

Coursework

by Siyu Chen

Submission date: 04-Jan-2022 01:06PM (UTC+0000)

Submission ID: 167517052

File name: 33162_Siyu_Chen_Coursework_393723_1501883934.docx (464.25K)

Word count: 3924

Character count: 22590

1 Introduction

In the 1950s, a group of scientists with dreams formally proposed the concept of Artificial Intelligence (Artificial Intelligence) at Dartmouth College in the United States, ushering in a new era of Artificial Intelligence. Since then, countless scientists have devoted themselves to artificial intelligence. In the decades that followed, artificial intelligence rose and fell, and the technology continued to evolve. As an important part of artificial intelligence, OCR (Optical Character Recognition) technology has also developed vigorously during this period. The concept of optical character recognition (OCR) was first proposed by The German scientist Tausheck[1] in 1929. The early definition of OCR is to recognize characters in print from paper documents. With the passage of time, the connotation of OCR has changed, which is no longer limited to characters in print, but also covers handwritten text recognition and natural scene text recognition.

As a branch of handwritten text recognition, handwritten digit recognition technology is much simpler than handwritten text recognition. Although it is different from printing font recognition, each person's handwriting has their own style, there are differences, but there are fewer categories, there are only ten categories (0~9), so it is also known as the beginning of handwritten text recognition technology.

It can be divided into offline handwritten digit recognition and online handwritten digit recognition. Among them, the online handwritten digit recognition is relatively simple, it use real-time monitoring digital input terminal will be written stroke into one dimensional signal, the tip of the mobile trajectory into coordinate point sequence input computer, therefore on-line handwritten numeral recognition processing is a one-dimensional stroke string of information, the information covers the stroke, stroke order, writing speed and total number of strokes, It has certain continuity and traceability. The offline handwritten digital recognition is different, which is mainly by the terminal input static, non-traceable two-dimensional digital dot matrix image, computer processing is complicated.

Online handwritten digit recognition is simpler, but less applicable than offline recognition. Because handwriting acquisition requires a specific input device, and offline recognition only needs pictures, it is more widely used in practical applications, and online recognition tasks can be converted to offline recognition tasks without time series, so the method adopted in offline recognition tasks is more widely applicable. This paper focuses on offline handwritten number recognition.

2 Related work

Ravi Badu[2] proposed a new approach to off-line handwritten digit recognition based on structural features. He used Euclidean minimum distance criterion to find minimum distances and used k-nearest neighbour(KNN) classifier. As a result, he got 96.94% recognition rate in MNIST. Bernard[3] used random forest(RF) algorithm to recognize the data in MNIST and got different recognition rates with different number of trees and the number of random features pre-selected. Markowska-Kaczmar[4] used support vector machine and got 98.50% recognition rate in MNIST. Besides, he combined KNN and reduced a lot of computing time. Wenfei[5] used convolutional neural network got a better recognition rate compared to the approaches based on KNN, SVM and back propagation(BP).

In this paper, only four basic methods are used to identify MNIST, including three traditional machine learning algorithms and a classical model based on convolutional networks.

3 Methodology

3.1 KNN

K-nearest-Neighbour (KNN)[6][6] algorithm is a common classification and regression algorithm. This section focuses on the k-nearest neighbour algorithm for classification problems. The basic idea of KNN is as follows: given a training set, for a new input sample, find K samples closest to it in the training set, and classify the new input sample into which category most of the K samples belong.

The working principle of KNN :

- (1) Suppose there is a data set with labels , in which each piece of data corresponds to the category to which it belongs.
- (2) Input a new data without labels and compare the new data with the data in the sample data set.
 - a) Calculate the distance between the new data and each item in the sample set
 - b) All the distance values obtained are sorted in ascending order (the smaller the distance is, the more similar the two samples are)
 - c) Select the labels corresponding to the first K samples
- (3) The labels corresponding to the first K samples were counted, and the labels corresponding to more samples were used as labels of new data.

