
Offline Cursive Handwriting Recognition using Holistic Features and Learning Vector Quantization

Caesar Ogole

Department of Computing Science
University of Groningen,
The Netherlands
C.Ogole@student.rug.nl

Abstract

Enhanced techniques for processing and extraction of features from images, combined with robust classification algorithms can be of paramount importance in handwriting recognition. Such methods are treated in this report for automatic recognition of scanned offline cursive handwriting. Specifically, a word-based approach to feature extraction and learning vector quantization machine learning algorithm are applied to construct a recognizer for historic handwriting from the Royal Dutch Queen's archive. Included in this paper is a description of the results of our experiments together with an overview of the methods used for data acquisition and representations, construction and training of the classifier network, and recognition of an unseen instance.

Keywords: offline cursive handwriting recognition, holistic features, learning vector quantization

1 Introduction

Humans have differing and varying styles of writing. Whereas some people write texts with discretely spaced successive letters, a greater number of people make scribbles of letters that are connected to each other. We refer to the latter type of handwriting as *cursive handwriting*, and the former may be called *isolated hand-printing*. Variations and variability in handwriting crop up due to style variations (allographs), affine transforms, neuro-biomechanical and temporal order (sequencing) variations of the handwritten characters. It's even more interesting to note that these differences occur not only in texts written by different people but also for scripts written by the same person. *Between-writer style variation* and *within-writer variability*, respectively, are the terms used to describe the phenomena.

In handwriting recognition, the task is to automatically classify a handwritten text that is (often) given in the form of an image. The image of a handwritten text is obtained by scanning (using flat-bed scanner). In this case, the handwritten text is called *offline handwriting*. In contrast, *online handwriting* is one obtained by (writing) using stylus pens and digitizing tablets. In general, the problem of automatic script recognition is difficult because of the large variations in personal handwriting style, different writing instruments, segmentation problem and the large vocabulary in human languages.

In this study, we explore some plausible methods for constructing recognizers for offline cursive handwriting. Connected-cursive scripts are

typically found in official documents, personal notes and communicating letters. Automatic recognition of handwritten texts has important industrial applications such as address field recognition on envelopes, text input on small hand-held devices (organizers, pen-based mobile phones), writer identification and signature verification, transcription of (offline) historical archives, archive cards, genealogical data, and journals, among others.

It is to be noted that automatic recognition of offline cursive scripts is more even challenging because, unlike in online handwriting, offline handwritten texts do not contain information such as the order and number of strokes, the changes in velocity and pressure, and other (such) dynamic properties.

We also note that ability to automatically recognize *pure cursive handwriting* with high certainty can be considered a big advance in the handwriting recognition research area since other types of handwriting (such as hand-printing) are simple versions of the pure cursive script.

We try to solve the problem of offline cursive handwriting recognition by combining enhanced versions of the techniques commonly used in image processing, feature extraction and classification of multi-dimensional data. Figure 1 shows the three main modules in a typical recognition system. In section 2 of this report, we describe the preprocessing and feature extraction methods that were used in our experiments. Specifically, this section seeks to provide an answer to the pervasive problem of segmentation in automatic script

recognition. Section 3 then tackles the problem of effectively recognizing preprocessed novel instances of a word image. For this purpose, we describe in this section learning vector quantization, an algorithm used for classifying and recognizing high dimensional data. The experiments and results are described in section 4 of this report. We conclude the study by making a summary of the study and some general remarks on the outlook on this very interesting research area.

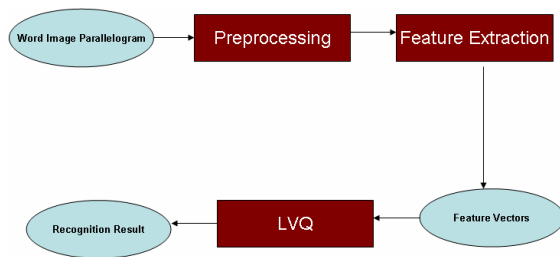


Fig. 1: a simplified recognition process

The pre-processing step is essential in reducing noise from the input image and producing a uniform image of the writing with less variation of the same character or word across different writers. Feature extraction routine inputs a normalized image and applies some image transformation techniques in order to produce a sequence of floating point values called a feature vector. The LVQ manipulates a set of input vectors in its attempt to convert an unknown word image into a corresponding word class.

