely $K(H)$ is the prefix Kolmogorov complexity of H as defined above. By the famous Kraft Inequality which governs the conditions for the existence of prefix-codes based on the lengths of the prospective code word set, $\sum_{H_i} 2^{-K(H_i)} \leq 1$. This means that the proposed probability distribution is proper. The optimality of this probability assignment is also proved in the sense that this probability assignment approximates any other *computable* probability distribution in a precise sense.

By Markov's inequality, for any two probability distributions P and Q, for all k, we have $Q(x) < P(x)/k$ with P-probability at least $1 - 1/k$. By definition of the universal density distribution $\mathbf{m}$ therefore, for each computable probability distribution $P(x)$ we have

$$\sum \{P(x) : 2^{K(P)}\mathbf{m}(x) \geq P(x) \geq \mathbf{m}(x)/k\} \geq 1 - 1/k,$$

for all $k > 0$. In this sense, with high P-probability, $P(x)$ is close to $\mathbf{m}(x)$, for each computable P. The distribution $\mathbf{m}$ is the unique (up to multiplicative constant) distribution which has that property. If the problem instances are generated algorithmically, then the distribution is computable. In absence of any a priori knowledge of the actual distribution therefore, apart from that it is computable, studying the average behavior under $\mathbf{m}$ is considerably more meaningful than studying the average behavior under any other particular computable distribution (like the uniform one).

However, all this is too good to be true, and there is bound to be some drawback. This is the fact that it is known that $K(H)$ is not computable. Hence, such an approach can hardly be practical. Jorma Rissanen suggested the following approach to approximate the Solomonoff idea. First convert (or encode) H to an integer belonging to $\mathcal{N} = \{0, 1, 2, ...\}$. Then we try to assign a prior distribution to each integer in $\mathcal{N}$. A previous suggestion by Jeffreys, [3], of assigning probability $1/x$ to integer x results in an improper distribution since $\sum_{x=1}^{\infty} 1/x$ diverges. Rissanen suggested to assign prior probability $2^{-L(x)}$ to x, where

$$L(x) = \log^* x + \log c$$

where $\log^* x = \log x + \log\log x + \ldots$, the sum including all the positive iterates, and $c = 2.865064....$ This way, we obtain the following desired properties:

1. $\sum_{x=1}^{\infty} 2^{-L(x)} = 1$, and
2. integer x is coded by a prefix code.

The coded descriptions of two integers, x_1 and x_2, can now simply be concatenated to produce the code (like $< x_1, x_2 >$) for the pair x_1, x_2.

To summarize, we have derived Rissanen's **Minimum Description Length Principle** which is informally stated as follows:

> *The best theory to explain a set of data is the one which minimizes the sum of*
>
> - *the length (encoded in bits) of the description of the theory—the term* $-\log_2 P(H)$*;*
> - *the length (in bits) of encoding of data with the help of the theory—the term* $-\log_2 P(D|H)$*, the error coding length.*

4 Model Development

4.1 Basic Assumptions

When an alphanumeral character is drawn on a planar surface, it can be viewed as a composite planar curve, the shape of which is completely determined by the coordinates of the sequence of points along the curve. The order of the sequence is determined by on-line processing the data from the scanning machinery at the time of writing the character. Since the shape tends to vary from person to person and from time to time, so do the coordinates of the point sequence. A key assumption in our treatment is that for a particular person writing consistently the shape of the curve tends to converge to an average shape, in the sense that the means of corresponding coordinates of the sampled point sequences converge.

That is, we assume:

- Each shape of an example curve for a particular character contains a set distinguished feature points.
- For each such point, the average of the instances in the different examples converges to a mean.
- Moreover, there is a *fixed* probability distribution (possibly unknown) for each such point which is symmetric about the mean value, and the variance is assumed to be the same for all the character drawings.

Essentially, we only assume that one person's hand-writing has a *fixed* associated probability distribution, which does not change.

4.2 Feature Space, Feature Extraction and Prototypes

A Kurta ISONE digitizer tablet with 200/inch resolution in both horizontal and vertical directions was used as the scanner to obtain and send the coordinates of the sequential points of the character curve on the tablet to the microprocessor of a IBM PS/2 model 30 computer. The system was implemented using programming language C. The coordinates were normalized on a 30 by 30 grid in horizontal and vertical directions.

