L7: Artificial Intelligence
(ACCA7026)

MNIST Handwritten Digits Recognition

MSc Data and Artificial Intelligence

Student ID: 2106353

Name: Chenghui He

Wales Institute for Science and Art (WISA)

University of Wales Trinity Saint David

January 10th, 2022

MNIST Handwritten Digits Recognition

Abstract

Handwritten digits recognition using machine learning techniques may be an effective approach to get a better knowledge of machine learning algorithms. I mostly employ KNN (k-nearest neighbours) and SVM (support vector machine) for recognition in this study using the MNIST dataset (Modified National Institute of Standards and Technology database). By tweaking the K values in KNN and choosing kernel functions in SVM, KNN achieves 97.06% accuracy while SVM obtains 98.08% accuracy.

1. Introduction

Handwritten Number Recognition is a subset of optical character recognition (OCR) [1]. It is an investigation into the use of electronic computers to recognise Arabic numbers written by hand on paper mechanically. Handwritten digit recognition technology combines elements of pattern recognition and image processing, artificial intelligence, formal languages and automata, statistical decision theory, fuzzy mathematics, combinatorial mathematics, information theory, and computing; it also incorporates elements of linguistics and phonetics, psychology, and biology.

Handwritten number recognition, as a subfield of optical character recognition technology, is largely concerned with how to utilise computers to automatically recognise Arabic numerals written by hand on paper. It may be classified into offline and online handwritten numeral recognition. The former does not require a special input device, but instead requires capturing the written text into the local machine and processing the image, which consumes a small amount of local space and enables rapid recognition, but the recognition accuracy is lower due to the lack of information such as the order of the strokes; whereas the latter requires real-time recognition of the written numerals and requires passing through a special input device, which can be quite expensive. The latter is a real-time recognition of written numbers that needs a specialised input device capable of storing information such as the stroke order. In comparison to the former, the accuracy of identification is greater due to the increased amount of information acquired, but the recognition speed is slower. The impact of handwritten numeric recognition technology on the growth of numerous sectors is expanding at the moment. Bank notes, financial records, and postal codes may all be recognised via handwriting body digital recognition, increasing efficiency and economising on human, material, and financial resources. Simultaneously, because to

the relative simplicity of the implementation of handwritten number recognition, it may be used as a benchmark for more difficult issues and techniques. [2].

Because handwritten digit recognition has no context and no semantic relevance, and because some sectors have relatively high requirements for digits, such as bank statements, postal codes, financial statements, and other similar documents, it is also necessary to achieve a higher level of accuracy for its recognition in order to be effective. Numerous research scholars at home and abroad are working to further improve the recognition rate and generality of numerical recognition, which has broad application prospects and practical value. It has important applications in the fields of literature retrieval, office automation, automatic postal sorting, automatic bank note recognition, among other fields. Furthermore, as more and more intelligent information processing systems and terminals are equipped with electronic camera or scanning equipment, digital recognition can be widely applied to the input of such equipment as well as the signature identification of electronic commerce systems, providing significant theoretical significance as well as broad application prospects for the automation of text information processing and the further enhancement of intelligent computer inputation.

The fundamental premise of handwritten digit recognition is to pattern match the input sample digits with the matching standard sample digits and then select the sample digit that is the most similar to the input sample digits as the recognition result. The selection of the sample feature extraction technique and the design of the classifier are critical components of the overall recognition process. The classification effect and performance of the recognition system are determined by the speed and comprehensiveness of the feature extraction technique, as well as the efficiency and accuracy of the classifier design.

Accuracy has been shown for machine learning techniques to handwritten digit recognition[3]. Yann LeCun et al. conducted a comprehensive experimental study [4] in which they compared the performance of linear and polynomial classifiers, the K-nearest neighbour classifier, a fully connected multilayer neural network, and three different types of convolutional neural networks, including LeNet 1, LeNet 4, and LeNet 5. On the MNIST data set, Boosted LeNet 4, a combination of three LeNet 4, has a best test error rate of 0.7%.