2 Data Acquisition

The data we used in our experiments were obtained from scanned images of 100 pages of cursive handwritten texts from the "Queen's Cabinet" (*Kabinet der Koningin*) at the Dutch National Archive (*Nationaal Archief*) in The Hague (The Netherlands). A sample page is appended to this paper (See Appendix). A total of 14,455 word images were labelled manually in a concerted effort that involved fellow student researchers. Rudimentary software tools developed by the lead researchers (course instructors) were used to aid the manual annotation of the word images. There were 2,442 unique words (classes). In this section, we provide a step by step description of how our fine-tuned experimental data was obtained from these raw data elements. Firstly, however, we give a brief overview of the underlying inspiration from the psychology of human reading.

2.1 Holistic Approach

The three paradigms commonly used to handwriting recognition are holistic, segmentation-based (over segment and merge) and segmentation-free (Hidden Markov Models) techniques. Some of the approaches try to provide a solution to the

problem of handwriting recognition by seeking answers to the questions: where are the words in a sentence? Where are the letters in a word? This is the pervasive problem of 'segmentation'. Our goal in this study is to make a recognizer for separate words. We want to be inspired by the psychological results on how humans read. We take the human brain to be a gold standard for modelling word recognition systems. Experiments have shown that humans use features of word shape such as length, ascenders, and descenders in reading [2]. Thusly, in our preferred approach for creating a handwritten word recognizer, a word is treated as a single, indivisible entity and an attempt is made to recognize words from their overall shape, as opposed to their character contents. A whole word is stretched to match known words, that is, recognition is globally performed on the whole representation of words and there is no attempt to identify characters individually. This is the so-called *holistic (or word-based)* approach. Using this method, we avoid the difficult problem of isolating words into individual characters. The pre-processing and feature extraction modules described in this section work holistically.

2.2 Preprocessing

An important step towards building an automatic script recognizer is to reduce noise in the input images (Figure 2) and to minimize variations in the handwritten texts. These preprocessing steps include the standard operations like noise filtering, binarization, thinning, skew correction, slant correction, estimation of baseline and main writing zones, horizontal and vertical scaling, and additional problem-dependent methods to separate handwriting from background. In this part of the report, we describe the details of these operations as performed in our experiments.



Fig 2: an example of original noisy input image

2.2.1 Removing background

This is done by removing pink page lines from image by replacing all pixels for which the color is close to a certain color value (Fig. 2) (previously sampled) with white pixels (Fig 3).



Figure 3

2.2.2 Conversion to binary image

This removes the background by removing all pixels below certain intensity (threshold). This because it is easier and faster to operate on binary

images instead of color and gray images. Figure 4 shows the result of binarization of the image in Figure 3.

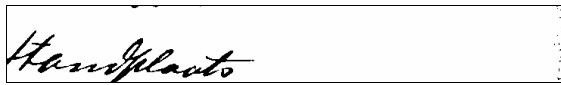


Fig 4: binarized image

2.2.3 Removing slant

This is achieved by applying a shear transformation (Fig 5). Why we chose to use point X should be quite self-explanatory.

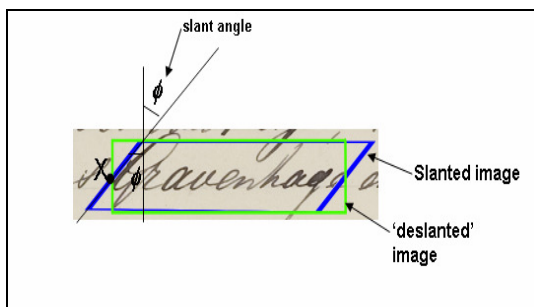


Fig 5. Shearing transform

In our experiments, we assumed a dominant slant (constant shear) angle to be $\pi / 4$ radians. Fig 6 shows the result of applying the transformation (followed by cropping) on the image shown in Fig 4. The writing's slant is transformed into an upright position.