The character drawn on the tablet is processed on-line. The sequence of the coordinates in order of time of entry is stored in the form of a linked list. This list is preprocessed in order to remove the repeating points due to hesitation at the time of writing, and to fill in the gaps between sampled points resulted from the sampling rate limit of the tablet.

The latter needs some explanation: the digitizer has a maximum sampling rate of 100 points/second. If a person writes a character in 0.2 seconds, only 20 points on the character curve will be sampled, leaving gaps between those points.

The preprocessing procedure ensures that in the resulting linked list any pair of consecutive points on the curve has at least one component of coordinates (stored as integers between 0 and 30) differing by 1 and no coordinate component differing by more than 1. For a preprocessed list $((x_1, y_1), \ldots, (x_n, y_n))$ therefore we have that for all i $(1 \leq i < n)$

$$\begin{aligned} |x_i - x_{i+1}| + |y_i - y_{i+1}| &\geq 1 \\ |x_i - x_{i+1}|, |y_i - y_{i+1}| &\leq 1 \end{aligned}$$

The preprocessed curve coordinate list is then sent to the feature extraction process. So far the coordinates are still integers in the range of 0 to 30.

The coordinates of certain points along the character curves are taken as relevant *features*. Feature extraction is done as follows. A character may consist of more than one stroke (a stroke is the trace from a pen drop-down to pen lift-up), the starting and ending points of every stroke are mandatorily taken as features. In between, feature points are taken at a fixed interval, say, one point for every n points along the preprocessed curve, where n is called **feature extraction interval**. This is to ensure that the feature points are roughly equally spaced. Actually the Euclidean length between any two points on the stroke curve, excluding the last point of a stroke, varies from n to $\sqrt{2}n$ (for the diagonal).

The sequence of the feature point coordinates extracted from a given character drawing constitute a *feature vector*. (If the character drawing contains more than one stroke, its feature vector consists of the concatenation of the feature vectors of the individual strokes in time-order.) The dimension of the feature vector is the number of entries in it—or rather twice that number since each entry has two coordinate components. Obviously the dimension of the feature vector is also a random variable since the shape and the total number of points on the character curve varies from time to time. The dimension of the feature vector is largely determined by the feature extraction interval.

The extracted feature vector of a character is viewed as a *prototype* of a character, and is stored in the knowledge base of the system as such.

4.3 Comparison between Feature Vectors

Before the system is employed to recognize characters, it must first learn them. It is trained with examples of the character drawings from the same data source which it is supposed to recognize afterwards. Here the 'same data source' means the same person writing consistently. The basic technique used in both training and recognition is the comparison or matching between prototypes or feature vectors. To compare any two prototypes or feature vectors of equal dimension, we can simply take the Euclidean distance between the two vectors. Mathematically this means subtracting each component of one vector from its corresponding component in the other feature vector, summing up the square of the differences and taking the square root of the sum. If the two prototypes are $\chi = ((x_1, y_1) \ldots, (x_n, y_n))$ and $\chi' = ((x'_1, y'_1) \ldots, (x'_n, y'_n))$, then the distance between them is

$$\sqrt{\sum_{i=1}^{n} (x_i - x'_i)^2 + (y_i - y'_i)^2}.$$

The knowledge base of the system is a collection of feature vectors stored in the form of a linked list. Each such feature vector is an example of a particular character and is called a *prototype* for that character. Each newly entered character drawing in the form of a feature vector is compared to the prototypes in the knowledge base. But we do not (cannot) assume that all feature vectors extracted from examples of the same character will have the same dimension.

Therefore, the comparison technique used in our system follows the spirit of this mathematical definition but is more elastic. As a consequence, corresponding feature points may be located in different places in the feature vectors. This problem is solved by so-called *elastic matching* which compares a newly sampled feature vector with the set of stored feature vectors, the *prototypes*. The elasticity is reflected in two aspects:

Dimension tolerance is a constant integer T_d such that the new feature vector is compared with all stored feature vector of which the dimension is not more than T_d different. That is, if the new feature vector has n feature points, it will be compared (matched) with all the prototypes with a number of feature points in the range of $[n - T_d, n + T_d]$.

Local extensibility is an integer constant N_e such that the ith feature point of the new feature vector is compared with the feature points with index ranging from $i - N_e$ to $i + N_e$ of each prototype satisfying the dimension tolerance. The least Euclidean distance found this way is considered to be the 'true' difference d_i between the two vectors at ith feature point.