I mostly employ KNN (k-nearest neighbours) and SVM (support vector machine) for recognition in this study using the MNIST dataset (Modified National Institute of Standards and Technology database).

2. MNIST Data Set

The MNIST dataset is a subset of the NIST dataset (Modified National Institute of Standards and Technology database). MNIST comprises a total of 70,000 handwritten digit pictures (60,000 for the training set and 10,000 for the test set), each with a 28x28 pixel bounding box and anti-aliasing. Each of these photos has a matching Y value that indicates the digit[5]. Figure 1 shows the first nine images.

Figure 1 Sample images of the MNIST data set

3. Classifiers

In this section, the two classifiers covered in class are selected for introduction and study.

3.1 Support Vector Machine

Many people use Support Vector Machines (SVM) because they achieve great accuracy while using minimal processing power[6]. Support Vector Machines, abbreviated as SVMs, are a kind of machine learning algorithm that may be used for regression and classification problems. Nonetheless, it is often used to categorise aims.

The Support Vector Machine algorithm's purpose is to find a hyperplane in an N-dimensional space (the number of features) that classifies the data points distinctively.This procedure is shown in Figure 2.

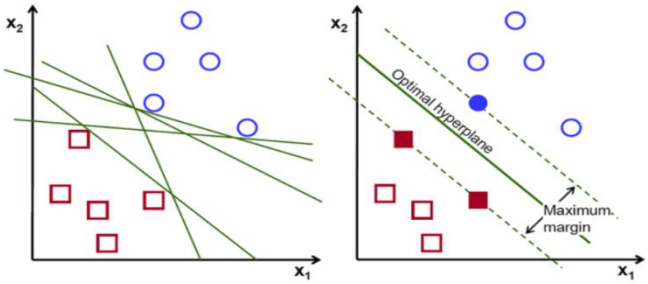

Figure 2 Possible hyperplanes

There are several potential hyperplanes for separating the two groups of data points. Our objective is to identify a plane with the greatest possible margin, that is, the greatest possible distance between the data points for the two categories. The maximisation of the marginal distance gives some reinforcement, allowing for more confident classification of subsequent data points.

The hyperplane is a decision boundary that aids in data point classification. Classification of data points on either side of the hyperplane is possible. Additionally, the hyperplane's dimension is dependent on the number of features. If there are just two input characteristics, the hyperplane is simply a line. When the number of input characteristics is increased to three, the hyperplane transforms into a two-dimensional

plane. When the number of characteristics reaches three, imagination gets tough.
Support vectors are data points that are relatively near to the hyperplane and have an effect on the hyperplane's location and orientation. By using these support vectors, we may maximise the classifier's margin. By removing the support vectors, the hyperplane's location is altered. These are the points that aid in the construction of the SVM.
Support vector machine methods are a subset of supervised machine learning techniques that utilise a range of technical concepts such as maximally spaced hyperplanes, Mercer kernels, convex quadratic programming, and relaxation variables to solve binary classification problems. The interval maximisation approach may be used to classify data that are linearly separable.
The precise content is as follows, assuming that a given training sample set has n samples:

$$\mathrm{D} = \{(x_1, y_1), (x_2, y_2).....(x_i, y_i)\}, y_i \in \{-1, +1\}$$

where x_i represents the i sample consisting of m-dimensional features, and y_i represents the classification result of the i sample.The basic idea of SVM is to find a hyperplane $w^T x + b = 0$ in the sample space of a known training sample set D.The hyperplane maximises the distance from the support vector (i.e. the training sample closest to the hyperplane) to the hyperplane by separating the different samples, and has the above characteristics.A hyperplane with these characteristics is known as a "maximum spacing" hyperplane. The hyperplane that finds this maximum interval is the one that satisfies y_i ($w^T x + b = 0$) ≥ 1 bounded by w and b and such that $\frac{2}{\| w \|}$ is maximum.
When the original sample is unidentifiable linearly, an appropriate kernel function may be used to map the sample feature space to a higher dimensional sample feature space, so making the sample linearly distinct in the higher dimensional feature space. Because the kernel function has an effect on the classification accuracy of an SVM, choosing an appropriate kernel function is a critical part of applying SVM techniques. Three popular kernel functions are as follows:

Linear Kernel: $K(x, x') = x \cdot x'$

Polynomial Kernel: $K(x, x') = (1 + x \cdot x')^p$

Gaussian Kernel: $K(x, x') = \exp(-\gamma \| x - x' \|^2)$

Since SVM is a binary classification machine learning algorithm, i.e. the classification result is either A or B. However, in real life, it is mostly a multi-category application scenario. To solve the conversion from binary classification to multi-category, there are two common methods, One to One (One vs One, OVO) and One to Rest (One vs

Rest, OVR). The OVO algorithm solves the problem by assuming that there are n categories of classification tasks, and combining these n categories two by two will result in n(n-1)*2 binary classification tasks. During the OVR algorithm selects one class as the positive example and all other classes as the negative example for each training session to train N classifiers. At test time, if only one classifier predicts a positive class, the corresponding class is marked as the final classification result.

SVM advantages:

(1)By maximising the interval of the tangent plane, support vector machines eliminate dependency on data size and distribution, resulting in superior performance for machine learning tasks involving small samples.

(2)SVM converts the low-dimensional feature dimensional space to a high-dimensional dimensional space, which, although seeming to be more computationally expensive, converts the classification task's non-linear issue to a linear one. By defining a kernel function using a minimal number of support vectors, not only is the dimensional disaster avoided, but also the sample dimensionality is divorced from the algorithm's complexity.

(3)While conventional approaches may introduce issues such as overfitting and local minima, the SVM optimization method potentially produces a globally optimum solution. Additionally, SVM outperforms other learning algorithms in terms of resilience, learning ability, and generalisation ability.

3.2 K-Nearest Neighbor

K-Nearest Neighbor (KNN) is a statistical classification approach first presented by Cover and Hart in 1968 as the simplest machine learning method. basic machine learning approach, the study has been reasonably mature theoretically[7].

The KNN method performs very well for text classification tasks such as information retrieval, machine translation, automated digesting, information filtering, and email categorization. Text classification is also frequently employed in search engines, and the KNN algorithm is capable of doing regression as well as classification. The characteristics of a sample may be determined by identifying the sample's K closest neighbours and giving the average of their qualities to the sample. A more helpful technique is to weight the effects of neighbours at various distances on that sample differently, with the weights proportionate to the distance. Additionally, the KNN algorithm may be used to provide current user product suggestions based on what the user's closest neighbours have purchased.

The KNN algorithm's fundamental premise is that if a sample belongs to a category and the majority of its K closest neighbours in the feature space also belong to that category, then that sample likewise belongs to that category and exhibits the characteristics of samples in that category. When it comes to category judgments, the KNN approach uses a relatively small number of neighbouring samples.

For each point in a dataset with unknown class properties, the method executes the following procedures in sequence:

(1) Calculation of the distance between the point in the dataset of a known category

and the current point.
(2) Sorted in ascending order of distance.
(3) Selection of the k points with the smallest distance from the current point.
(4) Determine the frequency of occurrence of the category in which the first k points are located.
(5) Return the category with the highest frequency in the first k points as the predicted classification for the current point

Euclidean distance calculation:

The Euclidean distance between two points A(x_1 , y_1) and B(x_2 , y_2) on the two-dimensional plane is:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The Euclidean distance between two points A(x_1 , y_1 , z_1) and B(x_2 , y_2 , z_2) on the three-dimensional plane is:

$$D = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

The Euclidean distance between two N-dimensional vectors A(x_{11} , x_{12} ,..., x_{1n}) ,B(x_{21} , x_{22} ,..., x_{2n}) is:

$$D = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2}$$

4. Experimental Results

In this section, the SVM algorithm and KNN algorithm are tested on matlab against the MNIST dataset. Different amounts of data are used and the performance of the two algorithms is analysed using accuracy and confusion matrices.

