Handwriting Digit Recognition Based on Support Vector Machine and Neural Network

Zepei Zhao (5635405542) Mingjun Liu (1321657359)
Minghong Xu (3989201972) Zhixin Xie (7529805472)

Abstract: The main purpose of this study is to compare the performance of two algorithms, support vector machines(SVM) and neural networks (NN), for handwritten number recognition. In this study, SVM and ANN were used as the comparison items, and the data were preprocessed based on PCA and binary scaling. This experiment constructed a hand-written Digits Recognition Model based on two algorithms to measure the actual handwritten digital images. The experimental results showed that the accuracy of NN is better than that of SVM, and the time of computation was greatly reduced when the data set was dimensionality reduced by PCA.

1. Introduction and Related Work

A. Introduction

Handwritten digit recognition is a branch of optical character recognition(OCR). It mainly studies the use of computers to automatically recognize the 10 Arabic numerals (0-9) written on paper or electronic devices by humans. Handwritten digit recognition technology is very comprehensive, involving image processing, pattern recognition, artificial intelligence, statistical decision theory, fuzzy mathematics, comprehensive mathematics, information theory, computer and other disciplines, and in data statistics, postal systems, literature Retrieval, bill processing, office automation, etc. have very broad application prospects.[1]

In the computer vision field, object recognition is popular for solving problems related to digital image or digital video. Handwritten digit recognition is an essential but hard practical classification problem. Generally, digit recognition can be divided into two parts, print digit recognition and handwritten digit recognition.[2] Compared to print digit, handwritten digit is more difficult to recognize, due to there are numerous handwritten styles and the relationship of digits written is subtle. Therefore, handwritten digit recognition requires more effort to classify different digits.

In this paper, gray scale image and binary image are applied to pre-processing, and they are compared in different algorithms. In order to optimize model efficiency, dimensional reduction method is also introduced in our model. Due to easier to accomplish and simple computation, principal components analysis is used in our algorithms. To make a high performance model and also aim to combine the material learned in machine learning in data science courses, support vector machine and neural network are utilized in this paper. Through data pre-processing and parameter selection, training our models and making optimization.

B. Related Work

Digit recognition systems are generally divided into modules such as image preprocessing, feature extraction, and character recognition.[3] In image preprocessing, for color images, each pixel is usually represented by three components: red (R), green (G), and blue (B), with components between (0, 255). However, color images need large space to store and high computational performance. Hence, gray scale images are widely used in image processing. Gray scale image only has one sampled color per pixel, which is usually displayed as a grayscale from the darkest black to the brightest white, although in theory this sample can be any color of different shades, even can be different colors at different brightness. Binary Image, according to the name to understand there are only two values, 0 and 1, 0 represents black, 1 represents white, or 0 represents the background, and 1 represents the foreground. Its storage is also relatively simple, each pixel only needs 1Bit to completely store information. If each pixel is regarded as a random variable, there are a total of N pixels, then the binary image has a N-th power variation of 2, and the 8-bit grayscale image has a N-th power variation of 255. There are 255 * 255 * 255 Nth power variations. That is to say, for images of the same size, the binary map saves less information.[4]

To improve the compute efficiency, PCA is an important method combined with a genetic algorithm for feature selection, solving the time-consuming and low-efficiency problem caused by high dimensions. Besides, Linear Discriminant Analysis (LDA) and Isomap are also popular dimensional reduction methods.[5]

When lower dimension vectors obtained, machine learning algorithms are required to classify digit characters. There are abundant of research comparing performance of different classifier, such as support vector machine, neural network, K nearest neighbors, decision tree, convolutional neural network and etc,.[6] Anuj and Aashi make a comparison on accuracy and time between machine learning algorithms(RFC,KNN,SVM) and deep learning (multilayer CNN) using MNIST dataset.[7] Priya and their team focus on optimize the performance of SVM. They use proximal support vector machine over standard SVM classifier, with 98.65% accuracy, and it shows better performance than artificial neural network.[8] While, regarding artificial neural network, Saeed and others train 5000 samples from MNIST data set with gradient descent back propagation and test data with feed-forward algorithms. This multilayer perceptron neural network has 35 neutrons and 250 iterations, so that their accuracy is up to 99.32%.[9] Besides, different kernels have strongly effect on the performance of model, hence Fabien and others make an accuracy comparison between NN and SVM classifiers with various kernel.[10] Typically, pre-processing methods are researched on different algorithms. LeCun compare KNN, SVM, CNN and NN with various pre-processing methods, such as deskewing and making subsampling to certain size pixels.[11]

2. Overview of Data

A. Data Sample

The dataset is provided by the MNIST ("Modified National Institute of Standards and Technology") database, which is a classic dataset of handwritten images served as the basis for benchmarking classification for machine learning. Among the dataset, there are 42,000 training samples and 28000 testing samples, which are all size-normalized and centered in a fixed-size image. Therefore, we don't need to preprocess for data cleaning and formatting. The images were adapted from the black and white images from NIST with size normalized to fit in a 20*20 pixel box while preserving their aspect ratio by normalization algorithm. Then they were centered in a 28*28 image by computing the center of mass of the pixels, and positioning the point at the center of the 28*28 field.

