TYPING DIGIT CLASSIFIER
PATTERN RECOGNITION PROJECT

HENRY XIAO

1. Project Overview

This project is primarily a classifier design for recognizing typing digits. We follow the conventional classifier design process (see [1] section 1.4) to construct our classifiers. Typing digit features have been studied here from the training samples. A set of features can be extracted from our program and important ones have been showed through some statistics. Two classifiers have been derived using decision tree and nearest neighbor methods. We evaluate the classifiers through testing using different typing digit samples. We also propose some further discussions based on our studies through out this project.

The difficulty of recognizing typing digits is that various fonts existing today which make the feature extraction relatively hard as if more font options are considered. In general, if all font options are taken into account, the diversity of typing digits would match handwritten's. Then clearly, classifying typing digits will be equally hard as classifying handwritten ones. However, for the requirement of this project, we only develop our classifier based on some "regular" fonts and test on limited samples.

The two classifiers that we come out in this project are based on different criteria. We measure some intuitive features like "holes" in the digit etc. to come up the first decision tree classifier. This somehow does not work well because of various fonts as we have stated early. The latter classifier tries a different approach based on template matching. The feature set we measured at the beginning becomes useless here which is a little disappointing to us. The better testing results from the second classifier further shows the incapability of developing an highly accurate typing digit classifier with our intuitive feature set.

2. Digit Features

Feature extraction is an essential step towards a good classifier. As the beginning, we measure a set of intuitive features which can be directly visualized from many typing digits. The contents of the features are listed in Table 1 below.

We set the percentage for "H Line" and "V Line" to 90% and it is easy to verify that $Hole = Up\ Hole + Center\ Hole + Down\ Hole$. Some features like "Width" and "Height" are clearly not useful to be applied in the classification, and thus to be eliminated from our consideration later.

The template matching is not using the above features from the set. However, we create a $\{0, 1\}$ map for each of the digit extracted from an input digit image which

Date: November 2004.

Num#	Feature Name	Feature Description
1	Width	The bounding box width.
2	Height	The bounding box height.
3	Width/Height	The ratio of the bounding box width to the height
4	Hole	The number of holes in the digit.
5	Up Hole	The number of holes in the upper part of the digit.
6	Center Hole	The number of holes in the center
7	Down Hole	The number of holes in the lower part of the digit.
8	Notch	The number of notches in the digit.
9	H Line	The number of horizontal lines longer than given percentage of the width.
10	V Line	The number of vertical lines longer than given percentage of the height.
11	Digit/Area	The ratio of the black pixels to the total pixels
12	Hole/Area	The ratio of the pixels in the holes to the total pixels
13	Notch/Area	The ratio of the pixels in the notches to the total pixels.

TABLE 1. Intuitive feature set

becomes very useful for scaling the digit without applying some image processing algorithm. An example of the $\{0, 1\}$ map of a digit "8" is showed in Figure 1.

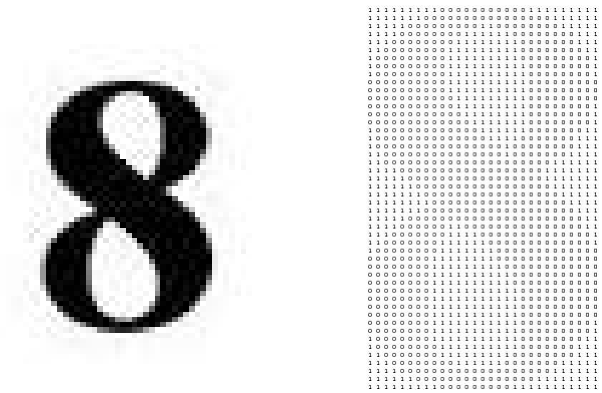

FIGURE 1. A digit "8" image and its $\{0, 1\}$ map.

It is not hard to imagine that the matching is simply a process of identifying agreed black (i.e.0) pixels between the template map and the input digit map. And the scaling of a map is relatively a straight forward task.

3. Classifier Design

Two classifiers are designed namely Decision Tree Classifier (DTC) and Nearest Neighbor Classifier (NNC) based on the classification method used.