Definition 2 If the dimension of the new feature vector x is n, then the *elastic* *distance* $\delta(x, x')$ between x and a prototype x' is defined as

$$\delta(x, x') = \sqrt{\sum_{i=1}^{n} d_i^2}$$

if x' is within the dimension tolerance T_d of x, and $\delta(x, x') = \infty$ otherwise.

For our particular problem, experimental evidence indicates that it suffices to set both T_d and N_e to 1. In our experiment we used elastic distance between a new feature vector x and a prototype x' as computed above with $T_d = N_e = 1$.

4.4 Knowledge Base and Learning

The knowledge base is constructed in the learning phase of the system by sampling feature vectors of handwritten characters while telling the system which character it is an example of. Our system uses the following **Learning Algorithm** to establish the knowledge base.

Step 0. Initialize the knowledge base S to the empty set $\emptyset$. (The elements of S will be triples (x, χ, c) with x a preprocessed feature vector (a prototype), χ is the character value of which x is a prototype, and c is a counter.) Assign values to weights α, β (used later to combine prototypes) so that $\alpha + \beta = 1$.

Step 1. Sample a new example of a character feature vector, say x after preprocessing, together with its character value, say χ. (Actually, the user draws a new example handwritten character on the tablet and indicates the character value—which character the drawing represents—to the system.)

Step 2. Check S whether or not any prototypes exist for character χ.

If there is no prototype for χ in S, then store x in S as a prototype for χ by setting

$$S := S \cup \{(x, \chi, 1)\}.$$

If $P_x = \{y, \ldots, z\}$ is a nonempty list of prototypes for χ in S, then determine elastic distances $\delta(x, y) \ldots \delta(x, z)$. Let $P_x^{min} \subseteq P_x$ be the set of prototypes in P_x such that for all $x' \in P_x^{min}$ we have

$$\delta(x, x') = \delta_{min} \stackrel{\text{def}}{=} \min\{\delta(x, y) : y \in P_x\}.$$

Now $x' \in P_x^{min}$ may or may not be one of the prototypes with the character value χ.

Step 2.1. If $x' \in P_x^{min}$ and $(x', \chi, m) \in S$ (the minimum distance δ_{min} is between x and one of the prototypes for χ in S), then

$$x' := \alpha x' + \beta x; m := m + 1.$$

(The new prototype is combined with the existing prototype by taking the weighted average of every coordinate to produce a modified prototype for that character. Moreover, we add one to the counter associated with this prototype)

Step 2.2. If Step 2.1 is not applicable, then for no $x' \in P_x^{min}$ we have $(x', \chi, \cdot) \in S$ (the minimum distance is between x and prototype(s) in S with character value $\neq \chi$). Then we distinguish two cases.

Case 1. There is $(x', \chi, m) \in S$ such that $\delta(x, x') \leq \delta_{min}(m+1)/m$. (Here m is number of character drawings which have consecutively be combined to form the current x' prototype.) Then set

$$x' := \alpha x' + \beta x; m := m + 1.$$

It is expected that the modified prototype will have minimal distance δ_{min} next time when a similar drawing of the same character value arrives.

Case 2. The condition in Case 1 is not satisfied. Then the new prototype will be saved in the knowledge base as a new prototype for the character by setting

$$S := S \cup \{(x, \chi, 1)\}.$$

Notice that more than one prototype for a single character may co-exist in the knowledge base.

4.5 Recognition of an Unknown Character Drawing

The recognition algorithm is simple. Assume the Learning Algorithm above has been executed, and a knowledge base S with prototypes of all possible characters has been constructed. When an new character drawing is presented to the system, it is compared to all the prototypes in the knowledge base with dimension variation within the range specified by the dimension tolerance. The character of the prototype which has minimum δ-distance from the presented feature vector is considered to be the character value of that feature vector. The rationale here is that the prototypes are considered as the 'mean' values of the feature vectors of the characters, and the variances of the distribution are assumed to be the same for all prototypes. Formally, the **Recognition Algorithm** is as follows.

Step 0. Sample a new example of a character feature vector, say x after preprocessing. (Actually, the user draws a new example handwritten character on the tablet which is preprocessed to form feature vector x.)

