Machine Learning Final Reports

Researches on MNIST with Various Learning Techniques

Liu Jiannan and Zhao Yao

[1] *Shanghai Jiaotong University CS420*
[2] *https://github.com/LiuChiennan/MlFinal*

June 14, 2018

In this article, we carry out the whole process of the research on MNIST database, including preprocessing, formatting, training, testing and evaluation of each model. Firstly, we exert de-noise methods to avoid extra disturbance. Then, we center digits by the center of mass method. Next, PCA is used for dimension reduction. These are preprocessing and formatting steps. In the next part, traditional learners, namely SVM and knn, and modern methods, convolutional neural network (a.k.a "CNN") are used on extracted training data, so that we finally build three classification models. The final part conveys some of comparisons of the three learners above.

1 Introduction

As a part of *pattern recognization*, image recognization play an important role in artificial intelligence. To recognize a figure, we need lots of samples to learn the targets' features, and according to what the model has learned to classify a figure and determine what it is. In this report, we build some classifiers to classify the MNIST dataset, and explore how these classifiers work and their disadvantages and disadvantages.

2 Preprocessing

The MNIST database is a dataset of handwritten digits, which has a training set of 60,000 examples and a test set of 10,000 examples. It is a subset of a larger set available from NIST. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

2.1 Fromatting

The dataset provided contains only raw figure pictures, which means some of pictures are polluted by noise and some/s center is deviated from their mass center. So it is considerably necessary to adjust these raw figures.

There are a number of previous researches on how to do de-noise on pictures, while the noise in the raw dataset is somehow more like spots (as shown in the figure 1) but not general random noise. That makes it hard to find papers on such a particular problem. In researching of this problem, we resorted to "Attention Mechanism" while found it needs GPGPUs or servers to accelerate. So eventually, we design a specific algorithm which we employ image segmentation algorithm in it. As the name goes, image segmentation algorithms can split a picture into several parts base on correlation in a certain area. Because grey-level figures our dataset contains, the very first method we try is Otsu Method (1979) (ref1), which is a threshold-based segmentation algorithm. The algorithm can be briefly summarized as below:

After Otsu Method, we can get several foreground color areas and background color areas (in general situations there is only one background area) . Then we need to distinguish which foreground area is the valid number area but not a spot. At the very beginning, we try to calculate the area of each part. But it turns out to be a failure because in some pictures the valid area is even smaller than some spots. Then we try to calculate the sum of grey-level in each area, while it fails also because of a similar reason. We find the fact that most spots are much similar to a ball and much smaller than the valid area, which means we can calculate the variance of each foreground area and find one has the biggest variance. That is just the valid area.

Here we gain the main area contains the body of

Figure 1: one hand writing figure

Algorithm 1 otsu method

Require: a grey-level figure which must have an obvious background color
Ensure: threshold: $\boldsymbol{t}$

1: make a gray-level histogram, whose x-axis stands for each level of grey and y-axis stands for frequencies for each level
2: Define $\boldsymbol{M}$ as the average level of the histogram. The part less than $\boldsymbol{t}$ is named $\boldsymbol{A}$ and the other part is named $\boldsymbol{B}$. Then $\boldsymbol{PA}$ is $\boldsymbol{A}$'s ratio of total and $\boldsymbol{PB}$ is $\boldsymbol{B}$'s ratio of total, and $\boldsymbol{MA}$ is the average of $\boldsymbol{A}$ and $\boldsymbol{MB}$ is the average of $\boldsymbol{B}$.
3: traverse $\boldsymbol{t}$ from 0 to 255. For each $\boldsymbol{t}$, calculate

$$\boldsymbol{ICV = PA \times (MA - M)^2 + PB \times (MB - M)^2}$$

Find a $\boldsymbol{t}$ maximize $\boldsymbol{ICV}$. Namely,

$$\boldsymbol{argmax_t PA \times (MA - M)^2 + PB \times (MB - M)^2}$$

Algorithm 2 locate valid area

Require: split threshold given by otsu method, the grey-level figure
Ensure: a unique valid area

