Handré: An Experimental Approach to Intelligent Character/Word Recognition using Support Vector Machines & Dynamic Time Warping

Sang Woo Jun   Chong-U Lim
wjun@mit.edu   culim@mit.edu

December 6, 2011

Abstract

In this report, we describe Handré, a full-system implementation of an experimental approach to intelligent character/word recognition using support vector machines and dynamic time warping. Handré is unique in that it attempts to use various computer font data instead of actual handwriting to approximate handwriting recognition. Our system implements three large subsystems. (1) Image segmentation using a time series edge detection method, (2) character recognition using both support vector machines and K-means clustering using dynamic time warping, and (3) word recognition using spelling correction. Our system was tested on professor Leslie Kaelbling's handwritten lecture notes. Even though out system includes various novel and experimental ideas, and suboptimally implemented in many areas due to lack of time and experience, our results showed resonable accuracy and promise.

Contents

1 Introduction

Optical Character Recognition (OCR) is the term used to describe the process of recognizing scanned images of text from documents, which may be handwritten manuscripts or printed text. An extension to OCR is Intelligent Character Recognition (ICR), which involves additional processing and recognition techniques in order to improve the accuracy of translating such documents by performing recognition on the level of words as opposed to individual characters. In this paper, we present an experimental system to outline the process of performing recognition of an entire handwritten document. Our approach outlines the feasibility of performing word recognition without having to collect hand-written samples for training. We make use of a windowed time-series edge detection method and pixel analysis to perform segmentation of the document invidiual words and characters respectively. To perform individual character recognition, we used 2 different approaches – a kernel-based support vector machine (SVM) classifier and a K-means clustering method using dynamic time-warping (DTW) which were both trained using a database of TrueType fonts. We then perform a committee based process of combining results from both models together with a probablistic spelling checker in order to perform the word recognition.

1.1 Emphasis

We emphasize the following multiple of Handré unique characteristics.

Feasibility of Font Training Data We conduct exploratory work on using default computer fonts such as Verdana, Comic Sans MS, and handwriting-like fonts such as angelina and harrison to train our learning methods, instead of collecting a large corpus of handwritten characters for training. Results show that this is a feasible method

Multiple Character Predictors We use both SVM and DTW methods to predict characters, and merge them through a committee based process. Because SVM is more accurate with more training data and DTW is more accurate when there is less training data, our implementation works at a reasonable accuracy at various amounts of training data.

Full-System Approach Instead of implementing a subset of a complete working system, we attempted to implement the entire system stack, up to the point where it was actually usable. Because there were multiple cross-checking functionality such as the committee based merging and spelling correction, we could obtain a certain level of utility even when some of the individual functionality was incomplete.

Feature-dtw The K-means clustering method goes through multiple, each a tree in a forest of less-accurate classifiers. The first classifier is a novel variant of multi-dimendional dynamic time warping named feature-dtw, which generates two 'feature' time series by scanning the pixel layout according to the dimensions, and adding the distance between the two pairs. This method almost always assures that the correct character is included in the top ten or so of possible characters, speeding up all subsequent classifiers by multiple orders of magnitude by reducing the problem space.

2 Segmentation

2.1 Word Segmentation

2.1.1 Time-series Segmentation

A real-time edge detection method for times series was used to extract individual word images from a scanned image of a document. The image is first scanned vertically, constructing a time series consisting of the number of black pixels per vertical position. This information is analyzed to separate each lines into individual images. Each segmented line image is then scanned horizontally, constructing a time series. This is analyzed to segment individual words.

In order to accurately segment lines and words, we adapted a time series edge detection method proposed to segment heartbeats to identify separating spaces between lines and words. This is done by first differentiating the time series and then doing a moving window integration, to discover the transition point separating the dense and sparsely colored regions. This generally resulted in more accurate results than simple pixel density analysis employed by most work.

Figure 2.1.1 shows the plot of a processed time series derived from pixel density of a single line image and its resulting word segmentation. This image was taken from a screenshot of one of professor Leslie Kaelbling's lecture notes.

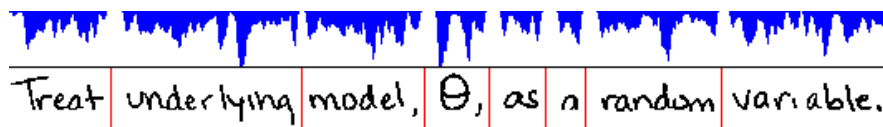Figure 1: Word segmentation of an example line.

2.1.2 Post-processing

Because scanned handwritten documents contain many types of noise information such as underlines and shadows, a certain amount of post-processing on the segmented word image

to obtain relatively clean character data. First, single pixel noise was eliminated by removing pixel 'islands', that existed by itself with only one or no neighboring pixels. Second, underlines were removed by picking the top one or two horizontal lines with a very high pixel density and removing each pixel from it, unless it had pixels above and below it. Third, strokes spilled over from lines above one in interest was removed, by scanning all pixels on the first row of the image and recursively filling all adjacent black pixels with white.

