Hidden Markov Model with Parameter-Optimized K-Means Clustering for Handwriting Recognition

Weijie Su
School of Mathematical Sciences
Peking University, Beijing, China
suweijie444@gmail.com

Xin Jin
Department of Computer Science
Peking University, Beijing, China
jinxin@net.pku.edu.cn

***Abstract*—Handwriting recognition is a main topic of Optical Character Recognition (OCR), which has a very wide application area. Hidden Markov model is a popular model for handwriting recognition because of its effectiveness and robustness. In this paper, we propose a hidden Markov model with parameter-optimized k-means clustering for handwriting recognition. We explore two deep features of the images of characters, thus significantly boosting the effectiveness of k-means clustering. The experiments show that our model largely increases the average accuracy of HMM with k-means clustering to 83.5% when the number of clusters is 3000.**

***Index Terms*—OCR; HMM; k-means; clustering;**

I. Introduction

Optical Character Recognition (OCR) is a special type of pattern recognition, which aims at mining pattern information through computer, statistics and computational mathematics methods. Handwriting recognition is a main topic of OCR. It can be applied to a wide area and attracts attentions both from academia and industry. Although there are lots of works have studied this topic, the problem is still not perfectly solved due to its complexity. Typical methods include Hidden Markov Model (HMM), Markov Random Field (MRF), Support Vector Machine (SVM) and Max-Margin [4][5][2][6]. Among these models, HMM is a popular one because of its effectiveness and robustness [3].

Typically, a handwriting character is represented by a $m \times n$ binary matrix after segmented. In HMM, the real state is the character the matrix stands for. However, because the total possible number of different $m \times n$ binary matrices is 2^{mn}, which is too large compared to learning samples, we cannot directly regard every matrix as a observation state. We should firstly cluster the matrices and then regard the clusters as hidden states. Most previous works think that k-means is not a good clustering method in this problem. However, through deeply analyzing the problem, we find that k-means can improve to be an excellent clustering method for the problem.

An $m \times n$ matrix can be regarded as a mn-dimension vector. In k-means, the distance is usually defined as the ordinary vector distance. If the distance between two vectors is large, it means the two matrices are largely different. However, such definition is too rough. It mainly has two drawbacks. Firstly, it does not consider the influence of neighbor pixels on a certain pixel. Secondly, it does not take the weights of pixels in different places into account. In this work, we improve the two shortages of k-means clustering and propose an HMM with parameter-optimized k-means clustering for handwriting recognition. In our method, we consider the influence of neighbor pixels and different weights of pixels in different places of the matrices, which are denoted by three parameters. Then we optimize the three parameters in order to gain the highest accuracy rate. We conduct a series of experiments to compare HMM with original k-means clustering and HMM with parameter-optimized k-means clustering. Results show that our parameter-optimized k-means clustering improve the average accuracy from 78.0% to 83.5% when the number of clusters is 3000.

II. Related Work

Many works have been done for handwriting recognition [4][5][2][6]. The HMM model for handwriting recognition has been exploited by many researchers. Based on HMM, related works can be divided into two main categories. One regards each character as a real state [7] and the other regards each word as a real state [8]. We choose the former one because it is more suitable for the entire language and does not require additional information. The latter one usually requires a dictionary to get the words. If the total number of words is large, the method will be not so efficient. So the latter one is more suitable for domains with a dictionary of small size. The former one does not need such information and has a much wider area for application.

Among numerous different clustering method, k-means clustering is one of the most popular because of its low cost and simplicity [10]. K-means clustering is a famous method for clustering. It is firstly proposed in [9]. It aims at minimizing the differences within each clusters and maximizing the differences among different clusters. The advantages of k-means clustering are simple, efficient and low-cost, which makes it widely used in various fields [10]. However, for handwriting recognition, this method is not recommended to be used in HMM because some experimental results show its performance does not meet a level of satisfaction [11][12]. In our work, we find two ways to improve it. The results show that k-means can improved to an excellent clustering method for HMM.

