Printed Digits Classification

1. Introduction

Printed digits recognition has important usage such as recognizing telephone number, zip code and barcode. Classification methods based on learning from examples have been widely applied to digits recognition from the 1990s and have brought forth significant improvements of recognition accuracies. The methods include statistical methods, artificial neural networks, support vector machine (SVM), multiple classifier combination, etc. Over the last two decades, neural networks (NN) have been widely used to solve complex classification problems. They are very fast classifiers with relatively fast training time. However, a single NN often exhibits the over fitting behavior which results in a weak generalization performance when trained on a limited set of training data. On the other hand, there is a consensus in machine learning community that SVMs are most promising classifiers due to their excellent generalization performance. Curse of dimensionality and over fitting in NNs, seldom occur in SVMs. However, in terms of running time, SVMs for multi-classes classification problems are slower than neural networks and their training on a large data set is still a bottle-neck. In addition, the performance of SVMs largely depends of the choice of kernels and also that multi-class SVM classifier is still an open problem.

This experiment presents a hybrid algorithm for training RBF network based on K-means and SOM. The algorithm consists of a proposed clustering algorithm to position the RBF center and givens least squares to estimate the weights. The aim of this experiment is to recognize printed digits (1-4) using the hybrid model. In the meanwhile, KNN and MLP with Scaled Conjugate Gradient will be implemented in order to show the comparative of different models according to the experiments.

2. Models and Algorithms

2.1. Models

Radial basis function (RBF) networks belong to one of major neural networks, and draw many researchers' attention because of good performance in many application fields. Radial basis function networks typically have two distinct layers as shown in figure 1. The right layer consists of a set of basis functions each of which is a function of the distance between an input pattern and a prototype pattern.

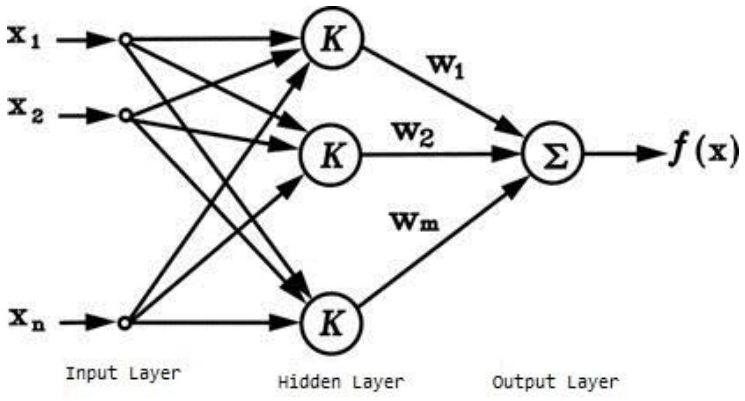


Figure 1 RBF Structure

A standard choice of basis function is the Gaussian:

$$\phi_j(\boldsymbol{x}) = exp\{-\frac{||\boldsymbol{x} - \boldsymbol{\mu}_j||^2}{2\sigma_j^2}\}$$

Radial basis function makes an approximation based on training data, and Gaussian function is used mostly as the radial basis function. In order to train RBF networks first we should find appropriate centre and radius of radial basis function. For this task, we may use some unsupervised learning algorithms like k-means clustering and Self-organization Map to converge the centers.

CVQ

The basic idea of CVQ is to classify each vector into o ne o f several categories according to its characteristics and then a set o f separate sub codebooks is designed for each category . T he computational complexity is also reduced since only o ne appropriate sub codebook which is much smaller than the overall codebook is searched for each input vector.

As a comparison, the k-nearest neighbor (KNN) and multiple layer perceptron with scaled Conjugate Gradient algorithm are implemented, and their accuracy and running time are recorded.

KNN, which is a type of instance-based learning, is used for classifying objects based on closest training examples in the feature space. In the classification phase, ***k*** is a predefined constant, and an unlabeled vector is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

MLP with SCG uses a numerical approximation for Hessian matrix; it avoids instability by combining the model-trust region approach from the Levenberg-Marquardt algorithm with the conjugate gradient approach. This allows scaled conjugate gradient to compute the optimal step

size in the search direction without having to perform the computationally expensive line search used by the traditional conjugate gradient algorithm.