The KNN algorithm diagram is shown in Figure 3.1

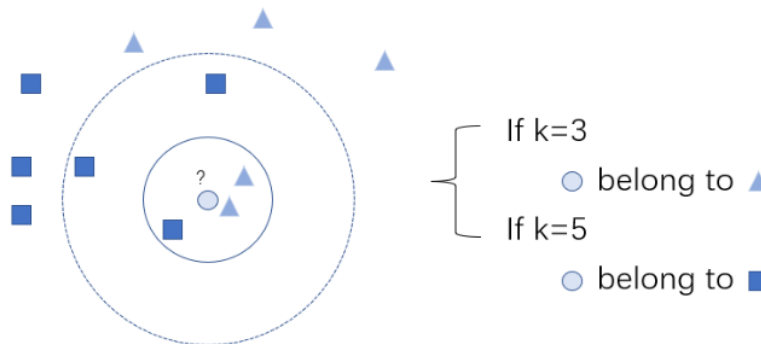


Fig 3.1 KNN sample diagram

As shown in Figure 3.1, rectangle and triangle are two different categories. Given a new sample -- circle, its category needs to be determined. If $k=3$ is taken, two of the three points closest to it belong to triangle category, and it can be inferred that circle belongs to triangle category. If $k=5$, then three of the five points closest to it belong to the rectangle class, so it follows that the circle belongs to the rectangle class.

3.2 SVM

Support Vector Machine (SVM)[7] is a classical binary classification algorithm. The segmentation hyperplane it finds has better robustness, so it is widely used in many tasks and shows strong advantages.

Given a binary classifier data set $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$, $y_n \in \{+1, -1\}$. If two classes of samples are linearly separable, there is a hyperplane

$$\mathbf{w}^T \mathbf{x} + b = 0 \quad (3.1)$$

If you separate the two types of samples, then for every sample there are

$$y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + b) > 0 \quad (3.2)$$

The distance from each sample $x^{(n)}$ in the data set $\mathcal{D}$ to the segmented hyperplane is

$$\gamma^{(n)} = \frac{|\mathbf{w}^T \mathbf{x}^{(n)} + b|}{\|\mathbf{w}\|} = \frac{y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + b)}{\|\mathbf{w}\|} \quad (3.3)$$

We define the interval (Margin) γ as the shortest distance from all samples in the entire data set $\mathcal{D}$ to the partitioned hyperplane:

$$\gamma = \min_n \gamma^{(n)}. \quad (3.4)$$

If the interval γ is larger, the partition of the two data sets by the partition hyperplane is more stable and is not easily affected by noise and other factors. The goal of support vector machines is to find a hyperplane ($\mathbf{w}^*$, b^*) so that γ is maximum,

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \gamma \\ \text{s.t.} \quad & \frac{y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + b)}{\|\mathbf{w}\|} \geq \gamma, \forall n \in \{1, \dots, N\}. \end{aligned} \quad (3.5)$$

Because at the same time zoom $\mathbf{w} \rightarrow k\mathbf{w}$ and $b \rightarrow kb$ will not change the sample $x^{(n)}$ the distance to the hyperplane segmentation, we can limit $\|\mathbf{w}\| \cdot \gamma = 1$ the formula (3.5) is equivalent to

$$\begin{aligned} \max_{\mathbf{w}, b} \quad & \frac{1}{\|\mathbf{w}\|^2} \\ \text{s.t.} \quad & y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + b) \geq 1, \forall n \in \{1, \dots, N\}. \end{aligned} \quad (3.6)$$

Data set all meet $y^{(n)}(\mathbf{w}^T \mathbf{x}^{(n)} + b) = 1$ sample points, are called Support Vector

For a linearly separable data set, there are many partitioned hyperplanes, but the hyperplane with the largest interval is unique. Figure 3.2 gives an example of the maximum interval segmentation hyperplane of a support vector machine, where the sample points with thickened contours are support vectors.

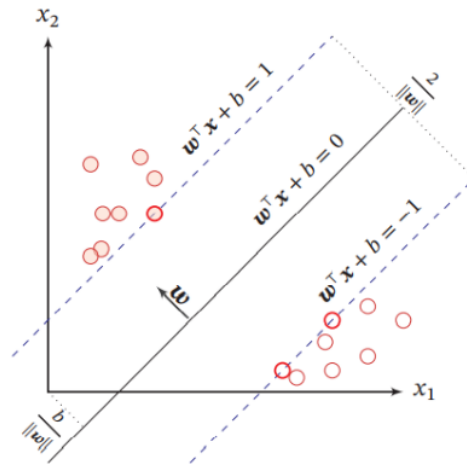


Fig 3.2 an example of Support vector machine

3.3 Bagging and Random Forest

3.3.1 Bagging principle and process

Bagging's weak learners do not have the same connection as Boosting. It features "random sampling". Random sampling (Bootstrap) is to collect a specific number of samples from the training set. The each fetched sample will be put back. In other words, samples previously fetched may continue to be fetched after being put back. For our Bagging algorithm, we generally randomly collect samples with the same number as the training set sample number m . In this way, the number of samples fetched is equal to that of the training set, but the contents of them are different. If we do T times random samples in training sets having m samples, then each sample sets of T will be different because of randomness.