The dataset has two files train.csv and test.csv, which contains gray-scale images of hand-written digits from zero through nine. Each image is of size 28*28 pixels, in total 784 pixels, and each pixel has a single pixel-value indicating the lightness or darkness between 0 and 255, while 0 means the whitest and 255 means the darkest. The training dataset has 42,000 rows and 785 columns, where the first column is the label of each image, each row corresponds to an image and each column corresponds to a pixel of the image. Assume the pixel is at the i^{th} row and j^{th} column of the image, then the pixel is at the $(28*i+j)^{th}$ column of the csv file. The test.cvs works the same as the

training file, but it does not contain the label column. Therefore, for benefiting the testing step, we divided the train.csv into two parts, 80% of it is for training the models, and 20% is for testing the models. Since the high dimension of the training data, we introduce two methods for dimension reduction to fasten the training process.

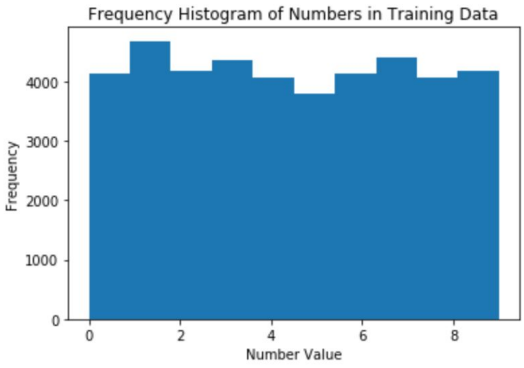

Fig 1 Number Value Frequency

B. Gray Image and Binary Image

The first method is changing the gray scale images to binary images, whose pixel values are 0 or 1. It converts the gray scale images to black and white only by changing the pixel values that are larger than 0 to 1. It contains less information in one single image but costs less memory and processes faster. For example, a binary image of size n pixels has kinds of different images, and a gray scale image of size n pixels has kinds of different images.

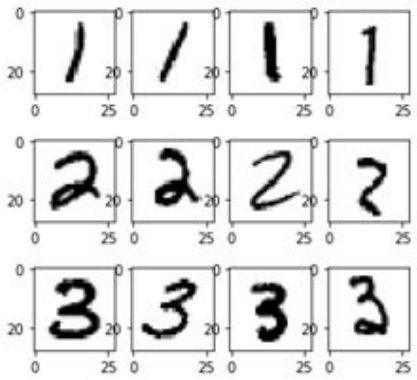

Fig 2 Original Images

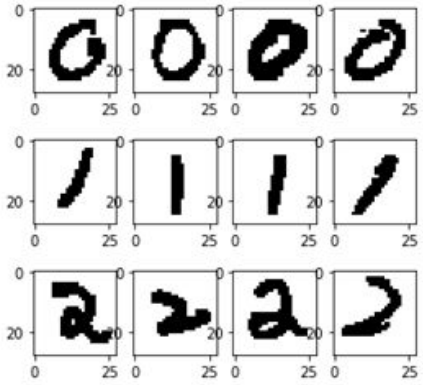

Fig 3 Binary Images

C. Dimensional Reduction: PCA

The second method is by principal components analysis, which is an unsupervised method to apply dimensionality reduction. PCA will first calculate the mean vector of the data on each dimension, then calculate the covariance matrix of the difference between each data to the mean vector, which is a n by n matrix. Next, find the eigenvalues of the covariance matrix, choose the eigenvectors that correspond to the first k largest eigenvalues, which means has the most influence on the data. Last, the first k principal components of each original data will be the product of the

matrix consisting of the chosen eigenvectors (k by n matrix) and the original data (n by 1 vector). Then the dimension of the original dataset is reduced from n to k.