III. Model Description

A. Hidden Markov model

Generally, each handwriting character is represented by an $m \times n$ binary matrix. If the value of a pixel is 0, it means the pixel is white; if the value of a pixel is 1, it means the pixel is black. Given a sequence of binary matrices, our problem is to recognize which of the characters each binary matrix stands for. HMM is a suitable approach for this problem. In HMM, the real states are the characters. Intuitively, all the different $m \times n$ binary matrices can be regarded as observation states. However, the total number of such matrices is 2^{mn}, which is too large even with small m and n. If we do not reduce the number of observation states, it will result in severe sparse problem. A smart solution is to employ clustering method. We can firstly cluster the matrices and then regard the different clusters as observation states, which can largely reduce the number of observation states. After clustering, we adopt Baum-Welch algorithm to approximately calculate the parameters of our HMM, which includes initial distribution of characters π, transition probability $P = (p_{ij})$ and generative probability $Q = (q_{il})$. Here p_{ij} is the probability of transition from the i^{th} character to the j^{th} character, where $1 \leq i, j \leq s$, s is the number of characters; q_{il} is the probability of generating the l^{th} observation state given the real state is the i^{th} character, where $1 \leq l \leq k$, k is the number of observation states. Then, given the sequence of testing binary matrices, we apply Viterbi algorithm, which uses the expectation maximization method, to predict which the real character is for each binary matrix.

In the step of clustering, we employ a method based on k-means clustering, which we call parameter-optimized k-means clustering. In order to apply k-means method, we should identify matrices as points in Euclidean space. A naive way is just to identify the binary matrix as a vector of $\mathbb{R}^{mn}$, with each component 0 or 1. Given two points

$$X = (x_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}, Y = (y_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}, \quad (1)$$

the distance between X and Y can be intuitively defined as normal Euclidean distance:

$$d(X, Y) = \sqrt{\sum_{1 \leq i \leq m, 1 \leq j \leq n} (x_{ij} - y_{ij})^2}. \quad (2)$$

However, if only using such definition of distance, the accuracy of the recognition would be very undesirable, as pointed out in many works [11][12]. By further analyzing the problem, we find two major shortages of the definition. Our work aims to solve these two shortages. Our results show that our parameter-optimized k-means clustering can significantly improve the performance of original k-means clustering. We discuss the two shortages and how to improve them in the following two sections.

B. Influencing of neighbor pixels

The first shortage of the definition in Equation (2) is that a slight translation of a character in a matrix can result in a large distance from the original matrix. Suppose a matrix A_1 denotes the character a. Then every pixel in matrix A_1 move from left to right for one pixel. We denote the new matrix by A_2. Then although the two matrices, A_1 and A_2, both stand for the character a, the distance between A_1 and A_2 under Equation (2) is very large. An efficient modification is to adopt an averaging method. We use the notion $(i_1, j_1) \sim (i_2, j_2)$ if (i_1, j_1) and (i_2, j_2) are adjacent. For a point X, we can define the average processed version of X':

$$x'_{ij} = x_{ij} + \lambda \sum_{i'j' \sim ij} x_{i'j'}. \quad (3)$$

Here λ is a positive constant. When λ is small, X' is close to X, whereas a large λ is an indication of large influence of neighbor pixels. The choice of λ depends on which λ enjoys the best accuracy.

C. Weight consideration

There is another important shortage about the distance definition in Equation (2). If we simply use the ordinary Euclidean distance, it implies that each pixel in the $m \times n$ matrix is equal important. However, it is not the case. The weight of pixels which are close to the edge of the matrix is not necessarily same to the weight of central ones. Due to this observation, without loss of generosity, we assume that the weights of central pixel in the matrix is 1, and the weight of other pixel follows the exponential rule:

$$x'_{ij} = \exp(\eta|i - \frac{m}{2}| + \mu|j - \frac{n}{2}|)x_{ij}, \quad (4)$$

where η, μ are both constants which will be optimized through experiments.