2.2   Character Segmentation

Because individual characters in a handwritten word often have overlapping widths or just plainly connected, the same method of using pixel density cannot be used to segment characters in a word. Furthermore, it is provably impossible to perfectly segment characters in a word without symantic context. For example, it is just as likely for 'm' to be segmented into 'm', as it is to be segmented into 'rn' or any other subsection of 'm' that may not exist in the english alphabet.

However, because this research is a preliminary feasibility search that does not heavily focus on character segmentation, we employed some preprocessing by hand and simple pixel analysis to perfectly segment characters in a word. First, word images were preprocessed by hand, by deleting some pixels so that no characters are touching each other. Then, simple pixel analysis was used to exploit this feature and correctly segment characters. This method was useful because it was much more productive than hand segmenting every character manually.

3   Recognition

In this section, we outline the techniques which we focused on in order to address recoginition task of our project. In Section 3.1, we describe desribe support vector machines (SVM) and how they can be used for classification of the different alpha-numerical characer by training on a dataset in order to learn a model which generalizes well for all characters.. Section 3.2 covers dynamic time warping, and describes our approach to using it to classify alpha-numerical characters by minimising the distance of difference between the pixel-representations of the different characters. Section 3.4 explains how a spelling checker is implemented in order to act as a type of regularization of the combined results from the SVM and DTW systems.

3.1   Classification using Support Vector Machines

Support Vector Machines (SVMs) are a form of supervized learning methods which can be used for classificaiton or regression problems. In a binary classification example, we would train the SVM on a labelled dataset and if they they are linearly separable, the SVM will find a unique separation boundary in the form of a hyperplane with points falling on each side

having different classifications. The separation boundary would be one in which the margin is maximised.

In general, not all data points will be linearly separable often as a result of overlapping class-conditional probabilities. Also, there is a chance of overfitting on the training data points which might negatively affect the generalizability of the classifier for future points. As such, we adopt the use of *slack variables* which results in a 'soft margin', in which we allow some data points to be incorrectly misclassified with a certain penalty with the aim of overcome overfitting. The general formulation of SVMs as constained quadratic programming problem is as follows

$$\underset{\theta}{\text{minimize}} \quad C \sum_{i=0}^{n} \xi_i + \frac{1}{2} \parallel \theta \parallel^2$$
$$\text{subject to} \quad y_i(\theta \cdot \mathbf{x}_n + \theta_0) \geq 1 - \xi_i, \ i = 1, \ldots, m$$

where x_i represents each training data point, with y_i being its corresponding target classification. θ is the model, or parameter, of the classifier with offset θ_0, while C and ξ represent the penalty and slack variables respectively.

Supporting Multiple Classes   In our project, we are interested in the use of SVMs to classify individual characters based the a given vector input of pixel values. Extending SVMs to support multiple target classifications requires us to train multiple binary classifiers, one for each individual target character. Given an input vector, each classifier would give a possible classification, and we then employ **voting** as a way of deciding the best classification result for our data. The common voting strategies to decide on a classification described as follows. In **one-versus-one** voting, the idea is to fit a classifier for each pair of classes, and when it comes to making the prediction, we select the class with the most number of votes. In **one-versus-rest** voting, each character has a classifier which is fit to it. When making the predicition, the class with the highest classification output is chosen in a *winner-takes-all* strategy.

3.2   K-means Clustering using Dynamic Time-Warping

The second method used for character recognition is K-means clustering, using modified dynamic time warping as a distance metric. K-means clustering is an efficient method to use when classifying an input to one of the K possible clusters, which is a good fit to character recognition. Dynamic time warping (DTW) is a dynamic programming method to calculate the similarity between two time series by warping each time series into various ways to discover the warp configuration that makes the two series most similar. The difference between this method and normal euclidian distance can be seen in Figure 3.2 This is also a good fit for character recognition because characters are generally similarly shaped, but it is

stretched or warped in various ways. A window of ranges of warping allowed can be specified to aid in accuracy and speed.
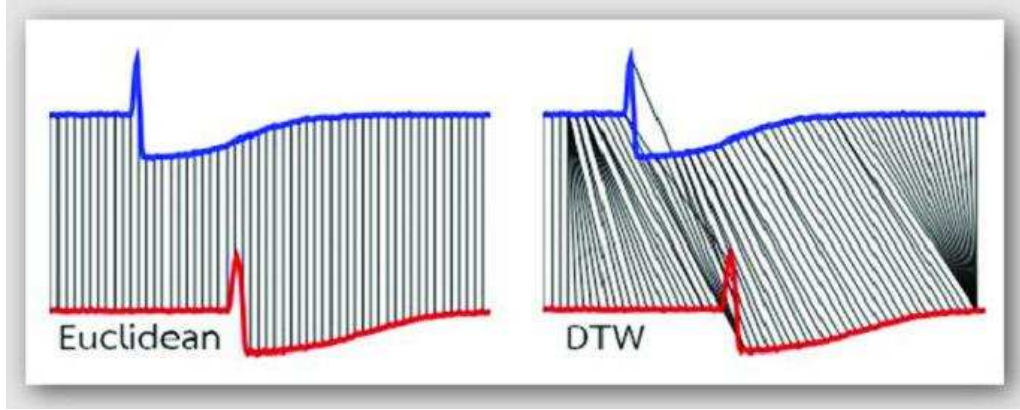