Notice that this is different from the GBDT subsample. The subsampling of GBDT is a no-put sampling, while the subsampling of Bagging is a put back sampling. In random sampling, when fetched a sample from a training set which has m samples, the probability of being fetched each time is $\frac{1}{m}$. The probability of not being fetched is $1 - \frac{1}{m}$. The probability of not being sampled is $\left(1 - \frac{1}{m}\right)^m$. When $m \rightarrow \infty$,

$$\left(1 - \frac{1}{m}\right)^m \rightarrow \frac{1}{e} \approx 0.368 \quad (3.7)$$

It means that about 36.8% of samples in the training set will not be fetched in every round of random sampling of Bagging. We always call these not fetched samples as Out of Bag data(OOB). As a result, these data can be used to test the generalization ability of the model because they don't relate to fitting of the training set model.

As same as Adaboost, Bagging don't limit to weak learners. But the most common ones are decision trees and neural networks. Bagging's strategy of aggregation is simple. For classification problems, simple voting is often used, with the category or one of the categories receiving the most votes being the final output of the model. For regression problems, simple averaging method is usually used. Arithmetic averaging will be used to figure out the regression results obtained by T weak learners for sake of obtaining the final model. Since bagging

algorithm takes samples every time to train the model, it has strong generalization ability and plays a vital role in decreasing the variance of the model. However, the degree of fitting for the training set will be worse to some extent, that is, the model bias will be larger. The bagging algorithm flow is as follows:

Input is sample set $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$, weak learner algorithm, weak classifier iteration times T .

The output is the final strong classifier $f(x)$

(1) For $t = 1, 2, \dots, T$:

(a) Random sampling is performed on the training set for the t -th time, and a total of m samples are collected, fetching a sampling set D_t having m samples

(b) Train the t th weak learner $G_t(x)$ with sampling set D_t

(2) If it is the prediction of classification algorithm, the category or one of the them which get the most votes by the T weak learners is the final one. If it is a regression algorithm, the final output model will be obtained through arithmetic average of the regression results got from T weak learners.

3.3.2 Random Forest

Random forest[8] is a supervised learning algorithm. As you can see from its name, it creates a forest and makes it random in some way. The "forest" built is an integration of decision trees and is mostly trained with a "bagging" approach. Bagging methods, bootstrap Aggregating, involve randomly placing selected training data, building classifiers, and finally combining the learned models to increase overall performance.

A random forest builds multiple decision trees and merges them together for more accurate and stable predictions. They can be used for both classification and regression problems. It is one of the great advantages of random forests. These two types of problems are exactly what most current machine learning systems face. How random forests can be used for classification problems will be focused on. To some extent, classification is sometimes considered a cornerstone of machine learning. A random forest of two trees can be seen on Figure 3.3

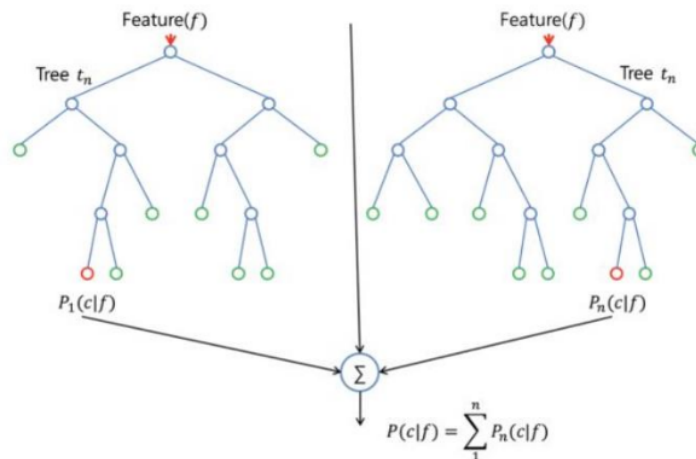


Fig 3.3 Random Forest(two trees)

Except a few exceptions, random forest classifiers use all of the decision tree classifiers as well as the bagging classifier's hyperparameters to control the overall structure. Compared to using the Bagging classifier and passing it to the decision tree classifier, using the random forest classifier class directly is much more convenient and optimized for the decision tree. The regression problem also has a random forest regressor corresponding to it.