4.1 Evaluation of SVM

LIBSVM[8] is a simple, easy-to-use and fast and effective package for SVM pattern recognition and regression developed by Associate Professor Lin Chih-Jen and others at National Taiwan University, which provides not only compiled executable files for the Windows family of systems, but also source code for easy improvement, modification and application on other operating systems. The software is also characterised by the relatively small number of parameter adjustments involved in SVM, providing a number of default parameters with which many problems can be solved; and by the provision of interaction tests (Cross-SVM regression and other problems, including multi-class pattern recognition problems based on one-to-one

algorithms. This means that the optimal SVM algorithm parameters can only be selected empirically, by experimental comparison, by extensive search, or by using the cross-check function provided by the software package to find the optimal function.

I use libsvm() function on matlab to train and test the SVM classifiers. Three different Kernel functions, such as linear kernels, polynomial kernels, and Gaussian kernels, were tested separately. Being interested in the impact of data volume on the performance of the algorithm, I tested it separately for 30,000 and 60,000 data volumes and analyzed the impact. The results of the experiments are shown in Table 1 and show the corresponding confusion matrix and give individual error cases.

Table 1 Summary for the experimental results of SVM.

Kernel	Amounts of data	Accuracy	Time(minutes)
Linear	30000	93.36%	1.02
	60000	93.98%	1.78
Polynomial(degree=2)	30000	97.8%	2.33
	60000	98.08%	5.5
Gaussian	30000	93.64%	5.87
	60000	94.46%	13.94

For Gaussian kernels, the parameter γ I have used the default value(γ =1) from matlab.

The distribution of the error cases can be clearly seen in the confusion matrix(Figure 3,4,5). From Figure 6, it can be seen that the reason for the misclassification is that the handwritten number 5 is very similar in shape to the number 6, which leads to the misclassification.