Figure 2: euclidian distance and dynamic time warping

Modified Dynamic Time Warping We were unable to use multi-dimentional dynamic time warping for our goals, which is a common form of modified dynamic time warping used for image and texture analysis. Multi-dimentional dynamic time warping is done by taking the euclidian distance between each possible set of two rows, from both images and using dynamic time warping on it. However, this method could not be used because of two reasons. First, character sizes were often very different, making euclidian distance between rows not a very accurate measure. Second, it took a prohibitively long time.
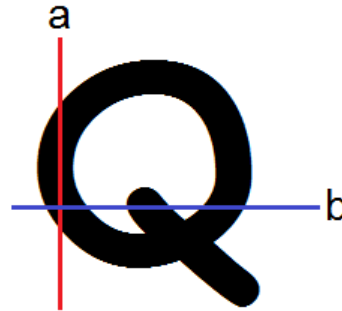
Figure 3: x and y features at position a and b

Therefore, we developed a novel method of character distance metric called feature-dtw, which has not been used in previous works, to the best of our knowledge. This method first generates two feature time series per character image, *xfeature* and *yfeature*, which are time series consisting of the number of pixel lumps that exists on that particular row or column. For example, in Figure 3.2, the item in xfeature at index a would be 1, and the item in yfeature at index b would be 3. This feature series can be expected to have enough

8

information in them, so when dynamic time warping is used on them, they can give a close approximation to the level of difference between two characters.

Because feature-dtw is not completely accurate, it is only used as the first tree in a forest of classifiers, to first prune the problem space by taking the top 10 or so of most likely characters. Feature-dtw can often mistake some different characters to have the same features, depending on the tilt or shape of the characters. So after taking the top 10 or so of most likely characters from feature-dtw, traditional multi-dimentional dynamic time warping is used to actually classify the character.

Modified K-means clustering   Because the level of difference between characters differ by lot, the default method of K-means clustering yielded multiple clusters with different characters in the same cluster, or with the same character split among different clusters. Therefore, we modified the K-means method to fit better into our ICR interface by picking character instances from different computer fonts that approximately separated the characters as much as possible, and using it as a 62-means cluster (26 + 26 upper- and lowercase letters and 10 digits).

A problem arises from this configuration, because as shown above, certain character sets such as q and a are virtually indistinguishable from clustering, and will fall into the same cluster. (Experiments showed that the distribution of 'q' was larger than 'a', causing most 'a's to be classified as a 'q') The natural way to solve this would be to have a fewer number of clusters than characters at this point, and use Hidden Markov Models or some other method to guess the word. However, in order to maintain the same interface between DTW and SVM methods, the 62 means cluster was used without merging clusters, and it is sorted out during spelling correction. This results in several of the 62 clusters actually being interchangeable, and actually same clusters with multiple labels. Therefore, the predicted words may not look like actual words, but can be backtraced to predict the original correct word.

3.3   Combining Approaches for Word Recognition

We can improve the results of performing a classification task by including the results from a group of classifiers, as opposed to a single one. This is often termed as a *committee*. We look to implement a form of committee-based decision by making use of the results from both the SVMs and feature-dtw methods when attempting to perform recognition on a character. In essence, each classificaton system will present a list of possible classification labels for a given input character, and by selectively pruning the sets into smaller subset, we hope to make a better informed decision on the classification. The exact process in which this is performed, along with the selection criteria and merging rules are both covered in Section 4.4.

3.4   Spelling Correction Using Probalistic Dictionary Model

In order to account for the K-means clusters with different labels explained in section 3.2 and increase the accuracy of the final predicted words, we implement a modified version of the spelling corrector covered by Norvig [1]. This implementation of the spelling corrector works by reading a large number of words from a corpus and comparing them with input. Its speed is very fast because it works by generating a small number of possible corrections and matching them against an indexed database of corpus words. The spelling corrector in our system serves two functions.

First, it corrects up to two mis-predicted characters in a predicted word. This is to increase the accuracy of predictions even with limited character recognition accuracy. If there are several possible matches, it first chooses the one with the least correct distance. If there is still a conflict, the one with the most occurrences in the corpus is chosen. Therefore, choosing a domain-specific corpus can improve accuracy. We used an entire body of words from a machine learning textbook as a corpus.

Second, it understands that the 62-means clusters actually consists of much smaller number of clusters and only applied multiple labels and looks for interchangeable spellings that belong to the same cluster. Such multi-clusters include "e,c" "s,t,f" and "i,l". This creates a large number of possible permutation of corrections, but eventually finds the correct word.

