`

[10601 Machine Learning Project Final report]

Optical Character Recognition with Hidden Markov Models

Jay-Yoon Lee
LTI & Lane Center
School of Computer Science
jaylee@andrew.cmu.edu

Minhee Jun
ECE Department
Carnegie Institute of Technology
mjun@andrew.cmu.edu

1. Project Topic

Optical Character Recognition(OCR) in word recognition using HMM.

2. Introduction

2.1 Importance of OCR

OCR is a technology that will make entering of characters, words, and sentences into computer more intuitive and easy as natural writing can replace keyboard. Furthermore, this technology will also enrich our lives by providing accumulated knowledge through digitalization of historic documents and literatures that was handwritten before. This process of digitalization in OCR will allow people to access freely over the web and also enable people to search inside these documents, and thus aiding and expediting researches.

2.2 OCR with HMM

In this project, we use Hidden Markov Model (HMM) in order to classify words using pixel information from characters. Basic idea of HMM is guessing sequence of hidden states based on the observed values. Because the model extracts the most likely sequence of states, the model is known to capture the temporal property of data. Since words have temporal sequence of characters, we experiment handwritten word classification with HMM and examine how HMM can enhance the character recognition.

2.3 Feature Selection for OCR

While understanding proper classifier for OCR is important, it is also important to know the form of feature space that best represents character's information that enables classifiers to discern one character from another. In order to do so, in training the emission probability, we used comparatively simpler multinomial Naïve Bayes model to readily observe the effect of different feature space to OCR. After comparison of features, we tried to combine features in a way that compensates other's weakness in OCR and also compare the performance of the system with previously studied OCR systems using different classifiers.

3. Model

3.1. HMM structure

We set the state, a latent variable, with 26 possible lower-case alphabet set {'a', 'b', 'c', 'd', …, 'x', 'y', 'z'} and sequence of these hidden states as words. Among the three parameters A, B, $\prod$, we first obtained $\prod$, initial probability of each state, and A, transition probability matrix. Initial probability $\prod$ is the occurrence rate of certain character in the first position of a word. The equation for element of $\prod$, $p(z_n)=\pi_k$ for each $z_n = k$ and the element of A, transition probabilities $p(z_n|z_{n-1})=A(j, k)$ when $z_{n-1}=j$ and $z_n=k$ is given below.

$$p(z_n) = \pi_k = \frac{P(z_1 = k)}{\sum_{j=1}^{K} P(z_1 = j)} \qquad A_{jk} = \frac{\sum_{n=2}^{N} P(z_{n-1} = j, z_n = k)}{\sum_{l=1}^{K}\sum_{n=2}^{N} P(z_{n-1} = j, z_n = l)}$$

Because we have labeled dataset, we directly calculated these distributions matrices from the equation above rather than using Baum-Welch algorithm.

While the process of calculating A and ∏ was fairly straightforward, there are several ways to calculate the emission probability matrix B. Here, we used the simple Naïve Bayes model to obtain emission probabilities B(k, y) = p(x_n =y| z_n =k) when x_n =y and z_n =k. As a base model, we defined Naïve Bayes model with feature vector of 128(16 by 8) conditionally independent pixels given character.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0
0	1	1	1	1	1	0	0
0	1	0	0	0	1	1	0
1	1	0	0	0	0	1	1
1	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1
1	0	0	0	0	0	1	1
1	0	0	0	1	1	1	0
1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 1. Character 'O' represented in 16 x *8 pixels*

As the dataset was given, each pixel is modeled as binary value and probability of each coordinate having value 1 is trained through MAP estimator. The reason for using MAP estimator is to prevent effect of some coordinate that having 0 MLE estimator on the whole likelihood value since the likelihood function is calculated through multiplication. (For instance, coordinate (0,0) is very much likely to have 0 MLE estimator.) The MAP estimator for specific coordinate (i,j) of pixel matrix A for given alphabet α with β prior belief is given by

$$MAP = \frac{\sum_{\alpha \in trainingSet} \delta(A(i,j)=1) + \beta}{\sum_{trainingSet} \delta(alphabet = \alpha)}$$