The growth of trees in random forest algorithms brings more randomness to the model. Differ from decision trees, each node is segmented into the best feature with minimized error, and the best segmentation is constructed from randomly selected features. Thus, in random forest, only the random subsets used to divide nodes are considered, and it is even possible to make the tree more random by using random thresholds on each feature, rather than searching for the optimal threshold as in a normal decision tree. This process yields a wide variety and often leads to better models.

In general, random forest is simple. And it is easy to implement with low computational cost. It is regarded as "a method representing the counting level of ensemble learning". As you can see, random Forest has made only minor changes to Bagging. The diversity of Bagging is only due to the randomness of samples. The diversity of the base learner of random forest comes not only from this, but also from the randomness of attributes. When the number of learners increases, the random forest always converges to a lower generalization error.

3.4 Neural Network

LeNet-5[9] is a very successful neural network model, although it was proposed earlier. A handwritten digit recognition system based on LeNet-5 was used by many American banks in the 1990s to identify handwritten digits on checks. The network structure of LeNet-5 is shown in Figure 3.4.

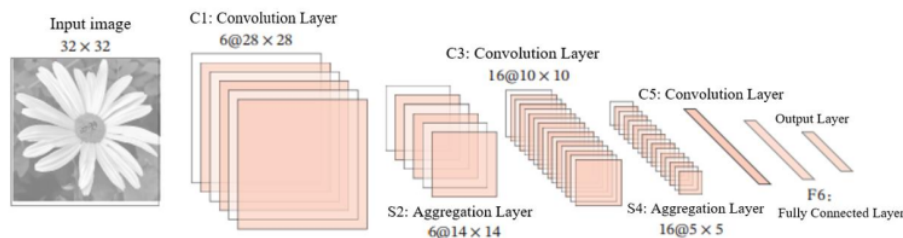


Figure 3.4 LeNet-5 network structure (image based on [LeCun et al., 1998])

LeNet-5 has 7 layers, accepts input image size of $32 \times 32 = 1024$, and outputs scores corresponding to 10 categories. The structure of each layer in LeNet-5 is as follows:

- (1) C1 layer is the convolution layer. Six 5×5 convolution kernels are used to obtain six groups of feature maps with a size of $28 \times 28 = 784$. As a result, there are 4704 (6×784) neurons in C1 layer. There are 156 ($6 \times 25 + 6$) trainable parameters. And the connection number is $156 \times 784 = 122\,304$ (including bias, the same below).
- (2) S2 layer is aggregation layer, sampling window is 2×2 , average aggregation is used, and a nonlinear function. There are 1176 ($6 \times 14 \times 14$) neurons, 12 ($6 \times (1 + 1)$) trainable parameters, and 5880 ($6 \times 196 \times (4 + 1)$) connections.
- (3) C3 layer is the convolution layer. LeNet-5 uses a connection table to define the dependence between input and output feature maps, as shown in Figure 3.4. A total of sixty 5×5 convolutional kernels are used to obtain 16 groups of feature maps with a size of 10×10 . The

ere are 1600 (16×100) neurons , 1156 ($((60 \times 25) + 16)$ trainable parameters, and 151 600 ($100 \times 1 516$) connections.

(4) The S4 layer is an aggregation layer, and the sampling window is 2×2 , and 16 characteristic reflections of 5×5 size are obtained. There are 32 (16×2) trainable parameters and 2000 ($16 \times 25 \times (4 + 1)$) connections.

(5) C5 layer is a convolution layer. 120 groups of 1×1 feature maps are obtained by using $120 \times 16 = 1 920$ 5×5 convolution kernels. There are 120 neurons in C5 layer , 48 120 ($1 920 \times 25 + 120$) trainable parameters and 48 120 ($120 \times (16 \times 25 + 1)$) connections.

(6) F6 layer is a fully connected layer. There are 84 neurons and 10164 ($84 \times (120 + 1)$) trainable parameters. The number of connections is as same as that of trainable parameters, 10164.

(7) Output layer: The output layer is composed of 10 Radial Basis functions (RBF).