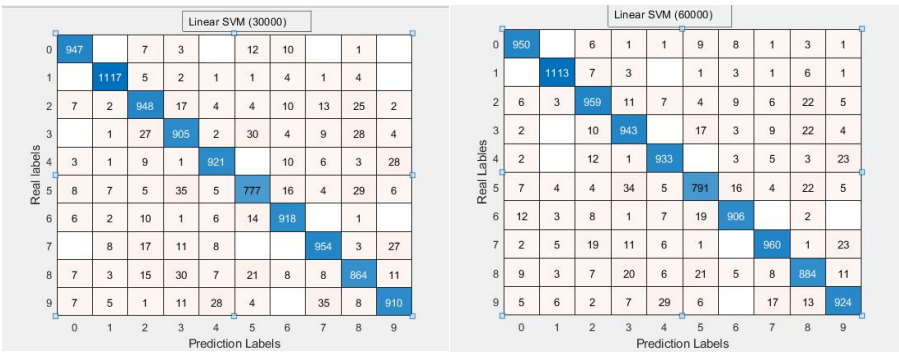

Figure 3 Linear SVM

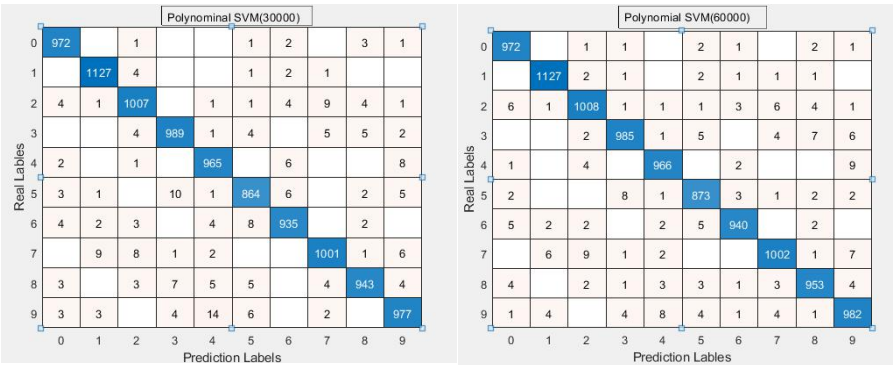

Figure 4 Polynominal(degree=2) SVM

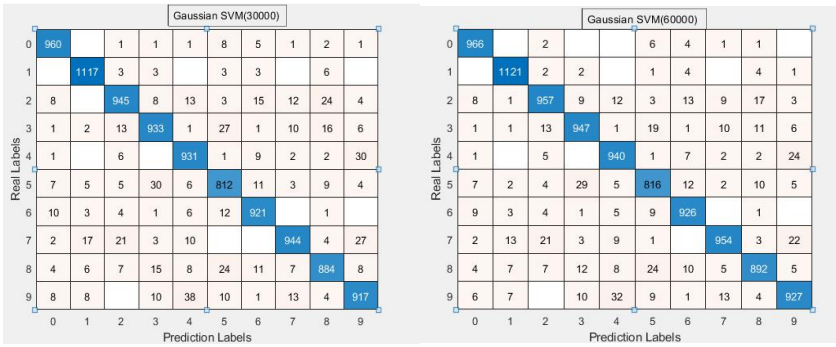

Figure 5 Gaussian SVM

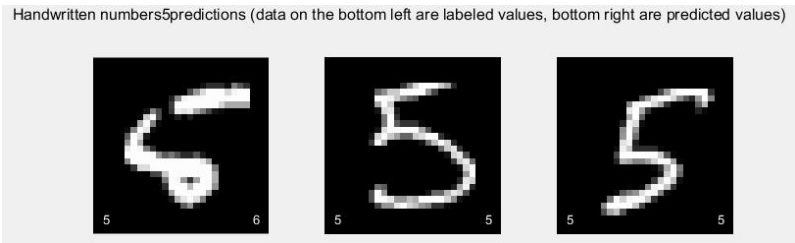

Figure 6 Handwritten numbers 5 prediction in SVM

4.2 Evaluation of KNN

I use fictknn() function on matlab to train and test the KNN classifiers. For the KNN classifier, I chose the Euclidean distance metric to compute the K nearest neighbours. I tried different values of K to perform the test and the results are shown in Table 2 and show the corresponding confusion matrix(Figure 8).

Table 2 Summary for the experimental results of KNN

K value	Accuracy	Time(minutes)
K=1	96.91%	2.56
K=3	97.06%	2.61
K=7	96.90%	2.61

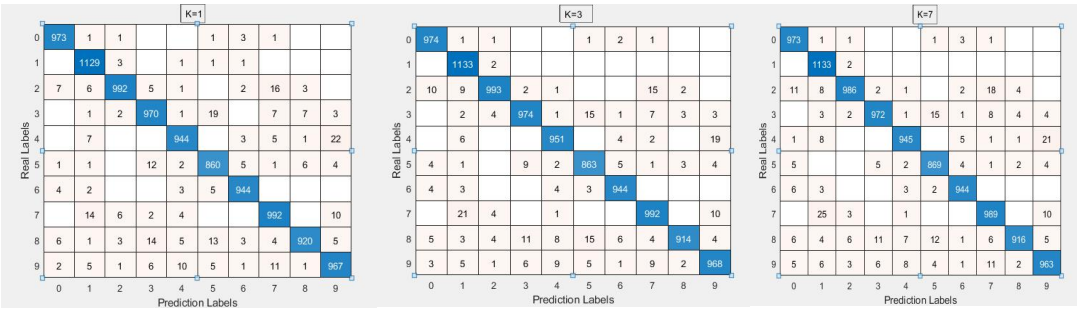

Figure 7 KNN(K=1,3,7)