3.2 Additional Features

3.2.1 Special Features

Our previous accuracy in the midway report was achieved up to 67.53% using only Naïve Bayes and HMM algorithm on 16x8 pixels of handwriting data. Then, we have been looking for algorithms and features to improve our accuracy rate. We tried to strengthen our classifier by applying more algorithms. We found that Principal Component Analysis(PCA) with polar transformation[1] achieves the accuracy up to 89% and that Genetic Algorithm with Directional Features[2] accomplished the accuracy around 80% [2]. However, we found that we have a strong limitation on our data set for applying those methods to our case because the resolution of our data set is only 128(=16x8) while other experiments are performed in higher resolution. (very very high resolution enough to extract directional components in [1] and 64x64(=4096) in [2]) As we have a data set with the pretty low resolution 128, we tried to add some special features which help us distinguish characters which were unclear to discern with our previous classifier. We analyzed the common patterns of classification in our data set, and add features one by one to improve accuracy.

3.2.1.1 Polar Coordinate

In order to express the loop information in the character such as 'a', 'b', 'o' and so on, we implemented polar coordination. For example, if you think of 'o', it will have similar radius from

the center and will have number of different angle values within the similar radius range. The character 'o's polar coordinate is depicted below. As we told before the pixels are confined to one or two radius values that are represented by vertical axis.

0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	2	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0
0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0
0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2. Character 'O' represented in 16 x *16 polar coordinate*

3.2.1.2 Bias

Even though the data set is pre normalized, we found that the handwritten characters are by itself biased: more pixels on one side, to left or right, depending on characters. For example, p and b have much more pixels on upper and lower pixels, respectively. With this observation in mind, we divided the 128 pixels into 4 segments and tried assess to which segment the character is biased in the training.

3.2.1.3 Min Max (row, column)

If we think of pairs of characters (x, y), (n m), (v w), because of its similarity with the pair, it would be hard to differentiate the characters in pair with Naïve Bayes. However by using additional information of max and mean value for these characters, it might be possible to discern these because we are able to observe that latter characters will have higher max row number because they are wider.

3.2.1.4 Vector Sums

Column vector sums can show the distribution of characters horizontally, and it could be important since we write words from left to right. We only took care of horizontal in particular because all the samples were normalized and the image was stretched from beginning to the end horizontally.

3.2.2 Relation of Features and Emission Probability

- **Obtaining Emission Probability**

  In training emission probability, we used Naïve Bayes and MAP estimator as shown above. In doing so, because we used conditional independence assumption, the complexity has reduced from 2^{128} to the order of 128.

- **Combining features in Emission Probability: Normalizing**

  We are using generative model characteristic of Naive Bayes in order to generate the emission probability of HMM and used this throughout all the features. The problem is, for example, when I try to use "bias" feature which is comprised with four features(say X1) along with 128 pixel information(say X2), then when we try to combine those two features, the scale of P(X1|Y) and P(X2|Y) wouldn't match when we try to multiply P(X1|Y)P(X2|Y). The value of P(X1|Y) would be much smaller than P(X2|Y) because the probability is less than one and X1 is 128 multiplications of it while P(X2|Y) is 4 multiplications of probabilities. In short, if we combine the

features in this way, the effect of X2 features would be diluted to 4 out of 132 features, while the information it has is bigger than just one pixel. So we came up with a normalization term that can match the scale of probabilities for different features. The idea is just to use as many times to match the largest size of pixels. Continuing this example, it would be multiplying P(X2|Y) 32 times in order to match P(X1|Y)

- **Combining features in Emission Probability: Weight**
  Even though we normalized the scale of each features, it does not mean we consider every feature to have the same relevance to the classification rule. For that, we weighted feature sets based on their accuracy in Naïve Bayes character classifier when combining them for the generation of emission probability for HMM. (This is possible through the generative model and conditional independence assumption of Naïve Bayes.) The equation we used for the emission probability and the weights of given features and its accuracy is as follows:

  Accuracy $\alpha_1, \alpha_2, \alpha_3$ corresponding to features X_1, X_2, X_3 and label Y will give

$$\log(P(X \mid Y)) = \frac{1}{\alpha_1 + \alpha_2 + \alpha_3} \sum_{i=1}^{3} \alpha_i \log(P(X_i \mid Y))$$

3.3 Decoding Algorithm(Viterbi)

In decoding the sequence, Viterbi Algorithm was implemented to find the most probable character sequence for a given pixel image. We gained the quantity

$\omega(z_n) = \max_{\{z1, z2, z3, \dots, zn-1\}} p(\ x_1,\ x_2, \dots,\ x_n,\ z_1, \dots, z_n\ )$ given X= $p(\ x_1,\ x_2, \dots,\ x_n),\ x_i \in \{0,1\}$

$\omega(z_n) = \max_{\{z1, z2, z3, \dots, zn-1\}} p(\ x_1,\ x_2, \dots,\ x_n)$

$\omega(z_{n+1}) = \ln p(\ x_{n+1} \mid z_{n+1}) + \max_{\{z1, z2, z3, \dots, zn-1\}} \{\ln p(\ z_{n+1} \mid z_n\ ) + \omega(z_n)\},$

$\omega(z_1) = \ln p(\ z_1) + \ln p(\ x_1 \mid z_1\ )$

As we can get recursive model like above, $\omega(z_n)$ can be evaluated based on the above three kinds of probabilities: the initial probability $p(z_n)$, the transition probabilities $A(z_n \mid z_{n-1})$, and the emission probabilities $p(x_n \mid z_n)$. Once we calculate the initial condition $\omega(z_1)$, we can evaluate $\omega(z_n)$ when n=2, 3, … , N recursively.

4. Experiment

We used MATLAB in order to implement Hidden Markov Model and Naïve Bayes.

4.1 Data

The dataset was acquired from Benjamin Taskar's, professor of Univesity of Pennsylvania, webpage and was originally collected by Rob Kassel at MIT Spoken Language Systems Group. From the original dataset, Taskar selected a "clean" subset of the words and rasterized and normalized the images of each letter and removed first letter of each words because it was in uppercase, forming 6877 word, 52,152 character, all-lowercase dataset.

Data set is given with total 52,152 characters and 128 pixels, which is representation of 16 by 8 space, per character. We divided dataset into three parts, train set, validation set and test set. The given dataset was comprised of 10 cross validation folders(labeled from 0 to 9) and those folders were reallocated to each datasets: folder 0,1,2 to train set, folders 3, 4 to validation set, and folder 5, 6, 7, 8, 9 to test set. Following graph is the number of occurrence for each character in the data set. [3]

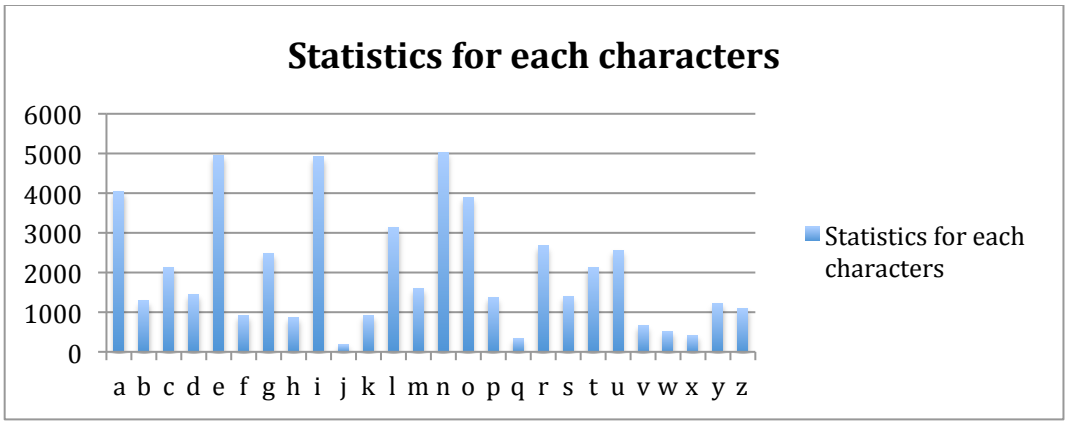

Figure 3. Number of each characters in dataset

4.2 Naïve Bayes

Implemented Naïve Bayes for character recognition by using every 128 pixel as feature. In order to obtain MAP estimator, we tested on validation set to get the optimal beta prior and chosen 1 as β value. The test result with β=1 was 62.77%.