4   Experimental Design

In this section, we desribe the experimental design which we undertook in order to apply the segmentation and recognition techniques from Sections 2 and 3. In Section 4.2, we describe the prodedure in which we trained the multi-class SVMs on training data which consisted of different images of each alphabet character for a variety of fonts. Section 4.3 outlines the application of DTW onto a single font set and how it was used to perform its classification. How the results from both experiments were combined together to make committee-based decision is covered in Section 4.4. Finally, Section 4.5 explains the use of the dictionary spelling checker to improve the result and come up with a final decision.

4.1   Training & Testing Data

For the purpose of our project, we tried to locate a resource of training data for alphabets which was commonly available. Several repositories containing data were either unsuitable or was paid-only. More importantly, we wanted to show-case that system could make use of a generic database for its training data, and not have to resort in having a particular target domain to have to first provide us with both a vast number of examples, but also go through the tedious task of hand-labelling them though it given the resources and time, it might

be a possible way to improve the system (Section 6). As such, we decided to using readily available TrueType fonts as our training data.

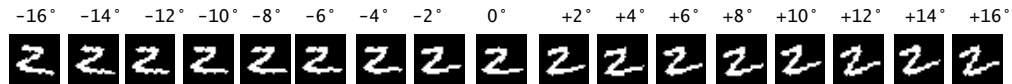### 4.1.1   Using Font Images as Training Sets

Figure 4: 17 different rotations for each alphabet character

We made use of fonts for our training data by randomly picking a set of handwritten fonts which are freely available on the web (see Appendix A.2 for the list and samples.) In order to normalize the characters, we resized each character to fit as a 28x28 white-on-black square image[1]. The character is then set to black-and-white mode, in which the pixel values are either 0 (black) or 255 (white). For each character in the font, we also created additional samples of the character image by rotating the characters by an angle between [-16°, +16°]. Thus, each training sample can be viewed as a 1x784 vector of pixel values. Figure 4 shows an example of a font character which has been normalized and its various rotations.

4.1.2   Testing on Handwritten Text

Figure 5: Segmenting and normalizing a scanned image into its individual characters

Image segments containing words which were obtained from the word-segmentation step (Section 2) were used for tests. The images were then cleaned and individual characters were extracted for the recognition process. We perform similar normalization steps in order to make the test images have the same dimensions and format as our training data – images have their colors inverted, are cropped and resized to 28x28 pixels. Using our optimal performing classifier, we then attempt to classify each individual character in the word of l characters. An example of a test image is shown in Figure 5.

---

[1]We referenced the dimensions used by the MNIST database, and serves as a reference point for common handwritten character sizes.

4.2 Recognizing Alphabet Characters using Multi-Class SVMs

In order to be train the best classifier for our recognition task, we varied the parameters and kernels used and performed cross-validation in order to select the best performing one. The following list describes the various parameters and choices we used

- Radial Basis Kernel
    - C = [1, 10, 100, 1000]
    - γ = [0.01, 0.001, 0.0001]
- Linear Kernel
    - C = [0.001, 0.01, 0.01, 1, 10, 100, 1000]

```
'R'   [('R', -1.532), ('k', -2.261), ('P', -2.450), ('H', -2.535), ...

'a'   [('a', -2.745), ('q', -2.749), ('Y', -2.846), ('t', -2.998), ...

't'   [('t', -0.476), ('f', -2.595), ('k', -3.050), ('q', -4.072), ...
```

Figure 6: An example of a prediction table after training on the scanned image containing the word 'Rat'

We optimised the parameters based on the precision score and recall score, and the results of which are shown in Section 5.1. Additionally, we made use of both the 1-versus-1 strategy and 1-versus-rest voting schemes to perform the multi-class classification. We also obtain the probability scores of each class (signifying how confident we are of our classification) to obtain a table containing l rows, and each row contains a vector of 52 probabilities – one each for the alphabets (26 upper-case and 26 lower-case characters.) As covered later in Section 4.4, we will make use of this prediction table to make a combined decision, together with the DTW method, in order to improve our word recognition results. An example of a prediction table is shown in Figure 6. In this case, the scanned image contains the word 'Rat', and after classification of each character in the word, we assign a probabilistic score to each the possible classifications for the character. We use the log-probabilities due to some probabilities becoming extremely small. In the image, the rows are sorted from highest probable classification to lowest from left to right.

4.3 K-means Clustering using DTW

After multiple experiments and cross validation, a 62 cluster means were chosen as a mixture of characters from fonts 'Verdana', 'Comic Sans MS' and a handwriting font 'angelina', all

of which are freely available from the web. And among the several possible multi-clusters that have been discovered, the following five has been decided upon as a feasible compromise between accuracy and speed loss that results from the increasing number of possible permutations that arises from introducing more multi-clusters. This information is used by the spelling corrector to guess the possible original word.

- List of Chosen Multi-Clusters

  - C1 = [a, q]
  - C2 = [u, v, y]
  - C3 = [c, e, f, t, r]
  - C4 = [i, l]
  - C5 = [h, n]

For the dynamic time warping method, a window of 8 was used to specify the maximum amount of pixels each pixel can be warped, both for feature DTW and multi-dimensional DTW.