Below is an example about PCA of a multivariate Gaussian distribution centered at (1,3) with a standard deviation of 3 in roughly the (0.866, 0.5) direction and of 1 in the orthogonal direction. The vectors shown are the eigenvectors of the covariance matrix scaled by the square root of the corresponding eigenvalue, and shifted so their tails are at the mean. [12]

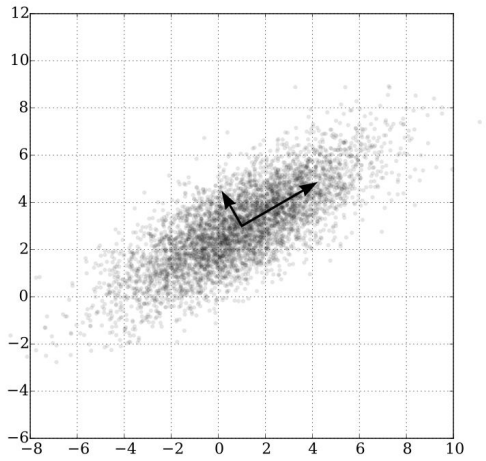

Fig 4 Eigenvectors in PCA

When applying PCA to images, we will calculate the mean vector of the training data on each corresponding pixel (dimension), then calculate the covariance matrix of the difference between each image pixel to the mean vector. Then follow the steps above to find the values for the first k pixels (dimension). In this project we employed the module sklearn.decomposition.PCA to perform dimension reduction. By using this module, we can simply define n_components as the number of components to keep, and the function fit_transform(sample) to fit the model with the sample and apply dimensionality reduction on it. We keep 90% information of the dimensions (n_components) in the images.

3. Digits Recognition with SVM

A. Introduction to SVM

The first algorithm we used in this project is the support-vector machines (SVMs), which are supervised learning models that perform linear or nonlinear data classification and regression. An SVM training algorithm builds a model that fits the given sample into a model with the given label and returns the parameters for the determination process. In other words, in a high-dimensional data space, a support vector classifier is to find the hyperplane that has the maximum margin that distinctly classifies the data points into the desired groups. For example, a two-dimensional data space with two classes requires an SVC to find a straight line that maximizes the margin between the two classes.

In linear SVC, the classes are linearly separable by a hyperplane, but it often happens that the sets are not linearly separable in that space, therefore, we introduce a kernel function to map the original finite-dimensional space into a higher-dimension space to fit the classes linearly. The kernel function can be linear, polynomial, RBF (Gaussian radial basis function), sigmoid or customized. The most common kernel function is RBF, because it has localized and finite response along the entire x-axis. In this project, the kernel function we used is RBF also.

To construct a linear SVM, we need to determine the support vectors first, which are the closest points to the hyperplane. [13] Given training vectors $x_i, i = 1, ..., n,$ in two classes, and a label $y \in \{1, -1\}$, SVC solves the primal problem:

$$\begin{aligned} \min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \\ \zeta_i \geq 0, i = 1, \ldots, n \end{aligned}$$

Its dual is

$$\begin{aligned} \min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \\ \text{subject to } y^T \alpha = 0 \\ 0 \leq \alpha_i \leq C, i = 1, \ldots, n \end{aligned}$$

where $e^T = 1^T$, C is the upper bound, Q is an semidefinite matrix with $Q_{ij} = y_i y_j K(x_i, x_j)$ where $K(x_i, x_j)$ is the kernel. This results the decision function :

$$\text{sgn}(\sum_{i=1}^{n} y_i \alpha_i K(x_i, x) + \rho)$$

For example, we define the margin is defined as as $|w * x + b| = 1$, where w is the weight vector, and b is the bias (intercept). Then we introduce a parameter alpha to determine the support vectors. If alpha is larger than 0, then the corresponding data point is considered as support vectors. The alpha is solved by quadratic programming with a dual problem. Next we need to determine the weight vector and intercept for the hyperplane. We determine the weight vector as $w = \sum_{n=1}^{N} \alpha_n y_n x_n$, where N is the number of support vectors, x_n is the corresponding data point, and y_n is the corresponding flag to the data point. The intercept is $b = y_m - w^T x_m$. Then the hyperplane is found, where the test data points are separated above and below.

In the module sklearn.svm.SVC, we can modify the parameters kernel as the kernel function, gamma as the kernel coefficient to be scale or auto, C as the regularization parameter, and tol as the tolerance, etc. After fitting with the training data, the attribute support_ returns the indices of support vectors, support_vectors_ returns the support vectors, n_support_ returns the number of support vectors for each class, dual_coef_ returns the coefficients (weights) of the support vectors in the decision function, intercept_ returns the constants in decision function, classes_ returns the classes labels, and fit-status_ returns 0 if correctly fitted. The predict(sample) function gives the prediction to each test data and score(sample, label) function gives the mean accuracy on the given test data and labels.