4.3 Effects of Additional Features

As we shall show in discussion section, there are sets of characters such as (‘a’, ‘c’, ‘e’, ‘o’) and (‘i’, ‘l’) that misclassifies one another often in our baseline Naïve Bayes HMM system. In aim to overcome this weakness, we came up with extra features explained in part ‘3.2 extra features’ and following Table 1, *Figure* 4, and *Figure* 5 are the result of the experiment.

Table 1. Accuracy comparison with features

Original Pixel	Polar Coordinate	Checking Min & Max	Column Vector Sum	Biased Quadratic	Naïve Bayes	HMM
√					63.70 %	67.53%
	√				67.14 %	72.11%
		√			34.79 %	-
			√		51.25 %	-
				√	46.93 %	-
		√	√	√	65.73 %	-
√	√	√	√	√	-	74.86%

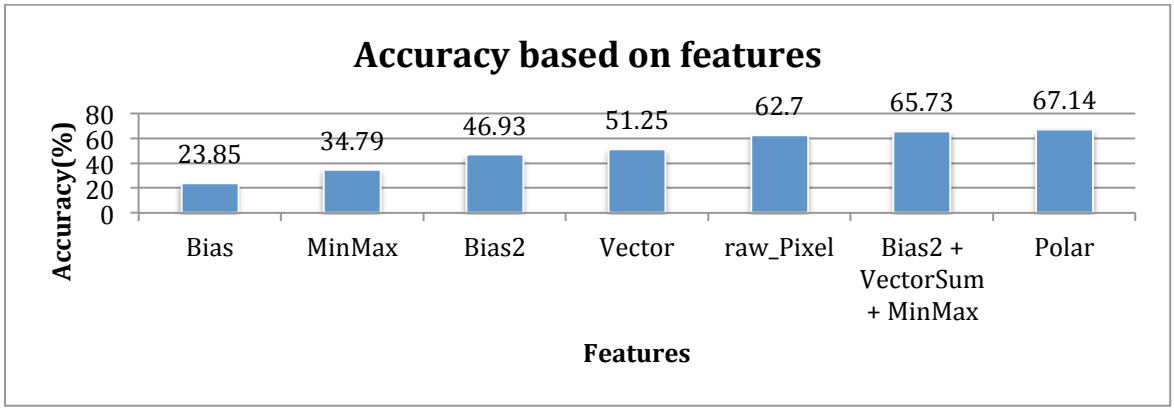

Figure 4. Accuracy using different features

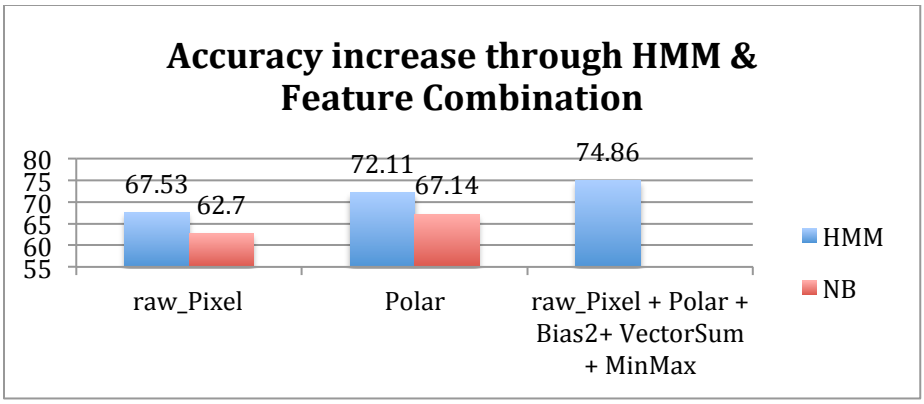

Figure 5. Accuracy increase by feature combination & HMM

If you see the *Figure* 5, we were able to pull the accuracy up to 74.86% by using combined feature and HMM from 62.7% of just using Naïve Bayes on raw pixel information. For a single feature, the most informative feature of all was the polar coordinate with 67.14% of accuracy Naïve Bayes classifier. These experiment results shows that the same information can add additional insights to the same classifier, Naïve Bayes in this experiment, when it's properly converted and that their combinations can be useful when combined appropriately. Furthermore, this experiments also again verifies that HMM can boost the accuracy of character recognition by character's sequential nature in words. [1]