1: find all foreground areas with threshold $\boldsymbol{t}$
2: for each area, calculate the variance of both axes and then add up them, the valid area possesses the biggest variance sum

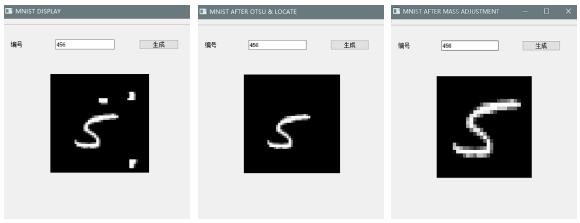

Figure 2: modifications after processes above. from left to right, the raw figure, figure after de-noising and locating, figure after extracting.

sample number, yet it still has a deflected center in the picture. Two traditional ways to correct it are bounding-box method and mass center method. In order to have a statistically better accuracy, we adopt the second one. This paper([2]) provides a way to accelerate the calculation.

Algorithm 3 mass center method

Require: $\mathbf{45 \times 45}$ grey-level figure which was output from algorithm 2
Ensure: $\mathbf{28 \times 28}$ standard MNIST classification size

1: calculate the $\mathbf{45 \times 45}$ figures mass center, make G equals the center
2: box an area around $\boldsymbol{G}$ with size $\mathbf{28 \times 28}$, then calculate the mass center of the sub-area, then update the G
3: if $\boldsymbol{G}$ changes then
4: go to 2
5: else
6: return $\boldsymbol{G}$
7: end if

In experiment, we find that some numbers‘ figures always have a deflected mass center itself for each, so the number seems to have a little different sizes of its four margins. For example, the number “6“ owns mass mostly in its bottom half part while the number “7“ and “9“ have mass mostly in their top half part. However, we finally ignore the fact because number “6“ all have a deflected center, which will not influence the distribution of “6“, so do “7“s and “9“s.

All codes above are implemented in C++ and are saved in the tool(git) folder.

2.2 decomposition: PCA

Principal component analysis (a.k.a PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. If we take the first several principal components out sort by variance, we can achieve dimension reduction but lose as less information as possible.

Extracted figures after centering are 28×28 scale, namely 784 dimensions. We note that many pixels

of figures are always 0, especially the ones near the borders, which contributes to the necessity to execute PCA on it to reduce low variance dimensions. We assume PCA remains 95% information and it turns out that the setting of 240 reduced dimensions is appropriate. Traditional ways to implement PCA are eigenvector method and SVD method and they are mathematically equivalent. We implement both, each's details are as teacher's lectures' description.

PCA's codes are implemented in C++ and are saved in the pre_treatment(git) folder.

3 Methods

3.1 Support Vector Machine

In machine learning, support vector machine(a.k.a SVM) is supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. In addition to performing linear classification,SVM can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.
The derivation of SVM is shown in lectures so we omit it here. The final form of SVM's dual problem is as shown below:

$$\min_{\alpha} \frac{1}{2}\vec{\alpha}^T \cdot QMatrix \cdot \vec{\alpha} - \vec{1}^T\vec{\alpha} \tag{1}$$

$$s.t. \quad 0 \leq \alpha_t \leq C, i = 1, 2, \cdot, n \tag{2}$$

$$s.t. \quad \vec{y}^T\vec{\alpha} = 0 \tag{3}$$

where QMatrix is a semi-positive definite matrix and acts as the quadratic programming's central matrix which each element at (i, j) is

$$q_{ij} = y_i y_j K(\vec{x}_i, \vec{x}_j),$$

where $K(\vec{x}_i, \vec{x}_j)$ is a typical kernel function. C describes the soft margin's tolerant distance and $\vec{\alpha}$ is the parameter vector of dual problem.
A modern way to reduce the problem is discovered by Rong-En Fan s (2005)(ref3). Here we use its algorithm as algorithm 4:

SMO method can update two parameter in each iteration and select a working set which can mostly enhance the gradient decent. Then, the MNIST dataset is a multi-classification problem. To use SVM on it, we build a one-vs-one C-SVC. For each two class, make a pair to buld a SVM classification model. Then for every input examples, work on all models and make a vote for the concrete output class.
SVM's codes are implemented in C++ and saved in the SVM(git) folder. Also, we build a simple test demo (git) for SVM, and post the screenshot as figure 3.