B. Experimental Process

In this section, we use four different combinations of handwriting digits recognition methods. The model, fitting time, test score and test time are shown in each part, which helps us evaluate the performance of each model and make an optimization.

a. Keep all pixel values

All pixel information is kept in this part, so that we focus on the difference of gray scale image and binary image. And also a basic implementation of support vector machine algorithm will be shown in this part.

a1. Gray Scale Image and SVM

Gray scale image is the most common representation for image in computer science, which projects the light and color to 256 different degrees, so the range of each pixel is between 0 and 255.

Model:

```
clf = svm.SVC(random_state = 42)

clf.fit(train,train_labels.values.ravel())

score = clf.score(test,test_labels)
```

a2. Binary Image and SVM

To optimize data storage, binary image is well wide used in image recognition, which converts each pixel on the image to two possible values or gray scale states. Therefore, values who are not equal to 0 are transformed to 1. Though each pixel only needs 1 bit to store information completely, binary image saves less information than color image and gray scale image.

Model:

```
data[data > 0] = 1

clf = svm.SVC(random_state = 42)

clf.fit(train,train_labels.values.ravel())

score = clf.score(test,test_labels)
```

b. Dimensional Reduction with PCA

The image in the digits recognition data set is divided into 784 pixels stored in a matrix, while most components are useless for recognition. Therefore, dimensional reduction method is introduced to extract features that contain more vital information. Principal components analysis uses the idea of dimensional reduction to transform multiple indicators into a few comprehensive indicators.

b1. Gray image, PCA and SVM

First, train and test data are transformed to standard format, and then plot all components to decide which components are meaningful.

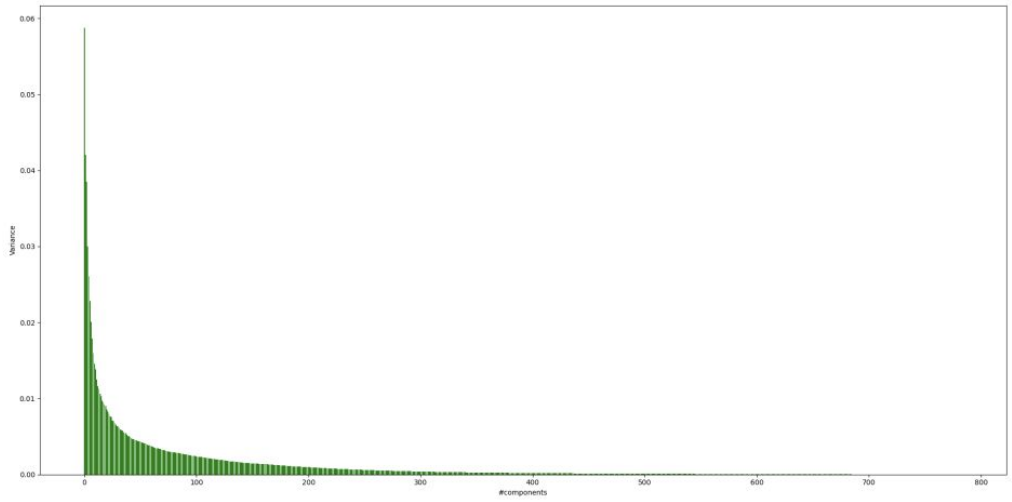

Fig 5 Percentage of Variance Explained by Components-Gray Scale Image

The figure shows that around 90% cumulative selected components can represent at least 90% information of the image. So in this case, 90%, that is 217 major components are selected for further model train and fit.

b2. Binary image, PCA and SVM

First, process data greater than 0 to 1. Then standardize training data and transform train data and test data. Hence, sklearn PCA can be used to fit train data and get the components information. Through visualization, important components are recognized for further computation.