4.4   Combined Character Recognition

With the resulting probability tables from the trained SVMs (Section 4.2) and the DTWs (Section 4.3), we then developed a policy to select the resulting individual characters. First, each row in the probability was ordered from high to low – with the most likely prediction in the front. Next, each character to be classified in the target string, we selected a subset from both the tables as a combined prediction by taking the intersection of the two sets. In the event that no characters are common between the two sets, we picked the most likely members from each set.

This is illustrated in Figure 7. In this example, the scanned image contains the word 'Rat'. The prediction table for the SVM system is shown at the top, while the table for the DTW system is at the bottom. For each character to be guessed, a subset of the top 3 predicted characters are chosen from both tables, so for the first character, we have the set $['R','k','p']$ and $['R','O','E']$ from the SVM system and DTW system respectively. Then, the intersection of the subets are assigned for first character of 'Rat', which is 'R'. For the second character, as there are no intersecting characters, we pick the top two predictions to form the subset $['a','h']$. The third character 't' follows the same discussion as the first character 'R'.

After some experimentation, we emperically decided that selecting subset of 3 characters from each prediction table row was the best policy due to potentially large number of permutations that might result from taking more. In the case where the intersecting set is empty, we picked the top predicted character from each of the tables.
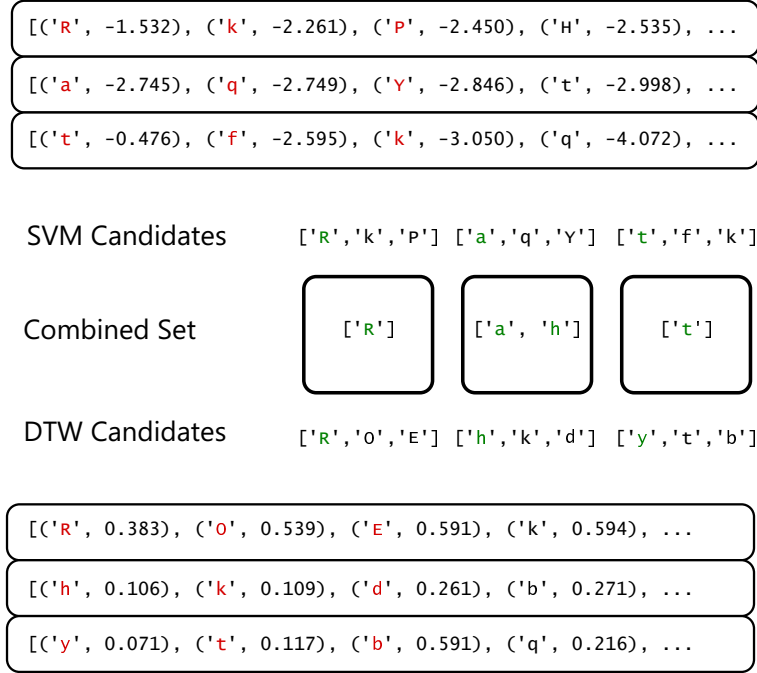
```
[('R', -1.532), ('k', -2.261), ('P', -2.450), ('H', -2.535), ...
[('a', -2.745), ('q', -2.749), ('Y', -2.846), ('t', -2.998), ...
[('t', -0.476), ('f', -2.595), ('k', -3.050), ('q', -4.072), ...
```

SVM Candidates   ['R','k','P'] ['a','q','Y'] ['t','f','k']

Combined Set   ['R']   ['a', 'h']   ['t']

DTW Candidates   ['R','O','E'] ['h','k','d'] ['y','t','b']

```
[('R', 0.383), ('O', 0.539), ('E', 0.591), ('k', 0.594), ...
[('h', 0.106), ('k', 0.109), ('d', 0.261), ('b', 0.271), ...
[('y', 0.071), ('t', 0.117), ('b', 0.591), ('q', 0.216), ...
```

Figure 7: Illustrating how predictions from both systems are combined.

4.5 Word Recognition Using Spelling Correction

From the the result from the combined prediction, we may now have several permutations of guesses of the word which we wish to recognise. In this case, several constraints can be assumed to be true, notably that the length of the word has been correctly identified and that any mis-recognition which might have occured is entirely due to a substitution error in the given word string. As such, we make use of the spelling correction technique described in Section 3.4 to the set of possible guesses for the word.

Following on our example from the previous section, we would have two potential guesses for the word – 'Rat' and 'Rht'. In this case, if the spelling checker runs on the former, it would give a successful result and return, without a need to consider the second case. However, imagine the scenario that the resulting words were instead 'Ret' and 'Rat. In this case, the spelling checker would note that neither words exist, but attempt to propose a possible solution, which would be 'Rat'. This provides an additional measure to contribute some form or prior knowledge into helping our classifiers make a classification, which both initially got wrong, eventually make a correct prediction.

Because 'a' is part of the multi-cluster C1 and 'h' is part of the multi-cluster C5 introduced in section 4.3, the spelling corrector would also check for possible spellings of 'Rqt' and 'Rnt'. However, because the corrector would already have returned with 'Rat' this path will not be taken in this example.