Step 1. If $S = \{(x_1, \chi_1, \cdot), \ldots, (x_n, \chi_n, \cdot)\}$ is the knowledge base, then determine elastic distances $\delta(x, x_1) \ldots \delta(x, x_n)$. If $\delta(x, x_i)$ is the minimal distance in this set, with i is least in case more than one prototype induces minimum distance, then set χ_i is the character value for x and

$$\text{Recognize character } \chi_i.$$

This concludes the main procedure of training and classification. A few remarks are in order to explain differences with the original elastic matching method.

1. This process differs from the original elastic matching method in the the way of prototype construction. More than one prototype are allowed for a single character. By our procedure in theLearning Algorithm, a prototype is the *statistical mean* of a number of positive examples of the character.

2. Every prototype is a feature vector which in turn is a point in the feature space of its dimension. Since the classification is based on statistical inference, the rate of correct classification depends not only on how well the prototypes in the knowledge base are constructed, but also on the variability of the handwriting of the subject. Even though more than one prototype is allowed for any character in the knowledge base, too many prototypes may result in an overly dense feature space. When the δ-distance between two points (two prototypes in the knowledge base) in the feature space is comparable to the variability of the subjects handwriting, then the rate of correct classification may drop considerably.

3. The prototypes in the knowledge base constitute the model for the system. How well the prototypes are constructed will essentially determine the rate of correct classification and therefore the performance of the model. For the scheme described above, the prototypes are constructed by extracting points at a constant interval. Generally speaking, more points in the prototypes gives a more detailed image of the character drawing but may also insert random 'noise' in the model.

   Application of Rissanen's MDLP to guide the selection of 'best' feature extraction interval is the main thrust of this work, to which we proceed now.

5 Implemented Description Lengths and Minimization

The expression in Rissanen's MDLP consists of two terms: the model and error coding lengths. The coding efficiency for both of these two terms must be comparable, otherwise minimizing the resulted expression of total description length will give either too complicated or too simple models. For this particular problem, the coding lengths are determined by practical programming considerations.

A set of 186 character drawings, exactly 3 for each of the 62 alphanumeral characters, were processed to feature vectors and presented to the Learning Algorithm, to form the raw database. The character drawings were stored in standardized integer coordinate system standardized from 0 to 30 in both x and y axis. Aftre preprocessing as above, they were then input to the Learning Algorithm to establish a knowledge base: the collection of prototypes with normalized real coordinates, based on a selected feature extraction interval.

Subsequent to the construction of the knowledge base, the system was tested by having it classify the same set of character drawings using the Recognition Algorithm. This procedure served to establish the error code length and the model code length which are defined as follows.

Definition 3 The *error code length* or *exception complexity* is the sum of the total number of points for all the incorrectly classified character drawings. This represents the description of the data given the hypothesis.

The *model code length* or *model complexity* is the total number of points in all the prototypes in the machine's knowledge base multiplied by 2. This represents the hypothesis.

The *total code length* is the sum of the error code length and the model code length.

Remark 1 The factor of 2 in the model code length is due to the fact that the prototype coordinates are stored as real numbers which takes twice as much memory (in programming language C) as the character drawing coordinates which are represented in integer form. One might wonder why the prototype coordinates are real instead of integer numbers. The reason is to facilitate the elastic matching to give small resolution for comparisons of classification.

Thus, both the model and error code lengths are directly related to the feature extraction interval. The smaller this interval, the more complex the model, but the smaller the error code length. The effect is reversed if the feature extraction interval goes toward larger values. Since the total code length is the sum of the two code lengths, there should be a value of feature extraction interval which minimizes the total code length.

This feature extraction interval is considered to be the 'best' one in the spirit of MDLP. The corresponding model, the knowledge base, is considered to be optimal in the sense that it contains enough essence from the raw data but eliminates most redundancy due to noise from the raw data. This optimal feature extraction interval can be experimentally determined by carrying out the above described build-and-test (building the knowledge base and then test it based on the same set of characters on which it was built) for a number of different feature extraction intervals.