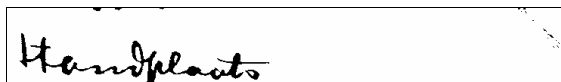


Fig 6: 'deslanted' image

2.2.4 Removing unwanted elements

Determine corpus lines using pixel density histogram

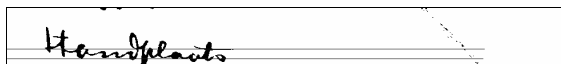


Fig. 7

Clean the image by removing disconnected components above and below the word. From left to right, following the contour of the image while trying to stay close to the upper/lower corpus line and removing any pixels above/below the current position (Figure 6)

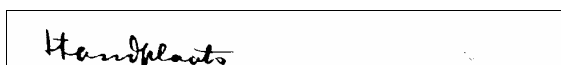


Fig. 7

2.2.5 Removing white space

The white space is removed by looking for the largest non-white space in the image and returning the top, left, right, bottom coordinates of the actual word image (Figure 8). Note that the word may contain whitespace (e.g. between two letters of a word) up to a magical value.

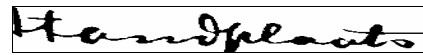


Fig. 8

2.2 Feature extraction

The input to LVQ is a sequence of numerical feature vectors. We therefore extract a sequence of important attribute values or features from the image. As it was explained in section 2.1, our module for feature extraction operates on a higher level of abstraction, extracting features from words rather than partitioning the word and attempting to recognize each part of it.

2.2.1 Obtaining upper and lower contour (using horizontal density histograms of scaled image)

For this purpose, we employ the histogram of pixel densities. Essentially, a histogram of an image refers to a histogram of the pixel intensity values. The histogram is a graph showing the number of pixels in an image at each different intensity value found in that image (Fig 9). There are 256 different possible intensities for an 8-bit gray-scale image. Therefore, the histogram will graphically display 256 numbers showing the distribution of pixels amongst those greyscale values. Histograms can also be taken of colour images – either individual histograms of red, green and blue channels can be taken, or a 3-D histogram can be produced, with the three axes representing the red, blue and green channels, and brightness at each point representing the pixel count. The exact output from the operation depends upon the implementation –it may simply be a picture of the required histogram in a suitable image format, or it may be a data file of some sort representing the histogram statistics. In our case, the output will be an array of numeric features (feature vectors).

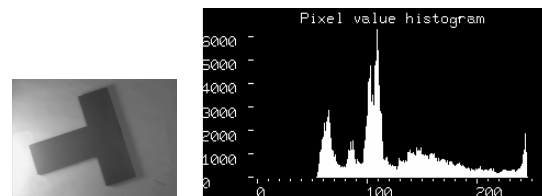


Fig 9: (a) an image and (b) corresponding histogram

2.2.2 Other features

Other features that were obtained include the number of horizontal stripes above/below

top/bottom corpus line and image width. Enhancements are possible.

3 Recognition of a novel instance

The ultimate goal of this quest is to build a recognizer that can automatically recognize (correctly) an unseen word with a relatively high certainty. The novel instance to be recognized is typically input to the recognition system in a scanned image parallelogram format. The image parallelogram that possibly contains noisy cursive handwriting is subjected to relevant preprocessing steps and feature extraction routine. The features extracted, and the procedure used, are just the same as those explained in the previous section. Consequently, we will then need a classifier that will base its recognition ability on the learned features of the labeled instances in the set that has been designated as the training set. In this section, we describe the non trivial task of creating a module that will automatically tell us what word a given unseen feature vector of a word image represents. Essentially, this problem is a classic example of a multiway classification problem.

3.1 Construction of a classifier

There exist several techniques for constructing classifiers. Popular classification schemes include the nearest neighbor algorithms, Bayes classifier, neural networks, SVM, among others. In our work, we employed a slightly more robust (yet quite intuitive) form of these algorithms. The term 'robust' is used in this sense to mean that the algorithm falls under a family of algorithms that can always be enhanced to classify multi-class data in a potentially high dimensional space. The performance of this algorithm has been shown to be generally higher than the basic ones cited above [4]. We introduce the basic notions of learning vector quantization (LVQ) in this section.

3.1.2 Learning Vector Quantization

The strategy used in LVQ learning algorithms is that each class in a given dataset can be represented by (some carefully generated) feature vector. This feature vector is usually determined by some heuristic criterion. We refer to this special representative or reference feature vector as the prototype (sometimes known as called prototype vector or codebook vector). Depending on some factors, such as the amount of training data available, there can be several prototypes for every class in the data. The quantized codebook values are then used for (nearest prototype) classification. A similarity measure, Euclidean distance, for example is used.