5 Results & Analysis

In this section, we cover the performance of the system as described by the experimental. In Section 5.1, we cover the performance of the SVMs on the trained font images as a result of cross-validation, and pick the optimal parameters for further testing. In order to explain the results in detail, we choose to focus the results and analysis of the performance on a particular handwritten string containing the word – **multivariate** which were obtained from Prof. Leslie Kaelbling's lecture notes. We discuss several other test results based on this and various other experiments in the (Section 5.4).

5.1 Alphabet Character Recognition using SVMs

Trained Parameters From varying the various parameters, we achieved the best results using a linear kernel with C=1, with a prediction rate of around 90% when cross-validation was performed on all the trained font data. Using the radial basis function kernels gave poor results between 3% – 5% recognition rate when varying γ. As such we made use of the linear kernel with penalty $C = 10$ for all further testing.
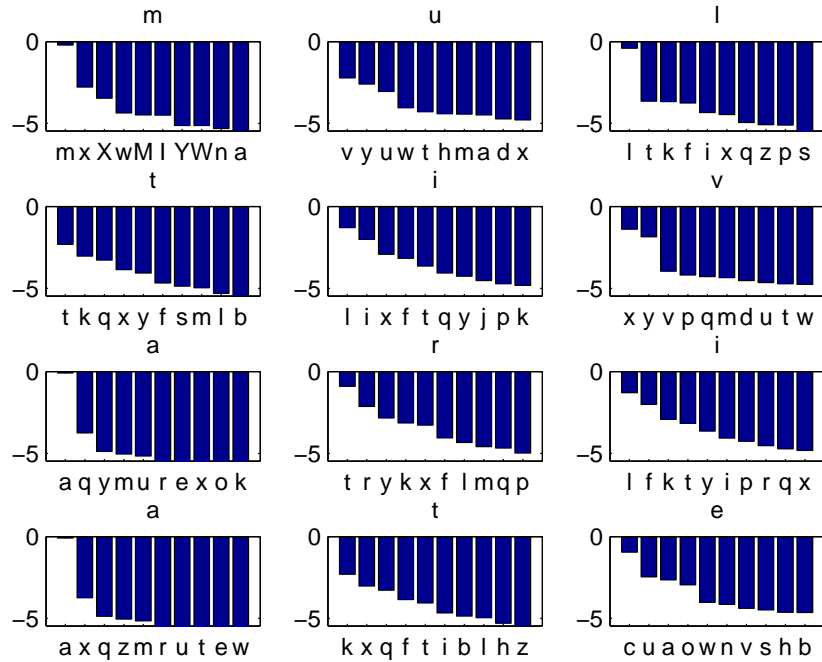

Figure 8: Results of the SVM for the world 'multivariate'. Larger values are better guesses.

 We performed the test on the scanned image of the word '**multivariate**' obtained from Prof. Leslie Kaelbling's notes, and achieved a precision and recall rate of about 40% using

15

the SVMs. The confusion matrix of this test can be found in Table 3 of the Appendix A.1.1 Figure 8 shows the results shows the performance of the trained SVM classifier in recognizing characters in the word 'multivariate'. For each character being guessed, we show the list of the top 10 matches based on their log-probabilities. As such, values which are larger (less negative) are more likely to be correct. For example, for the first guessed character 'm', we have the guess 'm' with almost zero log-probability. Compared to the other next candidates 'x' and 'X', we intepret this as being relatively more confident that the correct guess is 'm'. In the second guess 'u', we can see we are less confident about the whether it should be 'v', 'y' or 'u'. We note that for a majority of guesses, the correct character is usually within the top three candidates based on their log-probabilities. Thus, as we chose this is the reason why we elected to make use of the top 3 candidates for each guess to pass on to the combined predictor.

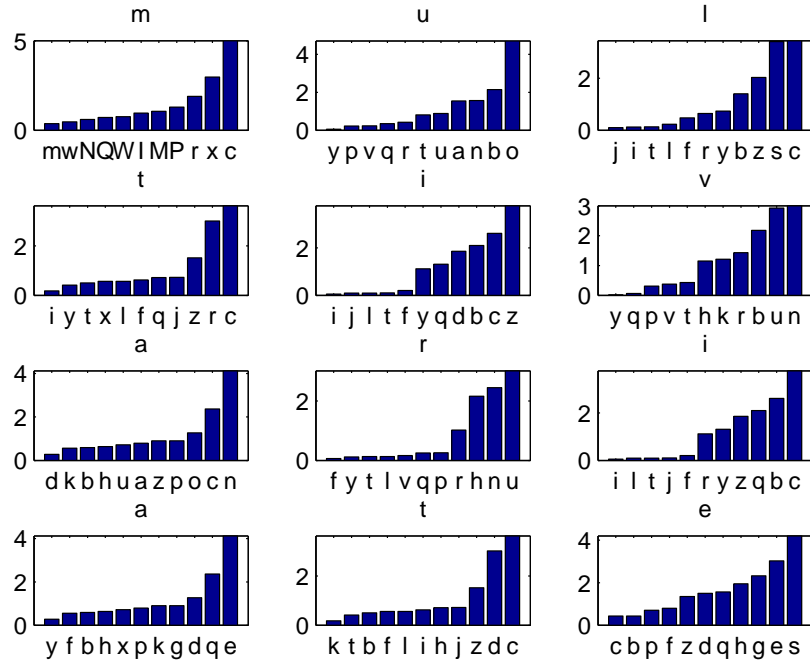## 5.2   K-means Clustering using DTW

Figure 9: Results of the feature-DTW for the world 'multivariate'. Smaller values are better guesses.