The actual optimization process was implemented on the actual system we constructed, and available to the user. For our particular set of characters and trial, the results of classifying by the Recognition Algorithm the same set of 186 character drawings used by the Learning Algorithm to establish the knowledge base, is given in Figure 1. Three quantities are depicted: the model code length, the error code length, and the total code length, versus different feature extraction intervals (SAMPLING INTERVAL in the figure). For larger feature extraction intervals, the model

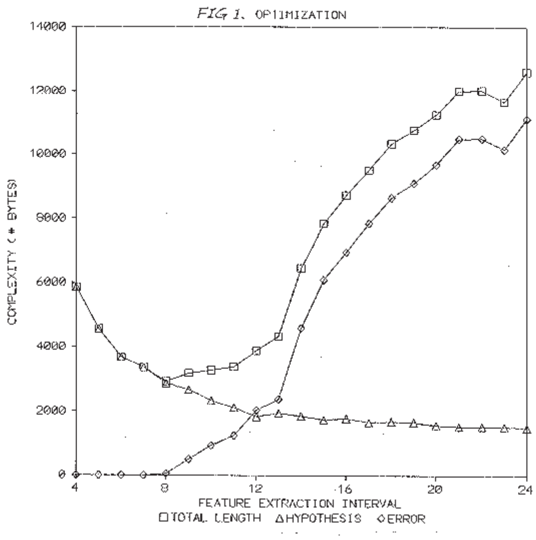

Figure 1: Experimentally Determined Error- and Model Code Lengths

complexity is small but most of the character drawings are misclassified, giving the very large error code length and hence the very large total code length. On the other hand, when the feature extraction interval is at its low extremal value, all training characters get correctly classified which gives zero error coding length. But now the model complexity reaches its largest value, resulting also in a large total code length again. The minimum total code length occurred in our experiment at an extraction interval of 8, which gives 98.2 percent correct classification. Figure 2 illustrates the fraction of correctly classified character drawings for the training data.

6 Validation of the Model

Whether the 'optimal' model, determined by choosing the interval yielding minimal total code length for the training data, really performs better than models in the same class using different feature extraction intervals, can be tested by classification of new data—new character drawings.

We have executed such a test by having the set of 62 characters drawn anew by the same person who provided the raw data base to build the knowledge base. After preprocessing, the feature vectors resulting from these data were entered in the Recognition Algorithm. The new data are considered to be from the same source as the previous data set.

This new data set was classified by the system using the knowledge bases built by the Learning Algorithm from the training data set of 186 character drawings, based

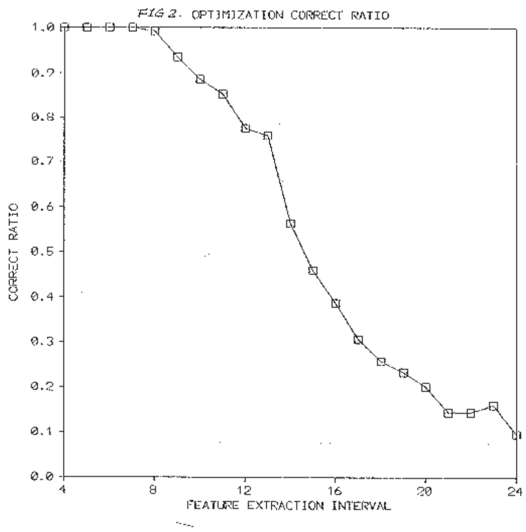

Figure 2: Fraction Correctly Classified Training Data

on different feature extraction intervals. The test results are plotted in Figure 3 in terms of the fraction of correct classification (CORRECT RATIO) versus feature extraction interval (SAMPLING INTERVAL). It is interesting to see that a 100% correct classification occurred at feature extraction intervals 5, 6 and 7. These values of feature extraction intervals are close to the optimal value 8 resulting from MDLP considerations. Furthermore, at the lower feature extraction intervals, the correct classification rate drops, indicating the disturbance caused by too much redundance in the model. The recommended working feature extraction interval is thus either 7 or 8 for this particular type of character drawings.

7 Summary

Rissanen's Minimum Description Length Principle is applied to learning to recognize isolated handwritten characters on-line using elastic matching and some statistical technique. A 'model' is a collection of prototypes built from raw training character drawings by on-line taking points on the curves of the executed character drawing at a constant feature extraction interval, and by combining closely related character drawings.