D. Calculation of initial parameters

Combining the two consideration discussed in two previous sections, we have a transformed version $\tilde{X}$ of X:

$$\tilde{x}_{ij} = \exp(\eta|i - \frac{m}{2}| + \mu|j - \frac{n}{2}|)(x_{ij} + \lambda \sum_{i'j' \sim ij} x_{i'j'}). \quad (5)$$

Then we can adopt k-means clustering on $\tilde{X}$ instead of X. Before using Baum-Welch algorithm, we need to calculate the priori parameters of Hidden Markov model. We use a frequency-based method to calculate all these parameters. Suppose the number of clusters is k. For $1 \leq i, j \leq s, 1 \leq l \leq k$, suppose the i^{th} character appears P_i times in the training data, and the adjacent pair of the i^{th} character and the j^{th} character appears Q_{ij} times. Furthermore, R_{il} denotes the number of the i^{th} character with the l^{th} observation states in the training data. Therefore, it is reasonable to set the transition probability matrix as

$$\hat{p}_{ij} = \frac{Q_{ij}}{\sum_{t=1}^{s} Q_{it}}, \quad (6)$$

the priori generative probability distribution is

$$\hat{q}_{il} = \frac{R_{il}}{\sum_{t=1}^{k} R_{it}} = \frac{R_{il}}{P_i}, \quad (7)$$

and the priori initial distribution is

$$\hat{\pi}_i = \frac{P_i}{\sum_{j=1}^{s} P_j}. \tag{8}$$

IV. Experiment

A. Experimental Settings

Our experimental dataset consists of 52152 characters and 6877 words [6], which has removed capitalized leading characters. Each character is rasterized into an image of 16×8 binary pixels, which can be identified as a 16×8 matrix or a 128-dimensional vector. Examples are shown in Figure 1.

Fig. 1. Examples of the dataset

We use 10-fold cross-validation to calculate the average accuracy. The experiment is repeated for 10 times. Denote the accuracy rate of the ten simulations by $AR_1, ..., AR_{10}$. Then the average accuracy rate AR is

$$AR = \frac{\sum_{i=1}^{10} AR_i}{10}. \tag{9}$$

B. Choice of optimal parameters

In this section our job is to find the optimal triple $(\lambda_0, \eta_0, \mu_0)$ with highest prediction accuracy. An immediate way to find the most desired triple is to calculate the average accuracy of all such possible triples. However, it is too time consuming since there are too many such pairs. Therefore, we make an assumption that the effectiveness of λ is independent of η, μ. In this case, it is justifiably to set $\eta = 0, \mu = 0$, and obtaining the optimal λ_0' which enjoys the highest average accuracy. On the other hand, setting $\lambda = 0$, we vary the value of (η, μ) to catch the optimal pair (η_0', μ_0'). Then we use $(\lambda_0', \eta_0', \mu_0')$ as an approximation of $(\lambda_0, \eta_0, \mu_0)$.

1) Finding the optimal η and μ: In the first group of experiments, we set $\lambda = 0$ and cluster number $k = 100$. We vary η and μ in the ranges $[-0.25, 0.5]$ and $[-0.5, 1]$ respectively. The results are shown in Figure 2. From Figure 2, we observe that when (η, μ) is around $(-0.1, 0.1)$, the average accuracy is the highest. However, if we immediately take $(-0.1, 0.1)$ as the optimal pair of (η, μ), the estimation will be too rough. In order to find the more precise pair, firstly we fix $\eta = -0.1$, and let μ vary near 0.1. The relationship between average accuracy and μ when $\eta = -0.1$ is shown in Figure 3. From Figure 3, we find that when $\mu = 0$ it achieves the maximum accuracy 61.12%. Therefore, we choose $\mu_0' = 0$. Then we fix $\mu = 0.1$ and vary η near -0.1. The result is shown in Figure 4. From the figure, we can easily see that the