5. DISCUSSION

5.1. Error Analysis

First, we checked the distribution of misclassified cases for each character to analyze the types of errors caused in original character pixels. The misclassified characters are calculated, and their distributions are plotted as below in *Figure 6, 7 to* show the misclassified character distributions for character 'c' and 'i' respectively. We were able to observe that each character is misjudged into just a few other characters. The same phenomenon that there are a few characters frequently shown up are also found in the plots of other characters.

To improve this error rate, we needed to figure out which characters can be mainly confused with the actual character in this classification we used. To concentrate these high-portioned misclassified characters, we checked which characters consist of more than 10% of errors of each given original characters in the given derived plots. In the *Table 2* below, the original characters are given in the shaded entries, and frequently confused characters that take more than 10% of total errors are given in the white entries. For example, the character 'k' is frequently misclassified to 'b', 'e', or 'h', and the character 'y' is easy to be classified as 'g' with the classifier we implement.

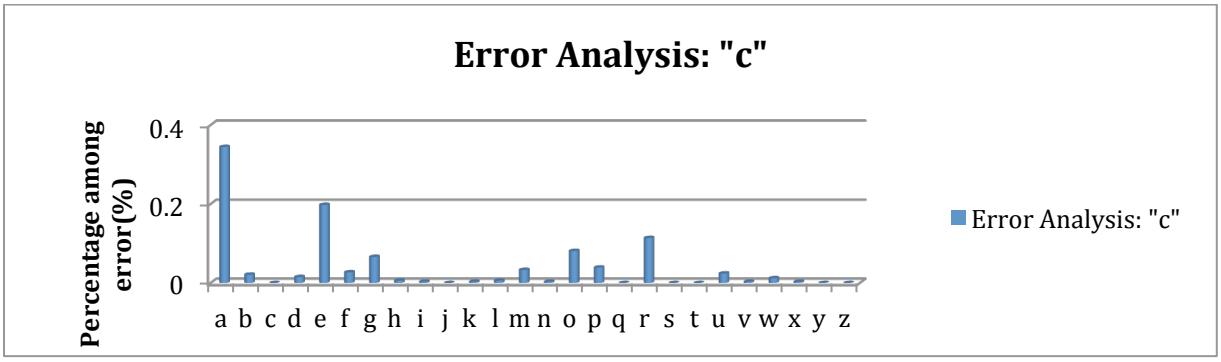

Figure 6. Error Analysis of charaacter "b"

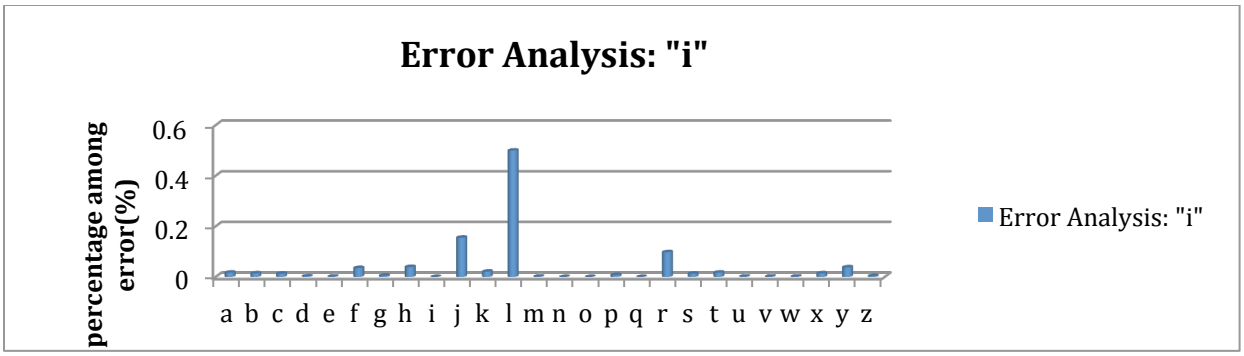

Figure 7. Error Analysis of character "i"