2.2. Algorithms

Finding Centers

First, we use unsupervised learning method k-means and SOM to find the centers of the hidden nodes, and then calculate the variance σ of the hidden nodes according to the distance among each data centers.

K-means clustering algorithm for centers has four steps:
Step 1: Initi K clusters randomly, then each data points will have c cluster label.
Step 2: Calculate the mean of each cluster C.
Step 3: Find similarity matching of the centers, $K(x) = \arg\ \min_k \| x(n) - t_k(n) \|$. If then minimum distance is not the value for this data point to its cluster center, then the cluster identity of this data will be adjust and update to the one that gives the minimum distance.if $k = k(x), t_k(n+1) = t_k(n) + \eta[x(n) - t_k(n)]$, otherwise $t_k(n+1) = t_k(n)$.
Step 4: Continue step1~4 until no changes of the center occur.

SOM algorithm for centers has six steps:
Step1: Set map size and randomize the map's nodes' weight vectors
Step 2: Grab an input vector
Step 3: Traverse each node in the map and use Euclidean distance formula to find similarity between map's nodes' weight vector.
Step 4: Track the nodes that produces the smallest distance ($d_{j^*} = \arg\ \min_d \| x(j) - w_{ij} \|$)
Step 5: Update the nodes is the neighborhood of C by pulling then closer to the input vector $w_{ij}(t+1) = w_{ij}(t) + \eta(j, j^*)(x_i - w_{ij})$.
Step 6: Continue step1~5 until no changes of the center occur.

Training

In this report, according to the empirical evidence, the error caused by speed is not so obvious. $\sigma^2 = 500$, not dynamic decided by the distance of each nodes. centers has been calculated in the finding center section, we consider the centers as perception and input the train dataset again to decide the weights ***w*** of the output nodes by pseudo-inverse. The training algorithm is as follow:

1. Calculate the outputs from each elements with Gaussian basis functions:$\varphi_{ij} = \exp(-\| xi - Cj \| 2/2\sigma 2)$ at each ***j***th center, where i-1,2,...n;j-1,2,...N.
2. Compute the correlation matrix and estimate the weight
   $y_i = f(X_i)\ W = (w_1, w_2, w_3, \ldots., w_n)^T$
   $y = t^T = (y_1, y_2, \ldots, y_N)^T, y = HW$
   Weight $W = H^{-1}y$

Testing

Due to the cross validation, input dataset has been divided into to train part and test part, in the train section above, train part dataset has been used and in this testing part, test part dataset will be used to test the network and evaluate the accuracy and running time of CPU. The algorithm of test is as follow:

1. Calculate the outputs from each elements with Gaussian basis functions:$\varphi_{tj} = \exp(-\| x_t - C_j \|^2/2\sigma^2)$ at each jth center, x_t is the vector of test part dataset.where i=1,2,…n; t=1,2,…N.

2. Normalize the outputs, and then compare with test part dataset to get the number of the correct items NC. Evaluation function would be use to get the accuracy of this network.

   $\text{Accuracy} = \frac{\text{NC}}{\text{number of test}}\%$.

3. Experiment

This experiment is conducted by Matlab 2011b on Windows 7 32 bits operation system .The computer is Intel Duo Core 2.2GHZ and 4G RAM.

Data set

Pre-processed digit images are stored into a 3961 x 600 matrix. Every image is one of the 1~4.There are 3961 samples,1~972 samples is for 1,973~1976 samples is for 2,1977~2975 sample is for 3 and the rest is for 4.They are 24.5%,25.4%,25.2% and 24.9% respectively. In the experiments, we consider the samples are divided equally for 25%.In order to improve the accuracy of the reorganization of the printed digits, we convert the target vector into 3961x4,1 is 1000,and 2 is 0100,3 is 0010 and 4 is 0001.

digits	Number of Samples	Rate	Dimension
1	972	24.5%	600
2	1004	25.4%	600
3	999	25.2%	600
4	987	24.9%	600

Table 1 Dataset of the experiment

Experiment Description

In the experiment,3961 images for 1~4 are portioned into a training subset and the rest images of the original dataset are in the test set, training and test set are 80% and 20% respectively.