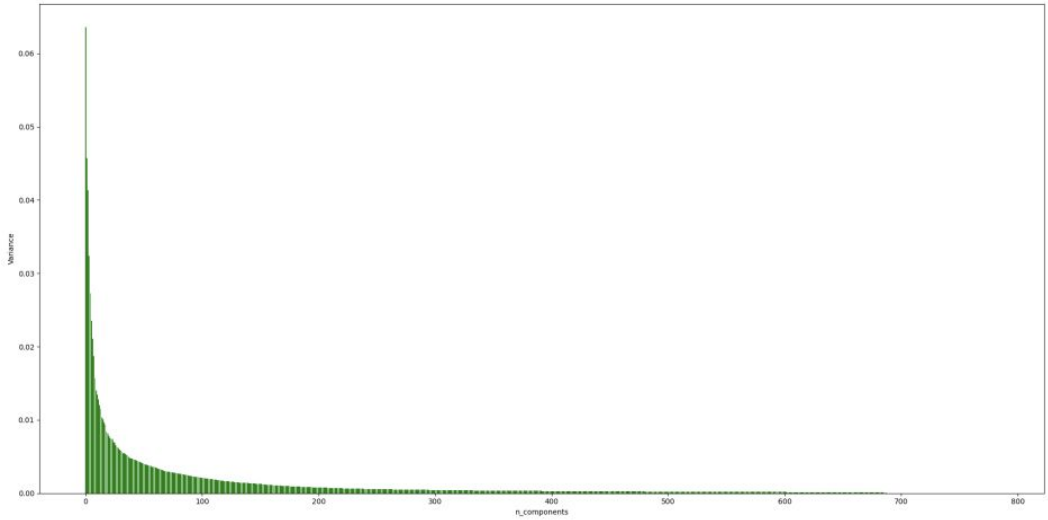

Fig 6 Percentage of Variance Explained by Components-Binary Image

In this case, the figure is slightly different from the previous one. To save more than 90% information of image, 281 major components are advised to select for further model training. However, in order to control variables, we still selected 217 components in this case.

C. Experimental Analysis

a. Model selection

Model performance is evaluated by time for fitting train data, time for score test data and accuracy of test data. Due to less storage occupied and less execution time cost, binary image is a better choice for digit recognition, though it may miss some information. Besides, dimensional reduction is an essential process for pre-process data, which sharply reduces the execution time. Hence, principal component analysis will be utilized in our model. Through the analysis of indexes mentioned before, accuracy has slightly changed. As a result, the model generated is a combination of binary image, principal component analysis and support vector machine.

b. Parameter Selection

To find the optimized parameters for the model, grid search cross validation is applied.In the default SVM model, the parameters are {'C': 1.0, 'break_ties': False, 'cache_size': 200, 'class_weight': None, 'coef0': 0.0, 'decision_function_shape': 'ovr', 'degree': 3, 'gamma': 'scale', 'kernel': 'rbf', 'max_iter': -1, 'probability': False, 'random_state': 42, 'shrinking': True, 'tol': 0.001, 'verbose': False}. Basically, there are some important parameters that largely determine performance of the model. One is kernel function, which makes the data in the feature space separable by mapping the linearly inseparable data in the input space to a high latitude feature space as the goal. The Gaussian function (RBF) is a strongly localized kernel function, which can map a sample to a higher dimensional space. This kernel function is the most widely used, whether it is a large sample or a small sample, and it has fewer parameters than polynomial kernel function. Therefore, RBF is selected as the kernel function in our model.

There are two other vital parameters, ‘C’ is penalty coefficient, shows the tolerance for errors. The higher the C, the more intolerable the error is, and it is easy to overfit. The smaller C, the easier to underfit. C is too large or too small, the generalization ability becomes poor. Generally, C has a reference list for users to select, that is "C":[0.1, 1, 10,100].

Besides, due to RBF being selected, gamma is also a very essential parameter to be the coefficient of RBF. It implicitly determines the distribution of the data after mapping into a new feature space. The larger the gamma, the fewer the support vectors, and the smaller the gamma value, the more the support vectors. The number of support vectors affects the speed of training and prediction. ‘gamma’ is generally selected from [1, 0.1, 0.01,0.001].

To figure out what are the best parameters for our model, Grid search is applied in this part. Choose the best hyper parameters among the 9 cases formed by C = (0.1,1,10,100) and gamma = (1, 0.1, 0.01,0.001), and use 5-fold cross-validation.