Each output feature map of the convolution layer depends on all input feature maps, which is equivalent to a fully connected relationship between the input and output feature maps of the convolution layer. In fact, this fully connected relationship is not necessary. We can make each output feature map dependent on a few input feature maps. Define a Link Table T to describe the connection between input and output feature maps. In LeNet-5, the basic setup of the join table is shown in Figure 3.5. The 0-5 feature mapping of C3 layer depends on every 3 consecutive subsets of the feature mapping group of S2 layer, the 6-11 feature mapping depends on every 4 consecutive subsets of the feature mapping group of S2 layer, and the 12-14 feature mapping depends on every 4 discontinuous subsets of the feature mapping group of S2 layer. The 15th feature map depends on all feature maps of S2 layer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Figure 3.5 Join table of C3 layer in LeNet-5 (Image source: [LeCun et al., 1998])

If the output feature map p th depends on the input feature map d th, $T_{p,d} = 1$, otherwise 0.
 Y^p for

$$Y^p = f \left(\sum_{\substack{d, \\ T_{p,d}=1}} W^{p,d} \otimes X^d + b^p \right), \quad (3.8)$$

T is a connection table of $P \times D$ size. Assume the connection table T nonzero number for K , the magnitude of each convolution kernel is $U \times V$, so need $K \times V \times U + P$ parameters in all.

4 Results and discussion

In this paper, four methods (KNN,SVM,RF and LeNet-5) were used to recognize MNIST. All methods are realized by MATLAB. And the code can be seen in the appendix.

```
>> KNN
Name          Size          Bytes Class  Attributes
images_test   10000x784         62720000 double
images_train  60000x784        376320000 double
labels_test   10000x1           80000 double
labels_train  60000x1          480000 double

Accuracy =

100.0000  98.3607  97.9592  93.6842  96.0784  97.2603  97.9592  97.2222  94.4954  96.1165

Avg_Accuracy =

96.9136
```

Fig 4.1 Recognition results based on KNN

```
Total nSV = 15238
Accuracy = 98.14% (9814/10000) (classification)
```

Fig 4.2 Recognition results based on SVM

```
Accuracy is 97.220000
```

Fig 4.3 Recognition results based on Random Forest

```
Accuracy is 0.990000
```

Fig 4.4 Recognition results based on LeNet-5

36

It can be seen from Figure 4.1-4.4 that the identification accuracy based on Lenet-5 is the highest (99%), followed by the method based on SVM (98.14%), the method based on random forest (97.2%) and the method based on KNN (96.91%). But the LeNet-5-based approach also took the longest time. The above recognition accuracy is not a definite value, which may be affected by many factors, including but not limited to the choice of kernel function, k value, number of trees, etc.

5 Conclusion

In recognition of handwritten numerals task, based on the traditional machine learning method (including KNN and SVM, RF) a small amount of calculation, the calculation time is shorter, but the method based on neural network precision is higher, although it seems only a two percent, however, as the character recognition task difficulty ascending (from optical character recognition, text recognition to the natural scene), The drawbacks of traditional machine learning methods will be exposed. With the improvement of computing power, some of the increased computation is no longer the first factor to be considered in non-real-time tasks, but accuracy is the primary factor. With the emergence of various neural networks, character

recognition has reached an unprecedented height, although accompanied by model degradation, model is too complex and other problems, but also appeared Residual Network[10], model pruning and other solutions. In general, in the field of character recognition, methods based on deep learning have become the mainstream and have been widely used in the society.

6 References

- [1] S. Mori, C. Y. Suen and K. Yamamoto, "Historical review of OCR research and development," in Proceedings of the IEEE, vol. 80, no. 7, pp. 1029-1058, July 1992, doi: 10.1109/5.156468.
- [2] U. R. Babu, Y. Venkateswarlu and A. K. Chintha, "Handwritten Digit Recognition Using K-Nearest Neighbour Classifier," 2014 World Congress on Computing and Communication Technologies, 2014, pp. 60-65, doi: 10.1109/WCCCT.2014.7.
- [3] S. Bernard, S. Adam and L. Heutte, "Using Random Forests for Handwritten Digit Recognition," Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2007, pp. 1043-1047, doi: 10.1109/ICDAR.2007.4377074.
- [4] U. Markowska-Kaczmar and P. Kubacki, "Support vector machines in handwritten digits classification," 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), 2005, pp. 406-411, doi: 10.1109/ISDA.2005.87.
- [5] W. Liu, J. Wei and Q. Meng, "Comparisons on KNN, SVM, BP and the CNN for Handwritten Digit Recognition," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications(AEECA), 2020, pp. 587-590, doi: 10.1109/AEECA49918.2020.9213482.
- [6] Y. Lee, "Handwritten Digit Recognition Using K Nearest-Neighbor, Radial-Basis Function, and Backpropagation Neural Networks," in Neural Computation, vol. 3, no. 3, pp. 440-449, Sept. 1991, doi: 10.1162/neco.1991.3.3.440.
- [7] M. E. Mavroforakis and S. Theodoridis, "Support Vector Machine (SVM) classification through geometry," 2005 13th European Signal Processing Conference, 2005, pp. 1-4.
- [8] Tin Kam Ho, "Random decision forests," Proceedings of 3rd International Conference on Document Analysis and Recognition, 1995, pp. 278-282 vol.1, doi: 10.1109/ICDAR.1995.598994.
- [9] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.
- [10] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