The basic algorithm assumes that an apriori known set of the reference vectors $W = \{w^i \mid i = 1, 2, \dots, c\}$ is available. A set of

labelled samples $D = \{\xi^i \mid i = 1, 2, \dots, P\}$ is used for further refinement of W .

At each iteration, indexed by t , the w^i are updated by D by using the following strategy:

1. For each element in D , that is ξ^μ , find the w^i that is closest to ξ^μ . Denote this vector by w^* .

2. Update $w^*(t)$ to form $w^*(t+1)$ by using ξ^μ , as follows:

$$w^*(t+1) = w^*(t) + \eta(t)[\xi^\mu - w^*(t)]$$

(a) IF ξ^μ is correctly classified, that is, it is labelled with the class corresponding to w^* ,

THEN continue with the next element of D ,
ELSE (ξ^μ is incorrectly classified), so update $w^*(t)$ as follows:

$$w^*(t+1) = w^*(t) - \eta(t)[\xi^\mu - w^*(t)]$$

(b) $w^i(t+1) = w^i(t)$ for $i \neq *$

The term $\eta(t)$ is an iteration-dependent parameter used to control convergence. For stability, $0 < \eta(t) < 1$, and $\eta(t)$ is constrained to decrease monotonically with t . In this basic strategy, correct classifications (or quantization) lead to a refinement of w^i in a direction toward ξ^μ , whereas incorrect quantization moves w^i in the opposite direction. The ξ^μ not close to w^i are not changed.

3.1.2 Determining number of classes

We assume that the training set contains examples for all the classes of the instances that will be encountered by the recognizer. The number of classes is therefore taken to be the number of distinct labels in the training set.

3.1.3 Generation of Initial prototypes

Instead of setting an initial prototype to be a single example for a given class chosen at random from the training data pool, we take the centroid of several random instances for the class in question to be our initial prototype for that class. The advantage of this method is that the initial prototypes are placed in positions that are already close to the desired final positions in the instance space. Good initial placement of prototypes in the instance space means that we don't need to use

large steps for the prototypes to reach their respective final positions. Large step size could mean that the prototype moves past (i.e. overstep) the solution or too far away in the wrong direction. In contrast, very small steps may go in the correct direction, but they also require a large number of iterations. This may be considered a stability/convergence issue in LVQs. Classifier sensitivity to initial prototype values is also minimized.

3.1.4 Notation

More formally, we denote a multi-class (n-class) training dataset D with:

$$D = \{ \xi^\mu \in \mathcal{R}^N, \sigma^\mu \in \{ \omega_1, \omega_2, \dots, \omega_n \} \}_{\mu=1}^{\mu=P}$$

Interpret this notation as “the training set D consists of a set ξ of P features, with corresponding class labels in the set σ , where the class memberships are $\omega_1, \omega_2, \dots, \omega_n$ ”

Similarly, denote a set of M prototypes by:

$$W = \{ w^i \in \mathcal{R}^N, \phi^i \in \{ \omega_1, \omega_2, \dots, \omega_n \} \}_{i=1}^i=M$$

The prototypes sit in the same space as the training instances.

3.1.5 Training the prototypes

Adapting prototypes is based on the idea that the initial prototypes can hopefully be relocated to some position in the instance space so as to improve the generalization ability of the initial prototypes. This is done in time steps. The direction in which a particular prototype is moved in the instance space depends on whether the learner (prototype) has made a correct guess (or not) about the class label of the example presented to it. At every time step, selection of a training example, and subsequently the prototype to be adapted follow some defined criterion.

The prototype is then updated according to the rule:

$$w^* = w^* \pm \eta(t)(\xi^\mu - w^*)$$

Where: w^* is the prototype that is closest to example ξ^μ at time step $t + 1$. We say that w^* is a correct winner if its class label coincides with that of ξ^μ , otherwise w^* is a wrong winner. If w^* is a correct winner, then it is attracted towards ξ^μ (for the classifier is correct currently). In this case, the operator \pm is "+". If w^* is wrong

winner, then it is repelled (the sign \pm becomes "-"). The step size is determined by the learning rate function.