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(SVC(), param_grid = {"C":[0.1, 1, 10,100], "gamma":[1, 0.1, 0.01,0.001]}, cv =5)
grid.fit(tr_pca2,train_labels.values.ravel())
print("The best parameters are %s with a score of %0.2f" %(grid.best_params_, grid.best_score_))
```

Parameter selection: kernel = ‘rbf’, C = 10, gamma = 0.001

c. Result

Based on the analysis, our model is clf = svm.SVC(random_state = 42, kernel = ‘rbf’, C = 10, gamma = 0.001).

Table 1 Result of SVM

Time for fitting model	Time for score model	Accuracy
54.55s	18.50s	0.9693

4. Neural Network

A. Introduction of NN

Artificial Neural Network (ANN), referred to as Neural Network (NN), is a mathematical model based on the basic principle of Neural Network in biology. After years of development, ANN is characterized by parallel distributed processing ability, high fault tolerance, intelligence and self-learning ability, combines the processing and storage of information, and attracts the attention of all disciplines with its unique knowledge representation and intelligent adaptive learning ability.

B. Algorithm

Neural Networks are composed of multi-layer perceptrons. There are one input layer, one or more hidden layers, and one output layer. And each layer has numerous perceptrons. When we design a neural network, the number of perceptrons in the input layer and output layer is fixed, which is equal to the dimension of input data point and number of results. Suppose our data point has N-dimensions.

$$x = [x_1, x_2, \ldots, x_N]$$

Each data point corresponds to one label. Then we have N perceptrons at input layer and one perceptron at output layer.

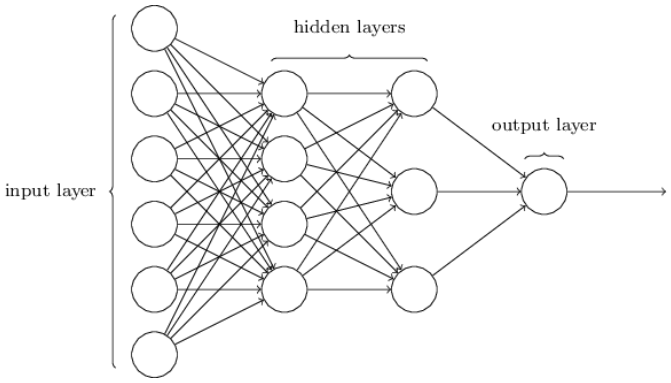

Fig 7 Structure of Neural Network

When looking closer for one perceptron in the hidden layer (one circle in the image at hidden layer part), it actually consists of a summation function and an activation function. Each dimension of the data point corresponds to one weight.

$$w = [w_1, w_2, \ldots, w_N]$$

The goal of Neural Network is finding the best weights that can predict the label of one data point. E.g. $w^T x = y$

In this specific perceptron, $y = f(\sum_{i=1}^{N} x_1 * w_1)$.

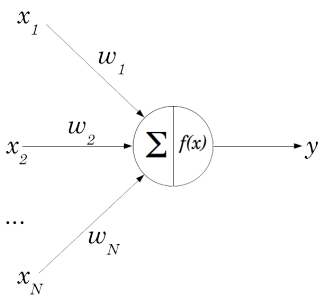

Fig 8 Structure of Neutron

f(x) is the activation function. Everytime we input new data, we need to adjust the weights. To avoid the influences of outliers, use the activation function to eliminate outliers. Common activation functions are:

-- tanh: get data range [-1, 1]
-- sigmoid: get data range [0, 1]
-- ReLu: keep value larger than 0, else become the value to 0

After going through the activation function, the perceptron would get a result y. We can output the y or if we want to have more layers, we can pass the y to the next layer. To have more results, just need to add more perceptrons using the same x as the input but use different w. To differentiate w, use 2-d subscripts to represent w. And to differentiate different layers, use superscripts l to represent layer number. So the input layer would be the first layer: $x = [x_1^{(0)}, x_2^{(0)}, \ldots, x_N^{(0)}]$.

And first layer weights would be: $w = [w_{1,1}^{(0)}, w_{2,1}^{(0)}, \ldots, w_{N,1}^{(0)}, w_{1,2}^{(0)}, \ldots, w_{1,j}^{(0)}, \ldots, w_{N,j}^{(0)}]$

j is the number of perceptrons in the second layer. So the output of first layer would be:
$[x_1^{(1)}, x_2^{(1)}, \ldots, x_j^{(1)}]$

For convenience, set $S_j^{(l)} = \sum_{i=0}^{d^{(l-1)}} x_i^{(l-1)} * w_{i,j}^{(l)}$.
Because the output is a function of $x_i^{(L)}$ and y, suppose it is $e(x_i^{(L)}, y)$.
To optimize the value of e, we need to take a derivative with respect to $w_{i,j}^{(l)}$ for all i, j, l.

$$\text{So } \frac{\partial e}{\partial w_{i,j}^{(l)}} = \frac{\partial e}{\partial S_j^{(l)}} * \frac{\partial S_j^{(l)}}{\partial w_{i,j}^{(l)}} .$$

$$\text{Set } \delta_j^{(l)} = \frac{\partial e}{\partial S_j^{(l)}} .$$

$$\text{So } \frac{\partial e}{\partial w_{i,j}^{(l)}} = \delta_j^{(l)} * x_i^{(l-1)}$$

To illustrate how to update weights. We use the most popular algorithm called back propagation algorithm.
(1) Compute all $x_j^{(l)}$ in forward direction including the final output $x_1^{(L)}$.

$$x_j^{(l)} = f(\sum_{i=0}^{d^{(l-1)}} x_i^{(l-1)} * w_{i,j}^{(l)})$$

Base case: $x_i^{(0)}$ = input data

(2) Compute all $\delta_i^{(l-1)}$ in backward descent.

$$\delta_i^{(l-1)} = f'(S_j^{(l-1)}) * (\sum_{j=1}^{d^{(l)}} \delta_j^{(l)} * w_{i,j}^{(l)})$$

Base case: $\delta_1^{(L)} = 2(x_1^{(L)} - y) * f'(S_j^{(L)})$

(3) Update w:

$$w_{i,j}^{(l)} = w_{i,j}^{(l)} - \eta x_i^{(l-1)} * \delta_j^{(l)}$$

η : Learning rate