Essential Resources

Appendix

(1)KNN

```
clear
load MNIST_digit_data
whos

rand('seed', 1);
inds = randperm(size(images_train, 1));
images_train = images_train(inds, :);
labels_train = labels_train(inds, :);

%train_size = 1000; %%%%%vary this to change training size
inds = randperm(size(images_test, 1));
images_test = images_test(1:1000, :);
labels_test = labels_test(1:1000, :);
k = 1;
a = size(labels_test);
index = 14;
% A(images_train, labels_train, images_test(index,:), k, 60000); %A
B(images_train, labels_train, images_test, labels_test, k); %B
% Avg Acc = C(images_train, labels_train, images_test, labels_test, k, 1); %c
% D(images_train, labels_train, images_test, labels_test, k) %D
%E(images_train(1:1000, :), labels_train(1:1000, :), images_train(1001:2000, :),
labels_train(1001:2000, :)) %E

function E(images_train, labels_train, images_test, labels_test)

    for k = 1:10
        [acc, avv_Acc(k)] = kNN(images_train, labels_train, images_test, labels_test, k, 1000);
        ki(k) = k;
    end
    display(avv_Acc);
    figure;
    bar(ki, avv_Acc);
    xlabel('values of k')
    ylabel('Average Accuracy')

end

function D(images_train, labels_train, images_test, labels_test, k)
    ki = [1,2,3,5,10];
    for i = 1:5
        acc_i = C(images_train, labels_train, images_test, labels_test, ki(i), 0);
        for j=1:10
            acc(i,j) = acc_i(j);
        end
    end
    display(acc);
    train_size = logspace(log10(30),log10(10000),10);
    figure;
    plot(train_size, acc(1,:), train_size, acc(2,:), train_size, acc(3,:), train_size, acc(4,:), train_size,
acc(5,:));
    legend('k(1)', 'k(2)', 'k(3)', 'k(5)', 'k(10)')
    xlabel('Training Data')
    ylabel('Accuracy')
```

end

```
function avv_Acc = C(images_train, labels_train, images_test, labels_test, k, disp)
    train_size = logspace(log10(30), log10(10000), 10);
    avv_Acc = zeros(size(train_size));
    for i = 1:10
        [acc, avv_Acc(i)] = kNN(images_train, labels_train, images_test, labels_test, k,
int16(train_size(i)));
    end
    %display(avv_Acc);
    if disp == 1
        figure;
        bar(train_size, avv_Acc);
        plot(train_size, avv_Acc);
        xlabel('Number of training data 30 to 10000')
        ylabel('Average Accuracy')
    end
end
```

```
function B(images_train, labels_train, images_test, labels_test, k)
    [Accuracy, Avg_Accuracy] = kNN(images_train, labels_train, images_test, labels_test, k,
60000);
    display(Accuracy);
    display(Avg_Accuracy);
end
```

```
function A(images_train, labels_train, image_test, k, train_size)
    [I, predict_num] = fitkNN(images_train, labels_train, image_test, k, train_size);
    show_img(predict_num, image_test);
end
```

```
function [acc, acc_av] = kNN(images_train, labels_train, images_test, labels_test, k, train_size)
    correct = zeros(1,10);
    total = zeros(1,10);
    acc = zeros(1,10);
    for i = 1:size(labels_test)
        point = images_test(i, :);
        [I, predict_val] = fitkNN(images_train, labels_train, point, k, train_size);
        if predict_val == labels_test(i)
            index = labels_test(i)+1;
            correct(index) = correct(index)+1;
        end
        total(labels_test(i)+1) = total(labels_test(i)+1)+1;
    end
    for i = 1:10
        acc(i) = 100*correct(i)/total(i);
    end
    acc_av = mean(acc);
end
```

```
function [I, predict_val] = fitkNN(images_train, labels_train, image_test, k, train_size)
    dist = pdist2(image_test, images_train(1:train_size,:));
    max_bin = zeros(1,10);
    for i = 1:k
        [M,I] = min(dist);
        [a, b] = max(dist);
    end
```

```

        dist(I) = a;
        max_bin(labels_train(I)+1) = max_bin(labels_train(I)+1)+1;
    end
    [a, predict_val] = max(max_bin);
    predict_val = predict_val-1;
end