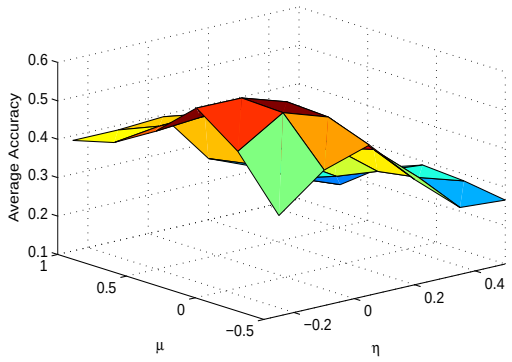

Fig. 2. Average Accuracy vs. η vs. μ

when $\eta = -0.15$ the average accuracy achieves the maximum value 58.87%. Therefore, we choose $\eta_0' = -0.15$. So we set $(-0.15, 0)$ as optimal pair (η_0', μ_0'). **Remark:** The above

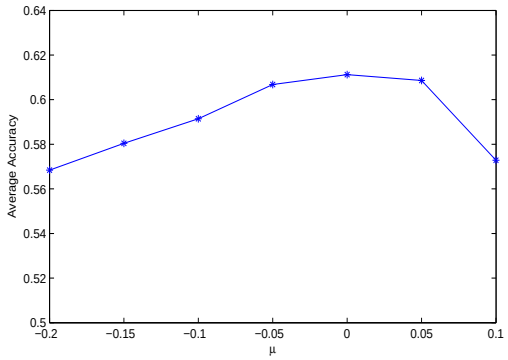

Fig. 3. Average Accuracy vs. μ

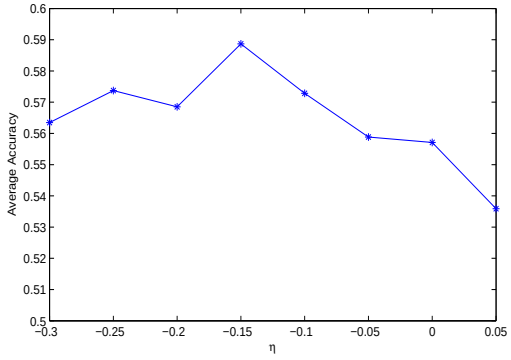

Fig. 4. Average Accuracy vs. η

discussion implies that for vertical direction, central part of the pixel matrix is more important, while there is no significant difference in horizonal direction.

2) Finding the optimal λ: In the second group of experiments, we fix $(\eta, \mu) = (0, 0)$ and vary λ to find its optimal value. Intuitively, no matter how much influence the neighbor

pixels have on a certain pixel, the influence cannot overwhelm the value of the pixel itself. So the optimal value of λ should be no larger than 1. Therefore, we search the optimal value for λ within the interval $[0, 1]$. The relationship between average accuracy and λ when $(\eta, \mu) = (0, 0)$ is shown in Figure 5. From the figure, we can see that the average accuracy increases along with λ. When λ is around 1, the average accuracy achieves the maximum value. Therefore we choose $\lambda_0' = 1$ as our optimized value.

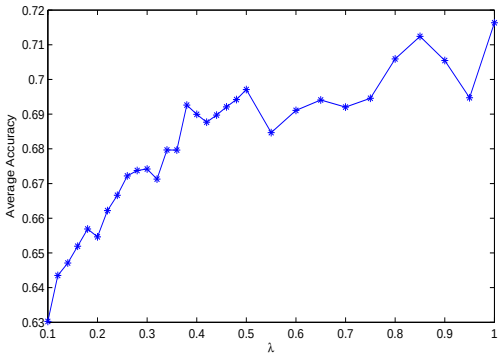

Fig. 5. Average Accuracy vs. λ.

3) Short Conclusion: From these experiments, we finally find the optimal triple $(\lambda_0', \eta_0', \mu_0')$. When (λ, η, μ) is $(1, -0.15, 0)$, our HMM with parameter-optimized k-means clustering can achieve highest average accuracy.