The function we called for our neural networks model was 'MLPClassifier' from the 'scikit-learn' package. MLP training was carried out on two arrays: one is the array X (n_samples, n_features), holding the training samples represented by floating point feature vectors; the other one is the array y (n_samples,), holding the target value of the training sample (class label). The function implemented a multilayer perceptron (MLP) algorithm and applied the form of gradient descent for training, and the gradient was calculated using a back-propagation technique. For classification, it minimized the cross entropy loss function and gave each sample an estimate of the probability P(Y|X) that the sample belongs to each category.

C. Experiment Process

a. Data Preparation

Except for the PCA and binary data process, we scaled each pixel value to [0,1] before we splitted data to train and test dataset since MLP is sensitive to feature scaling.

b. Parameter Selection

As the algorithm introduced above, we needed to decide our input layer. Because each line in the data represented one digit, used each line as one input data. The data had dimensions of 784, so the input size would be 784. And each line represented one digit, the output size would be one.

We chose the simple version of Neural Network, which has only one hidden layer. And one hidden layer was also enough for this problem. For tuning the parameters, we applied 'GridSearchCV' function which searched the hyper-parameter space for the best cross validation score. Based on the highest train score, we set our hidden layer size as 500, the learning rate as 0.1, and the L2 penalty as 0.1.

Use 80% training data as input. For each line in the training data, pass it through our neural network model and update the weights. After 1000 iterations of training, we obtain our model.

5. Comparison

A. Comparison of Experimental Results

Besides the 80% data used for training the model, use the remaining 20% data as testing data. And use testing data as input data to our fitted model, the digits can be predicted. Compare the predicting results with the true label. We can get the accuracy rate of each algorithm.

Table 2 Performance of SVM Model

Model	Time for fitting model	Time for score model	Accuracy
Gray scale+SVM	191.74s	72.21s	0.9735

Binary image+SVM	179.74s	78.02s	0.9694
Gray+PCA+SVM	82.81s	23.82s	0.9615
Binary+PCA+SVM	76.25s	21.31s	0.9602

Table 3 Performance of NN Model

Model	Time for fitting model	Time for score model	Accuracy
Gray scale+NN	100.33s	0.10s	0.9770
Binary image+NN	124.41s	0.17s	0.9726
Gray+PCA+NN	24.43s	0.04s	0.9730
Binary+PCA+NN	17.37s	0.04s	0.9680

From the tables shown above, there is some information that can be observed. For the model with PCA(principal components analysis), time for fitting model and score model is significantly reduced, because it decreases the dimension of data. However, it also decreases a little bit accuracy of the prediction result. This means PCA is useful for picking up major components, and meantime, saves the most valuable information.

Furthermore, the models that are based on gray scale and without PCA have the highest accuracy for both SVM model and NN model, because they contain the most complete information, while they cost more time to fit and score the model.

And binary images take less time than gray scale images, and there is little difference in accuracy between them. So, we choose a binary image as the representation of the digits for the SVM, combined with PCA. However, for NN, because NN transfer data to scope [0,1], the binary images do not have any promotions towards the fitting time nor accuracy. Binary images even increase the running time for both fitting and scoring time. The effect of binary images is decreased after applying PCA, because PCA diminished the dimensions. So we choose a gray image as representation of the digits for NN, combined with PCA.

B. Comparison of SVM and NN

The comparison between SVM and NN has been raised numerous times over the history. When the SVM was optimized by Vapnik and others in the 1990s, it showed advantages at multiple parts over NN like high efficiency and global optimization. And SVM replaced the NN to become the most popular algorithm since that time. NN entered into the ‘Frozen period’ and did not revive until recent years. NN came back as a modified version and a new name as ‘deep learning’.

Because we have not learned much about deep learning, the version of NN we used is still the old version from the 20th century. So by intuition, the performance of NN should be worse than SVM. However, our results show that NN shows advantage both on time and accuracy. In our experiment, the best model we chose for NN takes 24.43s to fit and reaches accuracy of 0.9730. The best model we chose for SVM takes 76.25s and reaches accuracy of 0.9612. For accuracy, the NN model is 1.1% higher than the SVM model. However, NN takes much less time to get these results, which is only 32% of SVM. The difference would be enlarged if using a larger database. PCA is very helpful

for both algorithms. The PCA saves around 58% time for SVM. And it even saves almost 76% time for the NN model.