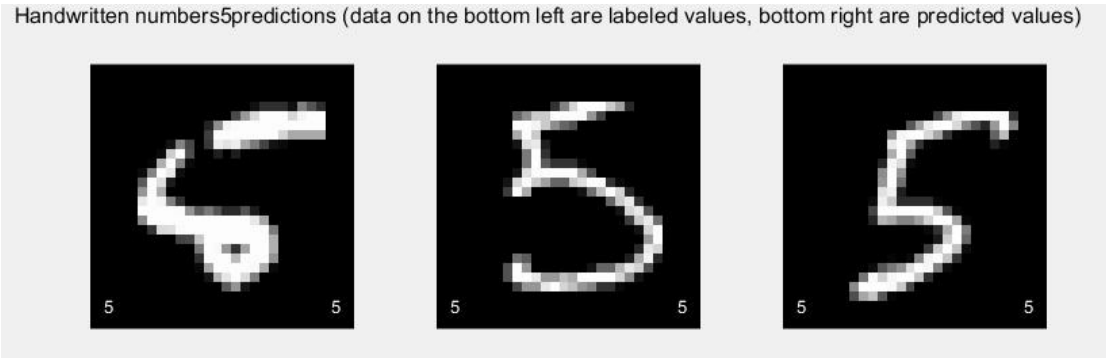

Figure 8 Handwritten numbers 5 prediction in KNN(K=3)

5. Conclusions

For SVM, non-linear kernels (i.e. polynomial and Gaussian kernels) take longer to train than linear kernels because the kernel product of each support vector needs to be evaluated, but better accuracy can be achieved for non-linear classification tasks. By comparing the 3 kernels, the polynomial kernel(degree=2) gives the best accuracy(98.08%), although the test time is the longest.

For KNN, the accuracy of different classifications can be obtained by adjusting the value of K, but the accuracy improvement is limited. In the (K=1, 3, 7) example, the best accuracy(97.06%) is obtained when K=3. The test time is short and the value of K has a small effect on the test time.

For the amount of data, increasing the amount of data can improve the accuracy but requires longer training and testing time.

For KNN and SVM, in this paper, KNN is simple to operate, requires only modifications to the k-values and has a shorter overall time. SVM has more hyperparameters and is more theoretical. By adjusting the hyperparameters as well as the kernel in a reasonable way, better classification results can be obtained. Ultimately, the best accuracy(98.08%) is obtained in SVM with a polynomial kernel.

References

[1] GovindanVK，Character recognition--a review． Pattern Recognition,1991,23(7)：671～683

[2] Zhang J . Comparison of Classification Algorithms for Handwritten Digit Recognition.

[3] Mitchell T M . Machine Learning[M]. McGraw-Hill, 2003.

[4] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, E. Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In International conference on artificial neural networks, volume 60, 1995.

[5] MNIST Data Set. http://yann.lecun.com/exdb/mnist/

[6] Miao S, Wang Z J, Liao R. A CNN regression approach for real-time 2D/3D registration[J].IEEE transactions on medical imaging, 2016, 35(5): 1352-1363.

[7] Zhang M L , Zhou Z H. ML-KNN: A lazy learning approach to multi-label learning[J]. Pattern Recognition, 2007, 40(7):2038-2048.