Procedure for RBF with K-means

1. Loading digits.mat .
2. Divided the training dataset (X) and target dataset(Y) into 20% and 80%.20% is for training, and 80% is for the test process.
3. Input ***xtrain*** into the ***kmeans*** with ***gaussian*** method to find the centers of the hidden nodes, then we get ***ctrs*** matrix.
4. Input ***xtrain*** again and speed into $h_{ij} = \Phi_j(\| X_i - c_j \|, \Phi_j(.)$ is the activation function of the hidden node,$C_j = X_j$ is the data center of the RBF function.we define the output matrix of RBF network is $H = [h_{ij}]$, $H \epsilon R^{NxN}$, the output o f RBF network is
   a) $F(X_i = \sum_{j=1}^{N} h_{ij} w_j = \sum_{j=1}^{N} w_j \Phi(\| x_i - c_j \|))$
   b) Set $y_i = f(X_i)$ $W = (w_1, w_2, w_3, \ldots., w_n)^T$
   c) So $y = t^T = (y_1, y_2, \ldots, y_N)^T$,$y = HW$
   d) Weight $W = H^{-1}y$
5. After we calculate the weight W, ***xtest*** will input the network to get the corresponding target vector, and then we use an evaluation function to get the accuracy of the model. Meanwhile, cputime is recorded.
6. Repeat step 1~5 for 100 times and record the average results.

Procedure for RBF with SOM

1. Loading digits.mat .
2. Divided the training dataset (X) and target dataset(Y) into 20% and 80%.20% is for training, and 80% is for the test process.
3. Set Mapsieze ***map_l*** 10x10,and use the ***som_make*** method of somtoolbox to generate the SOM map sM.
4. The center of RBF is the ***sM.codebook,ctrs = sM.codebook***.
5. Input ***xtrain*** again and speed into $h_{ij} = \Phi_j(\| X_i - c_j \|, \Phi_j(.)$ is the activation function of the hidden node,$C_j = X_j$ is the data center of the RBF function. We define the output matrix of RBF network is $H = [h_{ij}]$, $H \epsilon R^{NxN}$, the output o f RBF network is
   a) $F(X_i = \sum_{j=1}^{N} h_{ij} w_j = \sum_{j=1}^{N} w_j \Phi(\| x_i - c_j \|))$
   b) Set $y_i = f(X_i)$ $W = (w_1, w_2, w_3, \ldots., w_n)^T$
   c) So $y = t^T = (y_1, y_2, \ldots, y_N)^T$,$y = HW$
   d) Weight $W = H^{-1}y$
6. After we calculate the weight W, ***xtest*** will input the network to get the corresponding target vector, and then we use an evaluation function to get the accuracy of the model. Meanwhile, running time is recorded.
7. Repeat step 1~6 for 100 times and record the average results.

Experimental Results

Upon completion of the experiment procedures, the results are gathered and shown here. Interpretation of the following results will be discussed in conclusion of this study.

Accuracy and running time

Table 2 shows that the accuracy, running time and loops of the four models. Obviously, we can find that KNN has the highest accuracy and least running time. Accuracy of RBF with k-means and RBF with SOM are almost the same, however, the RBF with k-means network costs 3 times running time than RBF with SOM.MLP with SCG also performs a good classification rate with 89.52%, which is 2.5% higher than RBF with K-means.

Models	Average Accuracy	Average CPU time(s)	Loops
RBF with K-means (k=100)	86.98%	202.3489	100
RBF with SOM (map=10x10)	86.34%	26.3623	100
KNN(k=5)	93.43%	1.5756	100
MLP with SCG (hidden node=100)	89.52%	93.2371	100

Table 2 Accuracy and running time of Tested models

Converge Vectors

In the experiment, converge vector of k-means is displayed in figure 4 with 100 digits in size 300x200, and converge vector of SOM is displayed in figure 5 in the same format with figure4.Feature of figure 4 is quite different from figure5; the sort order of figure 5 is in order.