3.1. **Decision Tree with Digit Features.** The DTC is based on some features from the Table 1. We measure the importance of each feature through studying on a set of training samples. For each digit, we have a subset of 150 samples with different fonts and sizes. (i.e. there are in total 15 fonts used and 10 samples with different sizes for each font.) Figure 2 shows an example sample image for digit "0" with certain font and different sizes.

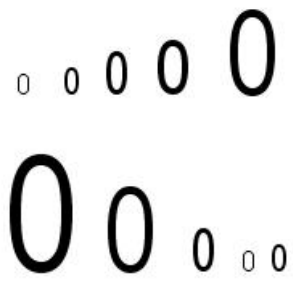

FIGURE 2. A training sample image for digit "0" with certain font and different sizes

We then collect the feature statistics from the training samples and identify possible features that can discriminate a digit from another, or others. The result statistics can be found in [2]. The final decision tree of our DTC is constructed manually using features identified from the studies. Figure 3 is a graph representation of our decision tree.

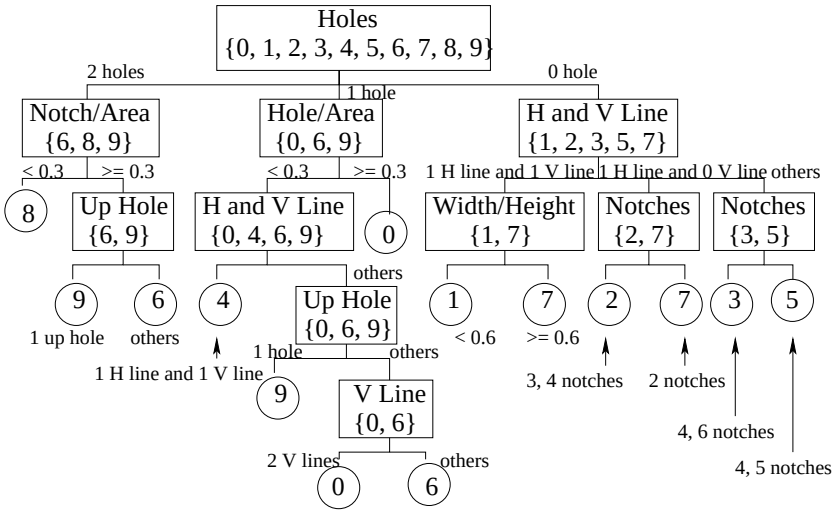

FIGURE 3. The decision tree for our DTC

Couple of things can be noticed from the graph. The DTC is rooted at the "Holes" which is regarded as the most important feature in our design. Thus, in

general, the digits with one hole or two (i.e. "0", "4", "6", "9", and "8") should have a better chance to be correctly classified than those without holes. The DTC has problem discriminating "2", "3", and "5" since the three digits are very likely in all our feature dimensions. The classification between "3" and "5" is no better than a random guess here.

3.2. **Nearest Neighbor with Template Matching.** The concept of the NNC is simpler than the DTC. The important thing is to get the template library right. The template library is a connection of digits with the value specified by the file name in our system (i.e. digit "4" template samples are stored in a file named "4_lib.dat"). The classification is based on the comparisons between the input samples and the library samples. The prediction result is the digit sample value given the minimum *distance*.

We have already seen the $\{0, 1\}$ map at the previous section, and it is only a problem of defining distance measure here in order to do the matching between two $\{0, 1\}$ maps. We measure the distance by the difference of the black (i.e.0) pixels between the two maps in percentage to the total black pixels. So formally, we define distance from one map to the other d as:

$$d = \frac{black\ pixel\ difference}{black\ pixel\ total}. \tag{3.1}$$

The finally distance D is calculated by applying Equation 3.1 twice with respect to the input sample and the library sample, and taking the average of the two d.

$$D = \frac{d_{input} + d_{library}}{2} \tag{3.2}$$

where $d_{input} = \frac{black\ pixel\ difference}{black\ pixel\ total\ of\ input\ sample}$, $d_{library} = \frac{black\ pixel\ difference}{black\ pixel\ total\ of\ library\ sample}$.