Table 2. Error Analysis – Most frequent errors

a	m, n, o	h	-	o	a, g	v	u, w
b	d, h, u	i	j, l	p	f, l, q, r	w	m, u
c	a, e, r	j	d, g, i	q	g, l, p,	x	n, y, z
d	a, j	k	b, e, h	r	c, f, p	y	g
e	a, c, p, z	l	c, i, y	s	a, g	z	a, e, x
f	p, r, t	m	a, n	t	d, f, l, z		
g	d, o, q	n	a, m, o	u	a, o, w		

This table gives us how our classifier make misclassifications. Character ‘a’ can be confused with character ‘e’, ‘m’, ‘n’, and ‘o’ because their pixels are commonly distributed around a circle. Character ‘m’ and ‘n’ are confused to each other because their pixel positions are densely located on the top. Also, characters ‘i’, ‘j’, and ‘l’ are confused as their pixel are mainly distributed around the central vertical line in their pixel by handwriting. So, we thought about adding additional features to the previous classifier to increase accuracy. First, we considered of transforming the pixel positions of characters into polar coordinate. The mean point of pixel positions is mapped to the origin in the polar coordinated, and the distance and angle information of each pixel from the mean point in Cartesian coordinate are calculated to do polar transformation. Then, we were able to get the new distribution of misclassified characters.

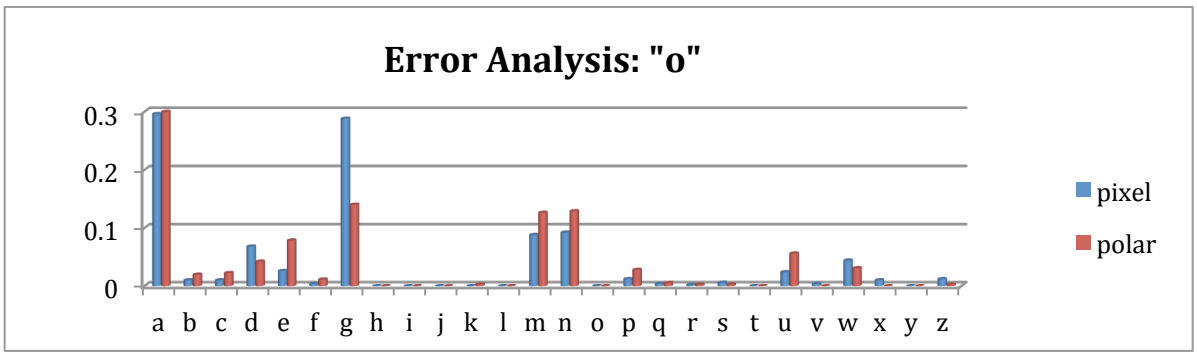

Figure 8. Error Analysis “o” pixel vs. polar

The above plot in *Figure 8* tells us about the distributions of pixel case and polar-transformed case of character ‘o’. As the distance from the mean point is given directly in the polar-transformed case, we can decrease the probability that ‘o’ is misclassified as ‘g’. This fact is exactly coherent with our experimental result that the polar-transformed matrix has higher accuracy (67.14% with Naïve Bayes, 72.11% with HMM) than pixel matrix (63.70% with Naïve Bayes, 67.53% with HMM). In addition, there is another instance for given

character ‘c’ in *Figure* 9, which shows us how the polar-transformed matrix could be beneficial as it provides additional information to our classifier. The misclassified rate of character ‘a’ and character ‘e’ are relatively lower in polar-transformed case than in the original pixel form while the misclassified rate of character ‘o’ relatively lower in the pixel form. So, we can expect to compensate the risk to predict the wrong characters by combining these two types of information given by pixel matrix and the polar matrix. This has to contribute to reach our final result of moving the experimental accuracy almost 10% up.