3.1.5.1 Selection of training example ξ^μ at time step t

The training data may be presented to the learner sequentially. Alternatively, an example to be presented to the learner could also be selected at random. We note, however, that although these strategies work relatively well, it would become more interesting if the learner were to participate in selecting the training example during the course of learning. We usually refer to this kind of query-based learning as active learning. In active learning, a learner ceases to be a mere passive recipient of information; rather, the LVQ network takes part in selecting the training data instance that it considers to be most informative at that time step. To clearly see what active learning entails, consider a simple two-prototype scenario illustrated below:

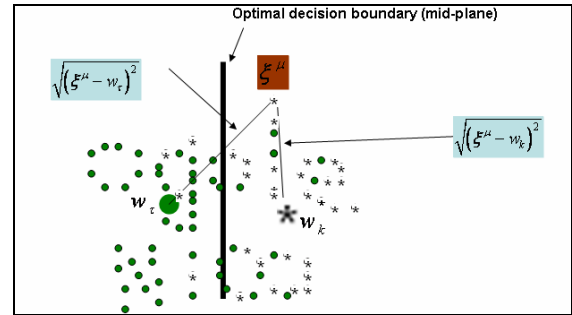


Fig 10: LVQ active learning for a simple 2 prototype situation

If w_k and w_τ denote the closest correct and closest wrong prototypes from example ξ^μ at step t , then we note that the most interesting example at this step is one with:

$$\min_{\mu} \left| \left(\xi^\mu - w_k \right)^2 - \left(\xi^\mu - w_\tau \right)^2 \right|$$

3.1.5.3 The learning rate

The ‘step-size’ discussed in sub-section 3.1.2 is the so-called learning rate. In our experiments, we used a slowly degenerating function as the learning rate function. The initial value is set to 0.3.

The learning rate is given by the function:

$$\eta(t) = \begin{cases} \eta_0 & \text{for } t < t_0 \\ \frac{\eta_0}{(1 + b(t - t_0))} & \text{for } t \geq t_0 \end{cases}$$

where t is a variable standing for the number of training steps, $\eta_0 = 0.3$, t =(general) time step, t_0 =number of steps to wait before decreasing the learning rate, b =speed at which the learning rate decays (higher = faster). In our experiments, b is set to the value 0.1.

The prototype is adapted by translating it by a factor $\eta(t)$ towards (or away from) the stimulus vector at a given training step. At t^{th} step, η is smaller in value compared to the learning rate value used in the $(t - 1)^{th}$ step. For the active learning algorithm, the slowly decreasing values of η even makes more sense (intuitively) since the learner's level of uncertainty reduces with increasing training steps. Theoretically, the learner approaches zero confusion with sufficient training and thus need not adapt much, if at all, to stimuli (as $t \rightarrow \infty$).

3.1.5.4 Stopping criterion

Preliminary experiments had indicated that the error on test data stabilizes after some number of training steps/ cycles. This simple guideline was used to stop the training process.

5 Experiments and Results

In this section, we describe two kinds of experiments. In the first set of experiments, we investigated the behavior of our classifier using sub samples of the data. The second set of experiments was aimed at determining recognition ability of our recognizer.

5.1 Classifier behavior

Investigating the behavior of the classifier in the course of training was important in visualizing the structure of the data and for testing correctness of the implementation. Consequently, we had to adjust some learning parameters in order to produce a better classifier. These parameters include the learning rate, the number of instances whose centroid is used as an initial prototype, the number of prototypes per class, etc.

5.1.1 LVQ and number of training steps

The learning curve (Fig 11) shows the variation of generalization error with number of iterations for *one training cycle* for some arbitrarily chosen number of prototypes and learning rate values. Note that, in general, the error rate decreases with increasing training steps. The fluctuations may be attributed to prototype initialization.

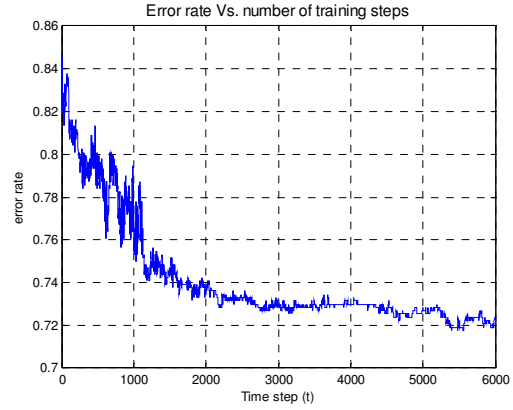


Fig 11: Variation of generalization error with training step

5.1.2 Sensitivity of classifier performance to learning rate parameter values

The object of this particular simulation was to visualize the effect of the values of learning rate on classifier performance. The trial-and-error approach helped us in determining the (roughly) appropriate values of η_0 , b , and t_0 used for training