```

```

function show_img(predict_num, image)
    close all
    im = reshape(image(:), [28 28]);
    imshow(im)
    title(num2str(predict_num));
end

```

end

(2)SVM

% install libsvm at first

clear;

clc;

for i=1:60000

str = strcat('./train_data/',num2str(i));

name = strcat(str,'.bmp');

name = char(name);

current_img = imread(name);

current_img = im2bw(current_img,5/255);

current_img = reshape(current_img,1,[]);

current_img = double(current_img);

train_image(i,:) = current_img;

end

load train-label.txt

model = libsvmtrain(train_label, train_image, '-s 0 -c 4.0 -t 2 -g 0.015625 -r 91.1 -d 3');

save('Model','model');

for i=1:10000

str = strcat('./test_data/',num2str(i));

name = strcat(str,'.bmp');

name = char(name);

current_img = imread(name);

current_img = im2bw(current_img,5/255);

current_img = reshape(current_img,1,[]);

current_img = double(current_img);

test_image(i,:) = current_img;

end

load test-label.txt

[predict_label1, accuracy, dec_values] = libsvmpredict(test_label, test_image, model);

(3)Random Forest

clear all;

clc;

for i=1:60000

str = strcat('./train_data/',num2str(i));

name = strcat(str,'.bmp');

name = char(name);

current_img = imread(name);

```

        current_img = reshape(current_img,1,[]);
        train_image(i,:) = current_img;
    end
    train_image=double(train_image);

    for i=1:10000
        str = strcat('./test_data/',num2str(i));
        name = strcat(str,'.bmp');
        name = char(name);
        current_img = imread(name);
        current_img = reshape(current_img,1,[]);
        test_image(i,:) = current_img;
    end
    test_image=double(test_image);

    lable_test = textread('test-lable.txt');
    lable_train = textread('train-lable.txt');
    % lable_train = lable_train(1:100);
    % lable_test = lable_test(1:100);

    model = TreeBagger(500,train_image,lable_train); % use train set to get a model

    result = predict(model,test_image); % predict based on the model

    result = cell2mat(result);

    sc=double(result) - lable_test;
    accuracy=sum(sc(:)==48)/100.0; % calculate the accuracy
    fprintf('Accuracy is %f\n',accuracy);

```

(4)LeNet-5

```

%load data
imageDim = 28;
numclasses = 10;
images = loadMNISTImages('train-images.idx3-ubyte');
images = reshape(images,imageDim,imageDim,[]);
labels = loadMNISTLabels('train-labels.idx1-ubyte');
labels(labels == 0) = 10;

%% initial parameters
lenet.layers = {
    struct('type','i')
    struct('type','C1','outputmaps',6,'kernelsize',5)
    struct('type','S2','scale',2)
    struct('type','C3','outputmaps',16,'kernelsize',5)
    struct('type','S4','scale',2)
    struct('type','C5','outputmaps',120)
    struct('type','F6','outputmaps',84)
    struct('type','Soft','num_classes',10)
};

lenet = initialParameters(lenet,images);

%% train the cnn
opts.alpha = 1;

opts.batchsize = 40;

```

```

opts.numepochs = 20;

lenet = cnnTrain(lenet,images,labels,opts);

%% test the cnn
testImages = loadMNISTImages('t10k-images.idx3-ubyte');
testImages = reshape(testImages,imageDim,imageDim,[]);
testLabels = loadMNISTLabels('t10k-labels.idx1-ubyte');
testLabels(testLabels == 0) = 10;

lenet = cnnff(lenet,testImages);

a = lenet.layers{8}.a;
[~,preds] = max(lenet.layers{8}.a,[],1);
preds = preds';
33
acc = sum(preds == testLabels) / length(preds) * 100%;
fprintf('Accuracy is %f\n',acc);

```

Coursework

ORIGINALITY REPORT

27%
SIMILARITY INDEX

11%
INTERNET SOURCES

15%
PUBLICATIONS

21%
STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Sydney Student Paper	3%
2	Submitted to University of Nottingham Student Paper	2%
3	Submitted to University of Hong Kong Student Paper	2%
4	Submitted to University of Florida Student Paper	2%
5	Submitted to Vrije Universiteit Brussel Student Paper	1%
6	www.coursehero.com Internet Source	1%
7	Submitted to Liverpool John Moores University Student Paper	1%
8	Submitted to University College London Student Paper	1%
9	Submitted to University of Wales, Lampeter Student Paper	1%