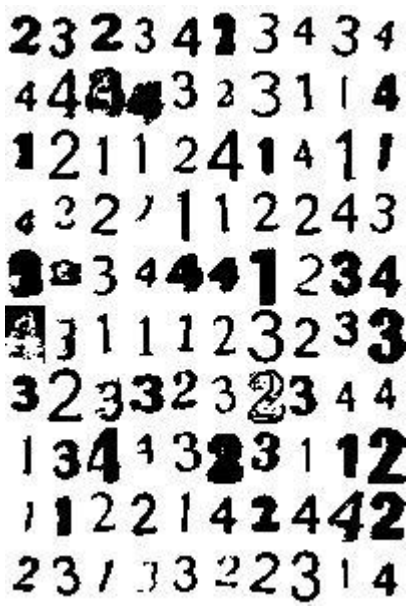


Figure2 RBF with K-means

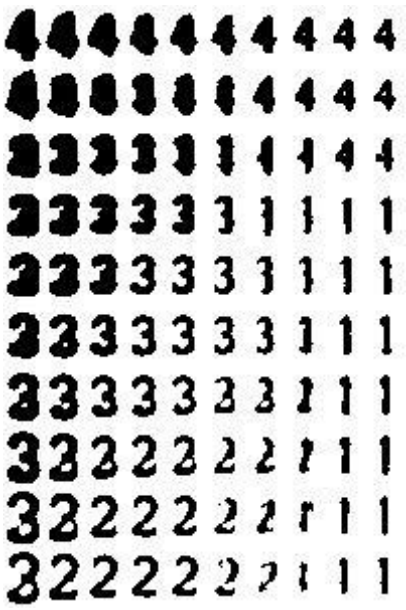


Figure3 RBF with SOM

In the experiment of RBF with K-means, we test k-means with different centers, the range of centers are 100~1000,however,when the number of centers more than 800,matlab crash and stop running, table 3 shows the accuracies and running time.

Centers	Average Accuracy	Average running time(s)
100	86.98%	202.3489
200	88.90%	313.8428
300	89.65%	360.5807
400	90.97%	397.7917
500	92.55%	454.7741
600	93.31%	492.1676
700	93.56%	455.2577
800	Crash	

Table 3 Accuracy and Running time of RBF with K-means

Figure 4 shows the plots of RBF with K-means corresponding to the data of table3.It can be found that accuracies and running time of RNF with K-mans are almost have positive correlation with the number of centers. A special case is that when the number of centers is 700, running time decrease anomalously. We cannot check the reason of decrease, due to the crash of following tests.

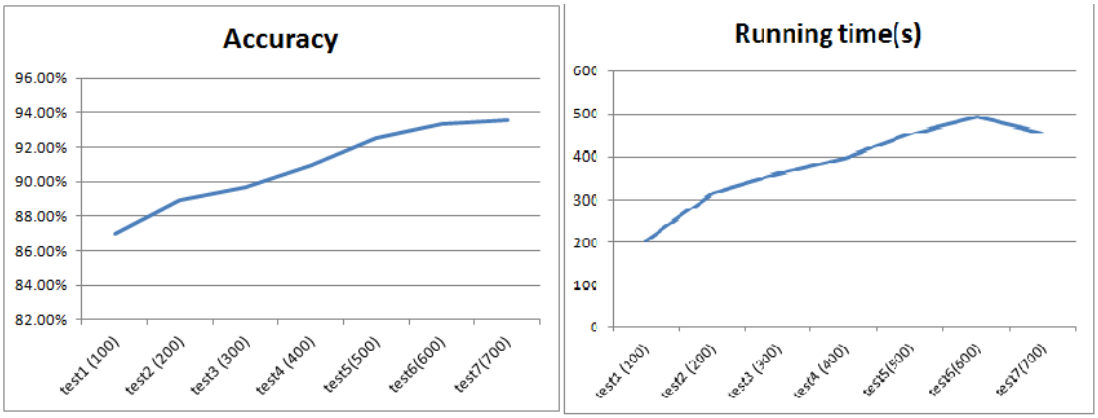


Figure 4 Plots of RBF with K-means

Table 4 shows that accuracy, running time, quantization error and topographic error of RBF with SOM. Map sizes are 10x10, 14x14, 17x17, 20x20, 22x22, 24x24, 26x26, 28x28, 30x30 and 32x32. That is, rang of the map are 100~1000. We can use the data to compare to RBF with K-means.