The results of the performance of the feature-DTW classifier for the same recognition task is shown in Figure 9. Similar to the SVM classifier, each character of the word we are trying to recognize has a list of potential candidate classificatons. The values indicate a measure of

the relative distance of the candidate to the target guess as compared to other candidates. As such, smaller values indicate a higher confidence of a particular candidate. For instance, in the first character 'm' to be guessed, we are more confident that the correct guess would be 'm'. This is only slightly more than 'w', but much higher than 'c', for example.

Looking at the guesses for the 'i' character in both the 5th and 9th graphs, we note that that certain clusters of characters tend to be closely similar. In this case, 'i', 'j', 'l','t' and perhaps 'f' are closely clustered. Emperically, we decided to adopt a similar selection process to that of the SVM classifier, in which the top three candidates for each character are used for the combined predictions later.

5.3   Combined Recognition & Spelling Correction

SVM Prediction Table		DTW Prediction Table	
m	$['m','x','X']$	m	$['m','w','N']$
u	$['v','y','u']$	u	$['y','p','v']$
l	$['l','t','k']$	l	$['j','i','t']$
t	$['t','k','q']$	t	$['i','y','t']$
i	$['l','i','x']$	i	$['i','j','l']$
v	$['x','y','v']$	v	$['y','q','p']$
a	$['a','q','y']$	a	$['d','k','b']$
r	$['t','r','y']$	r	$['f','y','t']$
i	$['l','f','k']$	i	$['i','l','t']$
a	$['a','x','q']$	a	$['y','f','b']$
t	$['k','x','q']$	t	$['k','t','b']$
e	$['c','u','a']$	e	$['c','b','p']$

Table 1: Prediction Tables from both classification systems

Progress of Spelling Correction		
mvltiyatlakc	$\cdots$	multlvarlake
mvltivatlakc	$\cdots$	mulrlvarlake
mvltiuatlakc	$\cdots$	muirivatlake
myltiyatlakc	$\cdots$	muitivarlake
multiyatlakc	$\cdots$	muirivarlake
mvltiyatlake	$\cdots$	mulrivatiake
mvltlyatlakc	$\cdots$	multivariake
mvitiyatlakc	$\cdots$	multivariate

Table 2: Table showing the spelling correction occuring on 'mvltiyatlakc'

Table 1 shows both prediction tables from the SVM classifier and DTW classifier. Our combining policy involves taking the *intersection* of the sets for every single row. Intersecting gives us $[['m'], ['y', 'v'], ['t'], ['t'], ['i', 'l'], ['y'], ['a', 'd'], ['y', 't'], ['l'], ['a', 'y'], ['k'], ['c']]$. As such we have several candidate word guesses, an example would be "mvttiyaylakc" and another would be "mvltiyatlakc". With the various permutations of characters which form our guesses for the word to be recognized, we perform the spelling correction step to each permutation in order to see if a valid word can be found. This process for the one of the permutations 'mvltiyatlakc' is shown in Table 2, and this particular run took 577 steps to eventually find the word '**multivariate**', which is the result for the word which we were intending to recognize.

5.4 Additional Results & Experiments

Additional Recognized words Talk about the other words like Rather, update, etc. which were recognized here.

Extended Experiments

Figure 10: Computer-generated testing data

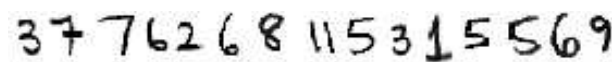**Testing on Image-captured Fonts** completely accurate

Figure 11: Testing data of digits

first 7 and second 3 was each recognized as 1 and 5. Important because spelling correction cannot be applied to digits
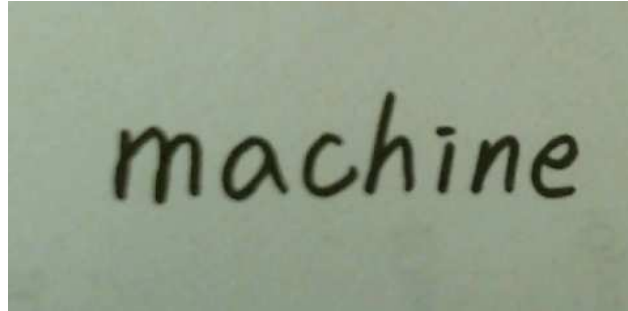
got 'mqchinc' on top
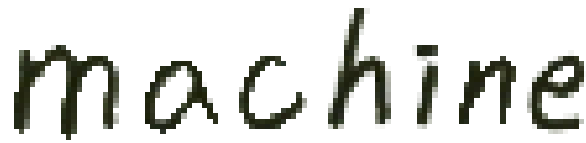
Figure 12: Handwritten text captured using webcam

Figure 13: Preprocessed handwritten text

6 Conclusions & Future Work

6.1 Contributions

Contributed repository of training fonts as well as app to do it. Exhibited Feasibility of using Fonts as training data for SVMs. Showed positive results for combining different predictors. Novel feature-DTW approach for recognition.