our *final* recognizer. (Due to the algorithm's high CPU-time demand, we could afford to carry out only a few training steps for illustrative purpose). The graph in Figure 12 shows the variation of error rates with training time for three different initial values of η_0 . Observe that the (significantly) varying error rate values is an indication that using very high, medium or very low step sizes will relocate the prototypes differently (in the instance space). The variations in the overall effect of learning rate on the performance of the classifier would be even bigger when different sets of values of η_0 , b , and t_0 are used.

It is therefore quite difficult to find appropriate values of $\eta(t)$.

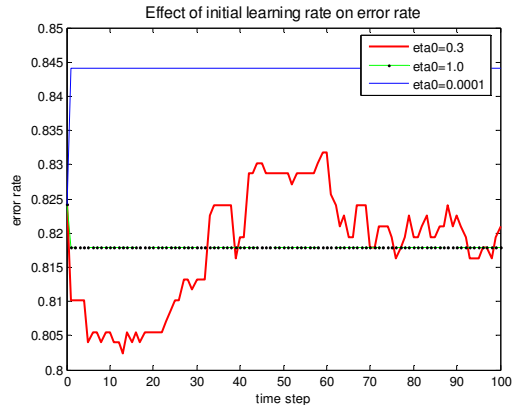


Fig 12: Effect of initial learning rate on classifier performance

5.1.3 Other interesting properties of the classifier

It is true that the generalization ability of any classifier depends on the amount of data used to train the system. We did not try to carry out this experiment ourselves but had to rely on the results given in literature. This is because that particular experiment is very expensive in CPU-time wise. The effect of active selection of training examples on network performance would also be an experiment in the same category.

5.2 Experiments with the recognizer

We could go through various steps, such as the 'lxj-notation' in order to (ultimately) recognize an unknown word. In this notation, "l" means a letter with an ascender (for example, 'k' is encoded as 'l'), "x" means a letter without ascender or descender (for example, 'a' is encoded as 'x') and "j" means a letter with a descender (for example, 'g' is encoded as 'j'). This notation is especially suited for segmentation based recognition technique.

Holistic recognizer only needs to map the entire word features to its labels. In this subsection, we report on the results of our recognizer.

5.2.1 Recognition ability

Our experiments were not fully exhaustive due to the time constraints. While we have demonstrated in the preceding section that training of the classifier can greatly improve upon generalization ability of the classifier, we could not carry out sufficient training on all the training set. The fact that this is a multi-class problem (as opposed to binary classification) means that a lot more time is required for adapting the prototypes. However, based on the small sample of data used, the results are promising. (See the test results and high-score values)

6 Issues for Discussion

Our study only sets yet another pattern in handwriting recognition using slightly different techniques than those that had hitherto been used in this research area. Despite the considerable success of our project, we need to point out some of the major limitations of the methods that we employed.

6.1 Holistic paradigm and Limitations

The major drawback of holistic method is that they are related to a prescribed lexicon of words (units by which the objects of recognition are compared.) Quantization of new words whose feature descriptions were not included in the training set requires fresh training of the recognizer using examples of such a new instance. This problem would not occur if the method had to rely on individual characters for recognition. We explained elsewhere that word segmentation approach has its

own problems too. We therefore remain content that the holistic approach of mimicking the way humans perceive text is a reasonable approach as it is even sometimes more robust in handling deformation within words, a common occurrence in cursive script. There are quite a number of advantages of the word-based paradigm. A holistic approach has the power to model features or effects that are unique to a given class by totally ignoring the segmentation issues and treating each distinct word as a new class. The changes in the appearance of the characters (or *co-articulation effects*) are mediated holistically by the shapes of its neighboring characters.

A word-based paradigm still works quite well in the scenario where the characters are so poorly written that segmentation would fail to separate them into individual entities. This is due to the orthogonality in holistic features. Our unanswered question therefore centers on how the holistic method can be extended to support the nearly infinite variety of inputs or lexicon of words.