10

www.ijariit.com

Internet Source

1 %

11

Hasan Asil, Jamshid Bagherzadeh. "A new approach to image classification based on a deep multiclass AdaBoosting ensemble", International Journal of Electrical and Computer Engineering (IJECE), 2020

Publication

1 %

12

Submitted to Colorado Technical University Online

Student Paper

1 %

13

Submitted to Manchester Metropolitan University

Student Paper

1 %

14

Submitted to The Hong Kong Polytechnic University

Student Paper

1 %

15

Submitted to University of Sheffield

Student Paper

1 %

16

Xu Xin, Fu Jun, Wang Li, Gong Dongmei, Wang Yujun, Tang Peilin. "Intelligent Research on Customer Complaints Based on 95598 Work Orders Track", Journal of Physics: Conference Series, 2019

Publication

1 %

17

dokumen.pub

Internet Source

1 %

18

Yuehui Han, Yusheng Hao, Weilan Wang.
"Research on the character recognition of
Tibetan ancient document based on CNN",
Journal of Physics: Conference Series, 2019

Publication

<1 %

19

conservancy.umn.edu

Internet Source

<1 %

20

dspace.bracu.ac.bd:8080

Internet Source

<1 %

21

Submitted to Washington University in St.
Louis

Student Paper

<1 %

22

Xin Hu, Hua Ouyang, Yang Yin. "Image
Recognition based on Convolution Neural
Network", 2020 IEEE 9th Joint International
Information Technology and Artificial
Intelligence Conference (ITAIC), 2020

Publication

<1 %

23

Congyi Lei, Jianzhong Mao, Xiaomin Zhang,
Lian Wang, Ding Chen. "Crack prediction in
sheet forming of zirconium alloys used in
nuclear fuel assembly by support vector
machine method", Energy Reports, 2021

Publication

<1 %

24

Han He, Sicheng Li, Lin Hu, Nelson Duarte,
Otilia Manta, Xiao-Guang Yue. "Risk Factor
Identification of Sustainable Guarantee

<1 %

Network Based on Logistic Regression Algorithm", Sustainability, 2019

Publication

25

vision.stanford.edu

Internet Source

<1 %

26

Lei Chen. "Deep Learning and Practice with MindSpore", Springer Science and Business Media LLC, 2021

Publication

<1 %

27

Yiwen Lin. "Research on Business Development Strategy of Application Software Store", 2020 2nd International Conference on Applied Machine Learning (ICAML), 2020

Publication

<1 %

28

"Proceedings of the International Field Exploration and Development Conference 2017", Springer Science and Business Media LLC, 2019

Publication

<1 %

29

Hao Liu, Yun Liang Chen, Ningning Cui, Da Zhao Xu, Jianxin Li. "An Effective Data Fusion Model for Detecting the Risk of Transmission Line in Smart Grid", IEEE Internet of Things Journal, 2021

Publication

<1 %

30

Uma Narayanan, Athira Unnikrishnan, Varghese Paul, Shelbi Joseph. "A survey on

<1 %

various supervised classification algorithms",
2017 International Conference on Energy,
Communication, Data Analytics and Soft
Computing (ICECDS), 2017

Publication

31	Submitted to NCC Education Student Paper	<1 %
32	Submitted to University of Glasgow Student Paper	<1 %
33	Submitted to Bilkent University Student Paper	<1 %
34	Submitted to Hong Kong University of Science and Technology Student Paper	<1 %
35	repository.its.ac.id Internet Source	<1 %
36	www.hindawi.com Internet Source	<1 %
37	Submitted to Shri Mata Vaishno Devi University(SMVDU), Katra Student Paper	<1 %
38	zenodo.org Internet Source	<1 %
39	"Innovations in Smart Cities Applications Edition 3", Springer Science and Business Media LLC, 2020	<1 %

40

Kayode S. Adewole, Abimbola G. Akintola, Rasheed Gbenga Jimoh, Modinat A. Mabayoje et al. "Cloud-based IoMT framework for cardiovascular disease prediction and diagnosis in personalized E-health care", Elsevier BV, 2021

Publication

<1 %

41

Lecture Notes in Computer Science, 2013.

Publication

<1 %

42

Shigeo Abe. "Support Vector Machines for Pattern Classification", Springer Science and Business Media LLC, 2010

Publication

<1 %

43

Submitted to University of Leeds

Student Paper

<1 %

44

doi.org

Internet Source

<1 %

45

Graupe, Daniel. "Deep Learning Convolutional Neural Network", Deep Learning Neural Networks, 2016.

Publication

<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off