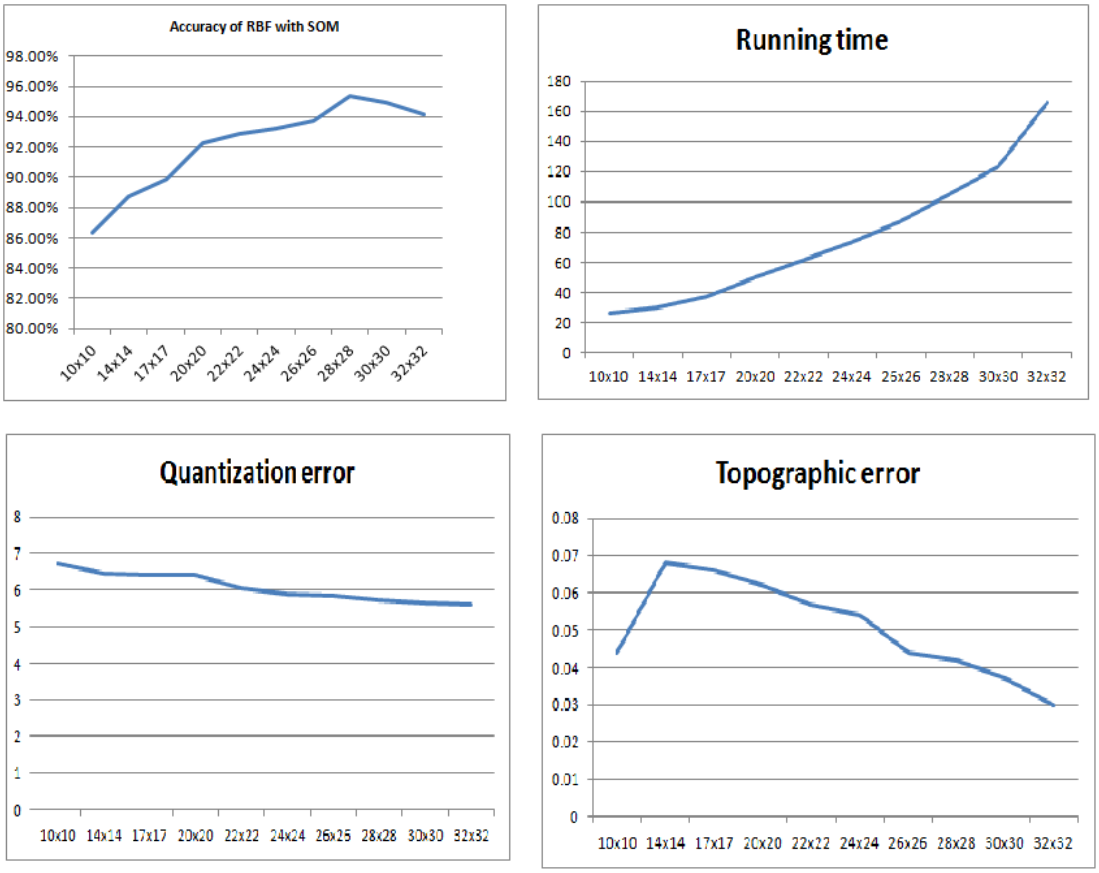
Map Size	Accuracy	Running time	Quantization error	Topographic error
10x10	**86.34%**	**26.3623**	**6.730**	**0.044**
14x14	**88.76%**	**30.1862**	**6.436**	**0.068**
17x17	**89.90%**	**37.2218**	**6.423**	**0.066**
20x20	**92.24%**	**49.6785**	**6.398**	**0.062**
22x22	**92.87%**	**61.7452**	**6.054**	**0.057**
24x24	**93.20%**	**73.5779**	**5.907**	**0.054**
26x26	**93.69%**	**86.994**	**5.841**	**0.044**
28x28	**95.33%**	**104.9809**	**5.721**	**0.042**
30x30	**94.95%**	**123.0068**	**5.658**	**0.037**