Feature selection may also be of interest to us. While we tried within limits of available resources (knowledge, time and development tools) to extract word features that are as informative as possible, we cannot be so sure whether we exhaustively extracted all relevant features. It could also emerge that there exist a number of (extra) irrelevant attributes in our feature vectors. Such an occurrence, known as *curse of dimensionality*, will certainly compromise recognition ability of any recognition device.

6.2 Classifier Limitations

In general, our classifier network seems to be performing relatively well. It is important to note that this is only a basic variant of the numerous algorithms that fall under the family of LVQs. More interesting results could be obtained through further study with the objective of improving generalization ability of the classifier. For this purpose, we may want to explore more recent LVQ variants such as Generalized LVQ (GLVQ), Generalized Relevance LVQ (GRLVQ), Generalized Matrix Relevance LVQ (GMRLVQ), Local Generalized Matrix Relevance LVQ (LGMRLVQ), and so on.

The distance measures are also an area to be investigated. In practice, it is quite unlikely that prior information about the underlying structure of data is known. This presents a problem of choosing the right distance metric for solving a given metric-based problem. Euclidean metrics are frequently employed without further justification, or different measures are compared in a trial-and-error approach. This strategy of metric selection may not be close to an ideal solution method. An appealing solution to this problem could be the use of adaptive metrics. Learning a Euclidean metric, for

example, could reduce to the problem of finding an appropriate transformation

$$D(\xi_i, \xi_j) = (\xi_i - \xi_j)^2 \rightarrow D(\xi_i, \xi_j) = \lambda(t)(\xi_i - \xi_j)^2$$

where: $D(\xi_i, \xi_j)$ is the metric function applied to some input vectors $\xi_i, \xi_j \in \mathfrak{R}^N$, and $\lambda(t)$ is a time-dependent relevance factor whose value is to be determined by some well defined criterion. The determination of this criterion is not straightforward. It follows from intuition that a major parameter to the criterion function $\lambda(t)$ is the *context* of the data input to the learner at time step t . The term “context” is an abstraction for all ideal parameters to the function. These may include data points and class memberships relevant for determining the right instantaneous metric, for example, points within the vicinity of (or far away from) a reference stimulus vector. An objective function (or cost function, to be precise) may be derived from the most relevant arguments and solved in order to deduce the optimal distance value. Solution to intermediate problems could involve semi-definite optimization in the case where the criterion function turns out to be a quadratic optimization problem.

It is also important to investigate the learning rate function. Much as we used a trial-and-error method guided by heuristics to determine a suitable function and initial values of associated parameters, we cannot posit that this is the most appropriate learning rate function. Some slight adjustments made on the learning rate values during our experiments had indicated sensitivity of the experimental results with respect to this very important learning parameter. It’s likely that better (alternative) learning rate functions exist.

It’s well known that which classifier is best depends on many factors, for example, available training set, number of free parameters, time and memory constraints, etc. Given the flexibility of LVQs, we can only agree that prototype-based learning are generally more appealing than its counterparts such as the support vector machines (SVM), neural networks, or classifiers based on naïve Bayesian frameworks.

7 Summary and Conclusion

This objective of this study was to try to build a word recognizer by combining some improved or novel image manipulation and classification techniques from the sub-disciplines of image processing and pattern recognition. Much inspiration was drawn from other disciplines such as psychology and cognitive science. The problems of automation (machine intelligence) and emulation of human intelligence, of course, generally fall

under the fields of computer science and artificial intelligence. Development of an effective solution to automatic script recognition will likely involve multi-disciplinary approach.

Our main approaches of obtaining features from word images followed by classification using LVQ networks seem to be promising. However, we need to pay attention to some of the issues that have been raised in the discussion section of this paper. For now, the system we have is only a *prototype*. (The term “prototype” is used in this case to mean a “working model” in the general engineering sense – not from the LVQ perspective). Construction of a real world word recognition system would, fortunately, follow the same approach but with more enhancements.