Notice that it is important to scale the library and the input samples into the same comparable size. This is done by taking the ratio of the two heights from the two maps and scaling the width by the same ratio. So the normalized two maps will have same number of rows with the number of column may differing. In order to simplify the normalization process, we further make the library contains "large" size digit such that the input sample digits have smaller size. This implies that we only need to scale our library samples by the ratio $r = \frac{ROW_{input}}{ROW_{library}}$, and then compare the normalized library map to the input sample map. It should be clear that because the two maps may have different number of columns, the number of total black pixels of the input sample map may be different from the one of the library sample map. Thus the distance D is more proper than single d.

We have stated the importance of the template library. And our current library is constructed from 10 digit image files respecting digits from "0" to "9". Each image contains 15 samples for a digit with different fonts. We demonstrate the digit "0" library image in Figure 4. The template library then consists of 10 template files storing $\{0, 1\}$ maps for each digit extracting from the 10 library images.

Figure 4 is marked with the fonts. We use five font families (i.e. Arial, Arial Narrow, Century Gothic, Times and Verdana) with respect to the columns, and use three font types (i.e. regular, italic, and bold) with respect to the rows. The library construction is very regulated consisting $10 \times 15 = 150$ samples in total. In other words, for each input sample, the classification program will do 150 comparisons

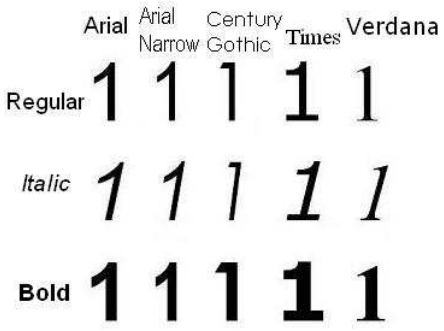

FIGURE 4. Digit "0" library image with 15 samples

to find out the best result with minimum D. We consider our construction of the template library fairly efficient for this project since the testing is limited with certain fonts too. One thing is noticeable from implementation point of view that the template matching NNC does have some significant delay at runtime compared with the DTC. The template size (i.e. number of samples in the library) is then critical and make 0 distance which is theoretically possible for nearest neighbor (i.e. the input sample is contained in the template library) is thus impractical. Above all, the question of how to choose the template samples is a hard one to be answered here.

4. Testing Result and Discussion

The testing results are proposed with our testing samples. We further discuss the testing results related with the features and the classifiers.

4.1. **DTC Result and Discussion.** We test the DTC with a testing sample set consisting 15 digit images and one character image. 10 digit images containing 32 samples each are designated for single digits from "0" to "9". Three digit images are with mixed digits. Other two digit images are used for different fonts. The testing set can be downloaded from [2]. We show the results from the DTC in Table 2 and 3.

Digit	# of Samples	Correct	Rate (%)
0	32	23	71.9
1	32	25	78.2
2	32	14	43.8
3	32	14	43.8
4	32	30	93.8
5	32	24	75.0
6	32	25	78.1
7	32	17	53.1
8	32	32	100.0
9	32	22	68.8

TABLE 2. DTC results on single digit images.

Image Des.	# of Samples	Correct	Rate (%)
mixed digit 1	40	28	70.0
mixed digit 2	40	25	62.5
mixed digit 3	32	20	62.5
mixed font 1	50	29	58.0
mixed font 2	50	32	64.0
character	16	0	0.0

TABLE 3. DTC results on various digit and character images.

From Table 2 on single digit images, we can see that digit "2" and "3"[1] are mainly misclassified. It is a predictable error from our DTC design. This reflects the fundamental limitation of using the intuitive feature set. Table 3 further shows the results on mixed digit images and character image. The main problem exposes through the test is that the italic font changes the behaviors of some important features such as "Hole/Area" and "Notch/Area". The consequence is the misclassifications between "0" and "4", and between "1" and "7". It has been cleared that our training sample set does not contain enough italic font digits to gather the statistics. One may argue that including sufficient italic font digits in the training phase to overcome the problem. However, this action shall reduce our useful feature set such that some features like "Notch/Area" and "Width/Height" may not be applicable in constructing the decision tree. We eventually conclude that really few improvement can be done with the current features here, and the DTC constructed can not give us an accurate digit recognizor. The DTC also can not discriminate digits from characters since it is generally hard to set up the rejections through the decision tree.