It is common to wonder why SVM would take more time but not perform so well over accuracy. After our research, the reason should be the principle of SVM. The classical SVM is actually a dichotomous algorithm. However, in our problem, we have 10 digits. In order to cope with multiple classes problems like the digits recognition, we need a combination of several dichotomous classification SVMs. Therefore, the ultimate fitting time becomes more than NN.

6. Conclusion

Text recognition has been developed for a long time. This study compared two common machine learning algorithms, named support-vector machines(SVM) and Neural Networks(NN). In addition, principal component analysis (PCA) and binary scaling were also adopted in this study to perform a pre-processing of data sets in the hope of obtaining more analysis. In this study, two machine learning algorithms combined with the data preprocessing actions of PCA and binary scaling, a total of eight different combinations of experimental results were finally sorted out. It is obvious from the results that if some preprocessing actions are made to the data set, the models can significantly reduce the computing time while maintaining the same accuracy level. In this study, only comparison and actual measurement of hand-written numbers have been made. In the future, it is expected that further recognition of Chinese characters can be implemented. Also, we optimized our estimator by cross-validated grid-search over a parameter grid which highly improved the efficiency of tuning parameters. If we explore more possibilities of tuning parameters, we believe the training structure will also have better performance.

7. Contribution

Zepei and Mingjun worked for data processing and implementation of the support vector machine algorithm. Zhixin and Minghong worked for data processing and implementation of the neural network algorithm. We are responsible for the optimization of our models and code.

8. Reference

[1] E. Tuba, M. Tuba and D. Simian, "Handwritten digit recognition by support vector machine optimized by bat algorithm", 24th International Conference in Central Europe on Computer Graphics Visualization and Computer Vision (WSCG 2016), pp. 369-376, 2016.

[2] Kaensar C. (2013) A Comparative Study on Handwriting Digit Recognition Classifier Using Neural Network, Support Vector Machine and K-Nearest Neighbor. In: Meesad P., Unger H., Boonkrong S. (eds) The 9th International Conference on Computing and InformationTechnology (IC2IT2013). Advances in Intelligent Systems and Computing, vol 209. Springer, Berlin, Heidelberg

[3] X. Yan, L. Wen, and L. Gao, "A fast and effective image preprocessing method for hot round steel surface," Mathematical Problems in Engineering, vol. 2019, Article ID 9457826, pp. 1–14, 2019.

[4] Mohan V.M., Kanaka Durga R., Devathi S., Srujan Raju K. (2016) Image Processing Representation Using Binary Image; Grayscale, Color Image, and Histogram. In: Satapathy S., Raju K., Mandal J., Bhateja V. (eds) Proceedings of the Second International Conference on Computer and Communication Technologies. Advances in Intelligent Systems and Computing, vol 381. Springer, New Delhi

[5] A. Das, T. Kundu and C. Saravanan, "Dimensionality Reduction for Handwritten Digit Recognition," EAI Endorsed Transactions on Cloud Systems, vol. 4, no. 13, 2018.

[6] Kamavisdar P, Saluja S, Agrawal S (2013) A survey on image classification approaches and techniques. Int J Advanc Res Comput Commun Eng 2(1):1005–1009

[7] Anuj Dutt, Aashi Dutt, 'Handwritten Digit Recognition Using Deep Learning', (IJARCET) Volume 6, Issue 7, July 2017, ISSN: 2278 – 1323, Page number – 990-997

[8] Priya, Rajendra Singh, Dr. Soni Changlani, 'Handwritten Digit Recognition using Proximal Support Vector Machine' , Journal of Emerging Technologies and Innovative Research, Volume 4, Issue 04,April 2017, ISSN Number: 2349-5162, Page No: 251-254

[9] Saeed AL-Mansoori, 'Intelligent Handwritten Digit Recognition using Artificial Neural Network', Int. Journal of Engineering Research and Applications, ISSN: 2248-9622, Vol. 5, Issue 5, (Part -3) May 2015, pp.46-51

[10] Fabien Lauera,,ChingY. Suenb, Gérard Blocha, 'A trainable feature extractor for handwritten digit recognition', Pattern Recognition Society. Published by Elsevier Ltd., Volume 40, Issue 6, June 2007, Pages 1816-1824, October 2006.

[11] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 'Gradient-Based Learning Applied to Document Recognition', Proceedings of the IEEE, v. 86, pp. 2278-2324, 1998.

[12] By Nicoguaro - Own work, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=46871195

[13] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Images Sources:
https://dzone.com/articles/deep-learning-via-multilayer-perceptron-classifier
https://lucidar.me/en/neural-networks/learning-rule-demonstration/