The model is optimized in the spirit of MDLP by minimizing the total code length, which is the sum of the model and exception code lengths, against different feature extraction intervals. The resulted model is optimal according to the theory. It is then validated by testing using a different set of character drawings from the

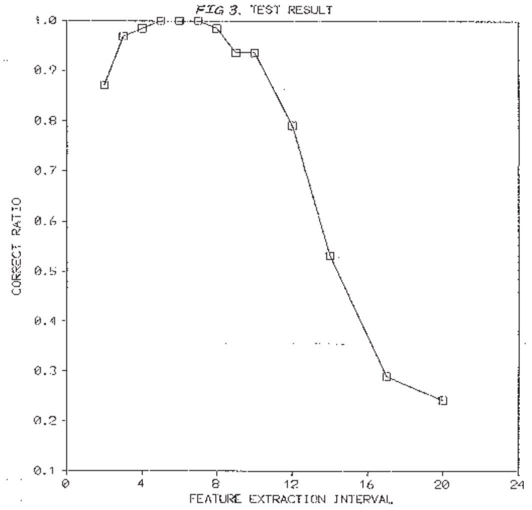

Figure 3: Fraction Correctly Classified New Test Data

same source. The result of this test validates MDLP as a good tool in the area of handwritten character recognition.

Acknowledgement

We are grateful to Les Valiant for many discussions on machine learning and the suggestion for this research.

References

[1] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. *J. Assoc. Comput. Mach.*, 35:929–965, 1989.

[2] Q. Gao and M. Li. An application of minimum description length principle to online recognition of handprinted alphanumerals. In *11th International Joint Conference on Artificial Intelligence*, pages 843–848. Morgan Kaufmann Publishers, 1989.

[3] H. Jeffreys. *Theory of Probability*. Oxford at the Clarendon Press, Oxford, 1961. Third Edition.

[4] M. Kearns, M. Li, L. Pitt, and L. Valiant. On the learnability of boolean formulae. In *Proc. 19th ACM Symp. Theory of Computing*, pages 285–295, 1987.

[5] A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems Inform. Transmission*, 1(1):1–7, 1965.

[6] J. Kurtzberg. Feature analysis for symbol recognition by elastic matching. *IBM J. Res. Develop.*, 31(1):91–95, 1987.

[7] M. Li and P.M.B. Vitányi. Minimum Description Length Induction, Bayesianism, and Kolmogorov Complexity, Submitted to *IEEE Trans. Information Theory.* (See http://www.cwi.nl/ paulv/learning.html)

[8] M. Li and P.M.B. Vitányi. Inductive reasoning and Kolmogorov complexity. *J. Comput. Syst. Sci.*, 44:343–384, 1992.

[9] M. Li and P.M.B. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*, Springer-Verlag, New York, 2nd Edition, 1997.

[10] J. Quinlan and R. Rivest. Inferring decision trees using the minimum description length principle. *Inform. Computation*, 80:227–248, 1989.

[11] J. Rissanen. Modeling by the shortest data description. *Automatica-J.IFAC*, 14:465–471, 1978.

[12] J. Rissanen. Universal coding, information, prediction and estimation. *IEEE Transactions on Information Theory*, IT-30:629–636, 1984.

[13] J. Rissanen. Minimum description length principle. In S. Kotz and N.L. Johnson, editors, *Encyclopaedia of Statistical Sciences, Vol. V*, pages 523–527. Wiley, New York, 1986.

[14] J. Rissanen. Stochastic complexity. *J. Royal Stat. Soc., series B*, 49:223–239, 1987. Discussion: pages 252-265.

[15] R.J. Solomonoff, A formal theory of inductive inference, Part 1 and Part 2, *Inform. Contr.*, 7(1964), 1-22, 224-254.

[16] R.J. Solomonoff. Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Inform. Theory*, IT-24:422–432, 1978.

[17] C.Y. Suen. Automatic recognition of handprinted characters—the state of art. *Proc. IEEE*, 68(4):469–487, 1980.

[18] C.C. Tapper. Cursive script recognition by elastic matching. *IBM J. Res. Develop.*, 26(6):765–771, 1982.

[19] L.G. Valiant. Deductive learning. *Phil. Trans. Royal Soc. Lond.*, A 312:441–446, 1984.

[20] L.G. Valiant. A theory of the learnable. *Comm. Assoc. Comput. Mach*, 27:1134–1142, 1984.

[21] A.K. Zvonkin and L.A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Math. Surveys*, 25(6):83–124, 1970.