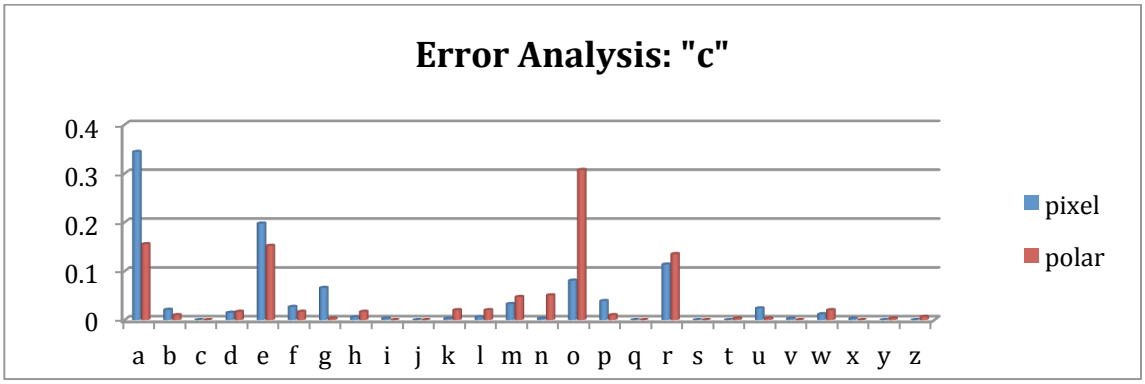

Figure 9. Error Analysis "c" pixel vs. polar

Moreover, our three additional features (Min-Max, Vector Sum, and Bias in quadratic plane) provided more information to classify characters clearly in our classifier. Min-Max measures the minimum and the maximum numbers of pixels along column vectors and row vectors separately. While its accuracy is relatively low (34.79%) when it independently classifies characters in Naïve Bayes, this feature is also important as an auxiliary feature in addition to a primary classifying feature. For example, character ‘m’ and character ‘n’ is hard to discern in our classifying in either pixel matrix or polar matrix. However, if we use this Min-Max features that calculate the sum of dot pixels along the row vectors, ‘m’ and ‘n’ can be distinguished. Next, Vector sum give a standard how to distinguish two left-right symmetric characters. For instance, character ‘p’ and character ‘q’ can be discerned with the column Vector sum features while the Min-Max features have similar probability values on both characters. (We did not count Vector sum along the row because it is normalized for horizontal direction) Finally, Bias in quadratic plain is a strong feature that provides us a clue to classify characters clearly. Pixels are divided into four planes along vertical and horizontal line which passing the mean point of the handwritten character and the sum of dot pixels in each plane is calculated for this feature. Then, this information allow us to classify characters with properties that were not able to be extracted with previously suggested features. For example, character ‘a’ and character ‘e’ is hard to discern even with Min-Max, Vector Sum after applying pixel matrix feature and polar-transformed matrix feature, but this feature, Bias in quadratic plan, enable us to do that.

6. Reference

[1] Discriminative HMM Training with GA for Handwritten Word Recognition, Tapan K Bhowmk. et. al., India Statistical Institute, Kolkata, India

[2] Discriminativ Training for HMM-Based Offline Handwritten Character Recognition, Roongroj Nopsuwanchai et. al., Computer Laboratory, University of Cambridge, {CB3 0FD, CB2 1PZ}, UK

[3] B. Taskar. OCR dataset [http://ai.stanford.edu/ btaskar/ocr/] . Stanford University, Arti_cial Intelligence Lab, Stanford, CA, 2003.

Team member role

Jay-Yoon: implemented Naïve Bayes for every feature spaces and calculation of emission probabilities.
Min-Hee: implemented Hidden Markov Model and adjusted weights of emission probabilities for combined features.