6.2 Conclusions

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque posuere molestie metus. Suspendisse tellus urna, porta sit amet rutrum eu, tristique quis urna. Donec varius pharetra purus, eget mollis tortor ornare vel. Nullam sagittis tellus id dui placerat eget congue libero facilisis. Donec mattis sagittis lectus, eget porta quam facilisis vel. Vestibulum non urna ante, nec mattis mauris. Nulla sit amet interdum eros. Nam congue lacinia nulla, vitae aliquet nisl tincidunt vel. Morbi gravida bibendum ipsum, at accumsan nisl suscipit ac. Sed accumsan cursus tortor a faucibus. Phasellus tempus, orci ac lacinia hendrerit, dui justo accumsan mi, congue dapibus massa turpis at lectus. Cras a tellus nisi. Aliquam vitae dolor id nunc lacinia fermentum et sit amet metus. Nullam viverra ante eu mauris ultrices nec adipiscing lectus dapibus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nam mollis commodo lacus, eget bibendum risus lobortis nec.

6.3 Future Work

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque posuere molestie metus. Suspendisse tellus urna, porta sit amet rutrum eu, tristique quis urna. Donec varius pharetra purus, eget mollis tortor ornare vel. Nullam sagittis tellus id dui placerat eget congue libero facilisis. Donec mattis sagittis lectus, eget porta quam facilisis vel. Vestibulum non urna ante, nec mattis mauris. Nulla sit amet interdum eros. Nam congue lacinia nulla, vitae aliquet nisl tincidunt vel. Morbi gravida bibendum ipsum, at accumsan nisl suscipit ac. Sed accumsan cursus tortor a faucibus. Phasellus tempus, orci ac lacinia hendrerit, dui justo accumsan mi, congue dapibus massa turpis at lectus. Cras a tellus nisi. Aliquam vitae dolor id nunc lacinia fermentum et sit amet metus. Nullam viverra ante eu mauris ultrices nec adipiscing lectus dapibus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nam mollis commodo lacus, eget bibendum risus lobortis nec.

7 Acknowledgements

A   Data & Results

A.1   SVM Training & Testin

A.1.1   Confusion Matrix

Confusion Matrix												
	X	Y	a	c	e	i	l	m	r	t	u	v
X	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0
a	0	0	2	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0	0	0	0
e	0	0	0	1	0	0	0	0	0	0	0	0
i	0	0	0	0	0	0	1	0	0	0	0	0
l	0	0	0	0	0	0	1	0	0	0	0	0
m	0	0	0	0	0	0	0	1	0	0	0	0
r	0	0	0	0	0	0	0	0	0	1	0	0
t	0	0	0	0	0	0	0	0	0	1	0	0
u	0	0	0	0	0	0	0	0	0	0	0	0
v	0	0	0	0	0	0	0	0	0	0	0	0

Table 3: Confusion matrix of prediction on 'multivariate'

A.2   Font Samples

B   Project Timeline

Week 1: 10/28 – 11/5

- Submit project proposal
- Narrow down choices to one for the project
- Read up on related papers/books

Week 2: 11/6 – 11/12

- Continue reading related works

- Word Segmentation investigation

- Numerical digit classification investigation

Week 3: 11/13 − 11/19

- Character segmentation

- Dynamic time-warping experiments

- Alphabet characters classification investigation

Week 4: 11/20 − 11/26

- Font database

- Dynamic time-warping for classification

- SVMs for training on fonts and classification

Week 5: 11/27 − 12/5

- Improving individual results

- Heuristics for combining both predictors

- Dictionary spell-correction

- Report writing

C   Division of Labor

Reponsibilities of the members are outlined below. Our aim was to implement two different classification systems, one using SVMs and the other using DTWs, which were later combined for a final recognition step. This allowed a roughly equal division of labor whilst ensuring that both parties mutually were sharing information and results with one another.

- Sang Woo Jun
    - .idx tool development
    - Implementation of Time-Series Word-Segmentation

- feature-DTW implemtation
  - Spelling-corrector implementation

- Chong-U Lim

  - Compiling font training data for SVMs
  - Training and tuning SVM parameters over:
    * MNIST database of digits
    * Font characters for multiple fonts
  - Implementing merge strategy prediction tables for combined predictions

References

[1] Peter R. Norvig. http://norvig.com/spell-correct.html, 2011.