8 References

- [1] Schomaker, L. & Segers, E. (1999). “*Finding features used in the human reading of cursive handwriting*”, International Journal on Document Analysis and Recognition, 2, 13-18.
- [2] Schomaker, L.R.B. (1994). *User-interface aspects in recognizing connected-cursive handwriting*. Proceedings of the IEEE Colloquium on Handwriting and Pen-based input, 11 March, London: The Institution of Electrical Engineers, Digest Number 1994/065, (ISSN 0963-3308).
- [3] Vuurpijl, L. & Schomaker, L. (1997). *Finding structure in diversity: A hierarchical clustering method for the categorization of allographs in handwriting*, Proceedings of the Fourth International Conference on Document Analysis and Recognition, Piscataway (NJ): IEEE Computer Society, p. 387-393. ISBN 981-02-3084-2
- [4] A. Senior and A. Robinson. An off-line cursive handwriting recognition system. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):309–321, March 1998.
- [5] A. Vinciarelli. A survey on off-line cursive script recognition. *Pattern Recognition*, 35(7):1433–1446, 2003.
- [6] C. Suen, C. Nadal, R. Legault, T.Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Proc. of the IEEE*, 80:1162–1180, 1992.
- [7] Multiple Classifier Systems in Offline Cursive Handwriting Recognition Inauguraldissertation der Philosophisch-naturwissenschaftlichen Fakultät der Universität at Switzerland, 2004.
- [8] M. Biehl, A. Ghosh, B. Hammer, “Dynamics and generalization ability of LVQ algorithms”, University of Groningen, Chausthal University of Technology
- [9] C. Ogole, “LVQ Active Learning for Boar Sperm Head Image Classification”, University of Groningen
- [10] R. Schakoff “Pattern Recognition”
- [11] R. Duda, P. Hart, D. Stock, “Pattern Classification”

Appendix

Display of an image scan of a sample of the 100 pages that were manually annotated for use in our set of experiments.

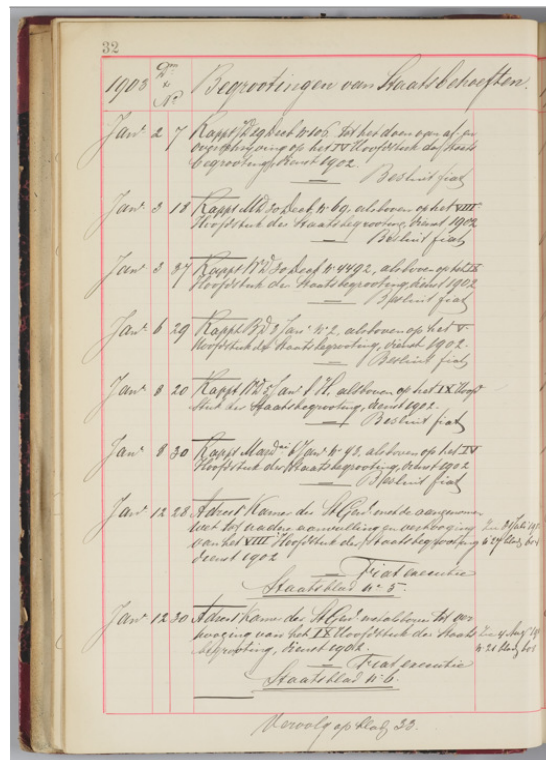


Figure 12: An example scanned page (image) of handwritten texts from the "Queen's Cabinet" (*Kabinet der Koningin*) at the Dutch National Archive (*Nationaal Archief*) in The Hague (The Netherlands).

Acknowledgements

This paper is a report for the research in "Handwriting Recognition", a 5- EC course unit offered at the Department of Artificial Intelligence, University of Groningen.

I must mention that this research work (brainstorming and lab experiments) was carried out in partnership with a fellow student researcher, **Mr. Nico van Benthem**. His contribution was enormous and indispensable. (He too had to prepare and submit his own report – a requirement for the course).

I am grateful to the student assistant, Drs. Axel Brink, for his friendly yet highly professional guidance throughout the course. Particularly, his statement "this is an unsolved problem" that saw every research couple come up with different but all plausible approaches to solve the problem at hand is such a mild but very workable introduction to the research world. The methods that Nico and I chose took us to the 'direction' that you have just (hopefully) read in this paper.

I should also mention that much of the work on the classifier networks employed in this project is a brainchild of the Intelligent Systems Research Group on LVQ of my home Department of Computing Science

Last but not least, I would like to thank Prof. dr. Lambert Schomacker, the course instructor for the introductory and very inspiring lectures on the Course "Handwriting Recognition"