[8] LIBSVM. https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Appendix

```
function images = loadMNISTImages(filename)
%loadMNISTImages returns a 28x28x[number of MNIST images] matrix containing
%the raw MNIST images
fp = fopen(filename, 'rb');
assert(fp ~= -1, ['Could not open ', filename, '']);
magic = fread(fp, 1, 'int32', 0, 'ieee-be');
assert(magic == 2051, ['Bad magic number in ', filename, '']);
numImages = fread(fp, 1, 'int32', 0, 'ieee-be');
numRows = fread(fp, 1, 'int32', 0, 'ieee-be');
numCols = fread(fp, 1, 'int32', 0, 'ieee-be');
images = fread(fp, inf, 'unsigned char');
images = reshape(images, numCols, numRows, numImages);
images = permute(images,[2 1 3]);
fclose(fp);
% Reshape to #pixels x #examples
images = reshape(images, size(images, 1) * size(images, 2), size(images, 3));
% Convert to double and rescale to [0,1]
images = double(images) / 255;
End

function labels = loadMNISTLabels(filename)
%loadMNISTLabels returns a [number of MNIST images]x1 matrix containing
%the labels for the MNIST images
fp = fopen(filename, 'rb');
assert(fp ~= -1, ['Could not open ', filename, '']);
magic = fread(fp, 1, 'int32', 0, 'ieee-be');
assert(magic == 2049, ['Bad magic number in ', filename, '']);
numLabels = fread(fp, 1, 'int32', 0, 'ieee-be');
labels = fread(fp, inf, 'unsigned char');
assert(size(labels,1) == numLabels, 'Mismatch in label count');
fclose(fp);
End
% SVM linear_kernel
% load training set and testing set
clear all;
train_set = loadMNISTImages('train-images.idx3-ubyte')';
train_label = loadMNISTLabels('train-labels.idx1-ubyte');
test_set = loadMNISTImages('t10k-images.idx3-ubyte')';
test_label = loadMNISTLabels('t10k-labels.idx1-ubyte');
```

```
% training
tic;
model = libsvmtrain(train_label, train_set, '-s 0 -t 0');
% classification
tic;
[predicted_label, accuracy, decision_values]=libsvmpredict(test_label,
test_set, model);
fprintf('Time: %0.2f minutes. \n', toc/60);
toc;
confusionchart(test_label',predicted_label)

% polynomial Kernel
tic;
model = libsvmtrain(train_label, train_set, '-s 0 -t 1 -d 2 -g 1 -r 1');
% classification
[predicted_label, accuracy, decision_values]=libsvmpredict(test_label,
test_set, model);
fprintf('Time: %0.2f minutes. \n', toc/60);
toc;
confusionchart(test_label',predicted_label)

%Gaussian kernel
% training
tic;
model = libsvmtrain(train_label(1:30000,:), train_set(1:30000,:), '-s 0 -t 2');
% classification
[predicted_label, accuracy, decision_values]=libsvmpredict(test_label,
test_set, model);
fprintf('Time: %0.2f minutes. \n', toc/60);
toc;
confusionchart(test_label',predicted_label)

% KNN
train_set = train_set';
test_set = test_set';
tic; % Timing starts
class = {'0','1','2','3','4','5','6','7','8','9'};
% K=1
model = fitcknn(train_set,train_label,'NumNeighbors',1,'ClassNames',class);
% K=3
%model = fitcknn(train_set,train_label,'NumNeighbors',3,'ClassNames',class);
% K=7
%model = fitcknn(train_set,train_label,'NumNeighbors',7,'ClassNames',class);
predict = predict(model,test_set);
```

```
predict = str2num(cell2mat(predict));
fprintf('Time: %0.2f minutes. \n', toc/60); % Shows the time taken for the
training session in minutes
toc % End timing, automatic display of time spent in seconds
temp = predict-test_label;
num = 0;
for i = 1:length(test_label)
if(temp(i)==0)
num=num+1;
end
end
accuracy = num/length(test_label)
confusionchart(test_label',predict)

ThreePerClsShow(x_test, y_test, predict);
% Three images from each category are selected for display in the test set,
% with the labeled values in the lower left corner and the trained predicted
values in the lower right corner
function ThreePerClsShow(x_test, y_test, predict)
for i=0:9
index = find(y_test==i,3); % Find the three instances of category i and return
their location
figure(i+2);
for j=1:3
TImage = full(reshape(x_test(index(j),:),[28,28]));
subplot(1,3,j);
imshow(TImage);
text(2,26,num2str(y_test(index(j))),'Color','w');
text(26,26,num2str(predict(index(j))),'Color','w');
end
sgtitle(['Handwritten numbers',num2str(i),'predictions (data on the bottom
left are labeled values, bottom right are predicted values)']);
set(gcf,'position',[200,300,600,300]);
end
end
```