32x32	**94.13%**	**165.6731**	**5.592**	**0.030**

Table 4 Accuracy, running time, Quantization error and topographic error of RBF with SOM

From figure5,we can see that accuracies of RBF with SOM increase with the increasing map size, until map size is 28x28.The highest accuracy is 95.32%.After this, the accuracy would not increase. Obviously running time of RBF with SOM has positive correlation with the map size.

Quantization error decreases with the increasing ma size. For topographic error, when map size is 10 x 10, the error is low. However, when map size is 14 x 14, the topographic error is the highest. The reason is that SOM can simulate the topology structure, and topology structure will change based on the map size. When map size is 10x 10, the structure of topology is simple, so the error is low. When the map size is 14 x14, the structure become complex and error rate increase more than increased rate of the map size. As the map size becomes larger, the rate of topographic error less then rate of maps size. The topographic error decrease naturally.



Accuracy,running time,quantization error and topographic error of RBF with SOM

Figure 5 Plots of RBF with SOM

4. Conclusion and Future Planning

This report proposes a hybrid classification of printed digits based on RBF, k-means and SOM. It is known that RBF networks are one of the most successful data mining or machine learning tools for classification. But, in this experiment, KNN and MLP with SCG are better than RBF. Obtained classification time and recognition rates of KNN better than the recognition time and the recognition rate of RBF with k-means, RBF-SOM, and MLP with SCG applied on the same dataset set. The reason for this situation is KNN has a good efficiency on classification when data is clipped and sorted in order, especially for the large datasets. Advantage of RBF networks based hybrid algorithms cannot perform in this situation. In this experiment, when centers are the same, RBF with SOM has a better classification rate and cost less CPU space than RBF with K-means. It is also proved that RBF with k-means need more powerful CPU and time to classify the digits. Another point is that RBF may not always be the best predictors due to the fact that they are

trained based on some clustering algorithms with different kinds of data sets .Setting appropriate number of clusters for best accuracy is a choice. But, determining the appropriate number of clusters is arbitrary in nature, so we incremented the number of clusters progressively to find some better RBF networks of accuracy in systematic manner. In this experiment, we find the best accuracy of RBF with SOM and RBF with K-means are 95.33% and 93.56% respectively.

5. References

[1].Bailing Zhang." *Handwritted Digit Recignition by Neual 'Gas' Model and Population Decoding*", University of Newvastle,Callaghan,Australia,1998.

[2]. Dejan Gorgevik, Dusan Cakmakov.*" An Efficient Three-Stage Classifier for Handwritten Digit Recognition", University "Sv. Kiril i Metodij"*, Faculty of Electrical Eng., Department of Computer and Information Technology,2004.

[3].M.Y.Mashor."*Hybrid Training Algorithm for RBF Network***"**,University Science of Malaysia,Perak Branch Campus,Malaysia,1999.

[4]. Bi Ran, "*Hand Written Digit Recognition and Its Improvement*",http:// cs2306-machine-learning.googlecode.com/svn,2009.

[5].Neil Assdrin,Andrew Smith and Doug Turnbull,"*Classifying Facial Expression with Radial Basis Function Networks ,using Gradient Descent and K-means*",Department of Computere Science,University of California,San Diego,USA,2003.

[6].Teuvo Kohonen,"*The Self-Organizing Map*",Vol.78,No.9,September 1990,IEEE.

[7]. Daniel Cruces Álvarez, Fernando Martín Rodríguez, Xulio Fernández Hermida." *Printed and Handwritten Digits Recognition Using Neural Networks*", Departamento de Tecnolog ías de las Comunicaciones. Universidad de Vigo, 1998.

[8]. Gaurav Jain, Jason Ko ,"*Handwritten Digits Recognition*", University of Toronto,2008.