4.2. **NNC result and Discussion.** The same test that is done with the DTC is performed under the NNC. The results are listed in Table 4 and Table 5.

Digit	# of Samples	Correct	Rate (%)
0	32	28	87.5
1	32	32	100.0
2	32	31	96.9
3	32	26	81.3
4	32	32	100.0
5	32	32	100.0
6	32	29	90.6
7	32	32	100.0
8	32	29	90.6
9	32	32	100.0

TABLE 4. NNC results on single digit images.

The NNC yields some significant improvements over the DTC and is very accurate in fact. This somehow proves the correctness of our distance measure using Equation 3.2 by examples. The good classification results by our NNC and current

[1]we set the DTC to favor digit "5" when coming to classify "3" and "5"

Image Des.	# of Samples	Correct	Rate (%)
mixed digit 1	40	40	100.0
mixed digit 2	40	39	97.5
mixed digit 3	32	28	87.5
mixed font 1	50	44	88.0
mixed font 2	50	36	72.0

TABLE 5. NNC results on various digit images.

template library inspires us to do some "extreme" testing with some "wild" fonts. Figure 5 shows two kinds of such fonts. Clearly, the DTC will fail on those fonts, which are more of handwritten looking.

(a) Algerian font (b) Bernhard Fashion font

FIGURE 5. Wild font digit test samples.

Table 6 shows the results of applying the NNC to five different fonts. To our surprise, the NNC still has a strong performance considering the template library is unchanged, which contains no sample instance related with the testing samples. The reason may be informally stated as that the distance measure of the agreed black pixels preserves some shape information, and the shapes of a digit with various fonts should mainly agree with each other since visually, we can recognize digits of different fonts by shape.

Font Des.	# of Samples	Correct	Rate (%)
Algerian	20	15	75
Berhard Fashion	20	9	45
French Script	20	16	80
Joker man	20	18	90
Viner Hand	20	8	40

TABLE 6. NNC results on various digit images.

Our motivation of designing the NNC with template matching comes from the poor performance of the DTC with the feature set. And indeed, the NNC is much better a digit recognizor than the DTC. The template library is a further subject

which could be studied to improve the NNC's performance. It may not a easy job as mentioned above, however. As far as the character concerned, we can set up a threshold for the distance D to reject some input samples. We experiment on the threshold and find above 0.3 on D could be rejected in most of the cases. But still, the NNC can not give a good way of discriminating the characters. It will fail in cases like character "O" that is almost the same as digit "0", or "q" which is very much like digit "9". Nevertheless, we conclude the NNC based on the template matching is a very acceptable digit classifier, especially for typing digits which are usually with regular fonts.

5. Summary

This typing digit classifier project is a good way of learning pattern recognition in practice. We have developed some precious knowledge through designing the classifiers. Despite the fact that pattern recognition is an intense research and practical field, some fundamental principles still hold here. The experience of designing the two classifiers gives us a vivid lesson of the well-known "Occam's Razor" principle as a complicated feature set may not work better than a simple measure.

Furthermore, the variations of the features really educate us. Recognizing typing digit which seemed trivial in the first place is not a simple task. Extracting feature for typing digits is far more complicated than getting features from visualization. And coming out a highly accurate classifier is a even harder thing to do.

References

1. P. E. Hart R. O. Duda and D. G. Stork, *Pattern classification*, second ed., John Wiley & Sons, Inc., 2001.
2. H. Xiao, *Typing digit classifier web page: http://www.cs.queensu.ca/home/xiao/pr.html*, October 2004.

CISC 859 Pattern Recognition 2004 Fall
E-mail address: xiao@cs.queensu.ca