C. Results of our optimized model

After we have got the optimal value for the parameters we need, we conduct experiments to evaluate the performance of our model and compare our model with HMM with original k-means clustering. The number of clusters k in k-means is very important. Since we have no solid background about how many clusters is best for prediction, we vary k in our experiments and get the average accuracy under different value of k. We set the steps from 100 to 6000. The results are shown in Table I and Figure 6.

From the results, we can see that when k is below 1000, the average accuracy of both original model and our optimized model increases along with the number of clusters increasing. However, when k is larger than 1000, the average accuracy of both the two models no more increases. When k is 3000, the average accuracy of the original model is 78.0% and our optimized model is 83.5%. Our optimized model improves the average accuracy by 5.5%, which is really a big improvement in handwriting recognition.

V. Conclusion

In this work, we propose a hidden Markov model with parameter-optimized k-means clustering for handwriting characters recognition. We find two shortages of the original definition distance in k-means. We improve k-means clustering by considering the influence of neighbor pixels and different weights of pixels in different places. We conduct a series of experiments to compare our optimized model and original model. Results show that our optimized model improves the average accuracy from 78.0% to 83.5% when the number of clusters is 3000. So our model improves the average accuracy of HMM with k-means clustering for handwriting characters recognition.

TABLE I
AVERAGE ACCURACY OF ORIGINAL MODEL AND OPTIMIZED MODEL

Number of Clusters	100	500	1000	2000
Original Model	56.6%	72.9%	76.5%	77.9%
Optimized Model	71.5%	81.3%	82.8%	82.2%
Number of Clusters	3000	4000	5000	6000
Original Model	78.0%	77.7%	78.3%	78.6%
Optimized Model	83.5%	83.4%	83.3%	83.2%

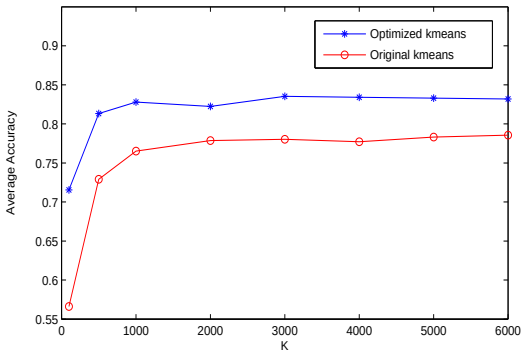

Fig. 6. average accuracy vs. number of clusters k

References

[1] Tappert, C. Charles. The state of the art in on-line handwriting recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990.

[2] O. Chapelle. Training a support vector machine in the primal, Neural Computation, 2007.

[3] L.R. Rabiner. A tutorial on hidden Markov models and selected applications inspeech recognition, Proceedings of the IEEE, 1989.

[4] N. Arica, F. Yarman-Vural. An overview of character recognition focused on off-line handwriting, IEEE transactions on systems, man and cybernetics - part C: Applications and reviews, 2001.

[5] S. Chevalier, E. Geoffrois, F. Preteux. A 2d dynamic programming approach for markov random field-based handwritten character recognition, IAPR International Conference on Image and Signal Processing, 2003.

[6] B. Taskar, C. Guestrin, D. Koller. Max-margin Markov networks, Neural Information Processing Systems, 2003.

[7] O.D. Trier, A.K. Jain, T. Taxt. Feature extraction methods for character recognition - a survey, Pattern Recognition, 1996.

[8] D. Guillevic, C.Y. Suen. HMM word recognition engine, Proceedings International Conference on Document Analysis and Recognition, 1997.

[9] J.B. MacQueen. Some methods for classification and analysis of multivariate observations, Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 1967.

[10] K. Jain, M.N. Murty, P.J. Flynn. Data clustering: a review, ACM computing surveys (CSUR), 1999.

[11] I.S. Dhillon, Y. Guan, B. Kulis. Kernel k-means: spectral clustering and normalized cuts, Proceedings of the tenth ACM SIGKDD, 2004.

[12] T. Jebara, Y. Song, K. Thadani. Spectral clustering and embedding with hidden Markov models, Machine Learning: ECML, 2007.