Algorithm 4 SMO for kernel SVM

Require: $\vec{y}$, C, QMatrix
Ensure: optimized $\vec{\alpha}$

1: set $\alpha = \alpha_1$ as the initialization, namely set the k in α_k is 1
2: if α_k is a stationary point of the problem then
3: stop
4: else
5: find a working set $B = \{i, j, \}$ by the following conditions:

$$i \in \arg\max_t \{-y_t \nabla f(\alpha^k)_t | t \in I_{up}(\alpha^k)\} \quad (4)$$

$$j \in \arg\min_t \{-\frac{b_{it}^2}{\vec{\alpha}_{it}} | t \in I_{low}(\alpha^k), \\ -y_t \nabla f(\alpha^k)_t < -y_i \nabla f(\alpha^k)_i\} \quad (5)$$

6: where

$$a_{ij} = q_{ii} + q_{jj} - 2 \times q_{ij}$$

is as shown above,and

$$b_{ij} \equiv -y_i \nabla f(\alpha^k)_i + y_j \nabla f(\alpha^k)_j > 0. \quad (6)$$

$$I_{up}(\alpha) \equiv \{t | \alpha_t < C, y_t = 1 || \alpha_t > 0, y_t = -1\}. \quad (7)$$

$$I_{low}(\alpha) \equiv \{t | \alpha_t < C, y_t = -1 || \alpha_t > 0, y_t = 1\}. \quad (8)$$

7: end if
8: let a_{ij} be defined as above, if $a_{ij} > 0$, solve the sub-problem below

$$\begin{aligned} &\min_{\alpha_\beta} \quad \frac{1}{2} \begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix} \begin{bmatrix} \alpha_\beta \\ \alpha_N^k \end{bmatrix} - \begin{bmatrix} e_B^T e_N^T \end{bmatrix} \begin{bmatrix} \alpha_\beta \\ \alpha_N^k \end{bmatrix} \\ &= \frac{1}{2} \alpha_B^T Q_{BB} \alpha_B + (-e_B + Q_{BN} \alpha_N^k)^T + constant \\ &= \frac{1}{2} \begin{bmatrix} \alpha_i \alpha_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ji} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + (-e_B + Q_{BN} \alpha_N^k)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \end{aligned} \quad (9)$$

$$\begin{aligned} subjectto \quad & 0 \le \alpha_i, \alpha_j \le C, \\ & y_i \alpha_i + y_j \alpha_j = -y_N^T \alpha_N^k. \end{aligned} \quad (10)$$

where $\begin{bmatrix} Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN} \end{bmatrix}$ is a permutation of the matrix Q.otherwise, solve this below:

$$\begin{aligned} &\min_{\alpha_i, \alpha_j} \frac{1}{2} \begin{bmatrix} \alpha_i \alpha_j \end{bmatrix} \begin{bmatrix} Q_{ii} & Q_{ij} \\ Q_{ij} & Q_{jj} \end{bmatrix} \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} + (-e_B + Q_{BN} \alpha_N^k)^T \begin{bmatrix} \alpha_i \\ \alpha_j \end{bmatrix} \\ &+ \frac{\tau - \alpha_{ij}}{4} ((\alpha_i - \alpha_i^k)^2 + (\alpha_j - \alpha_j^k)^2) \end{aligned} \quad (11)$$

$$\begin{aligned} subjectto \quad & 0 \le \alpha_i, \alpha_j \le C, \\ & y_i \alpha_i + y_j \alpha_j = -y_N^T \alpha_N^k, \end{aligned} \quad (12)$$

9: update α as the optimization in 8

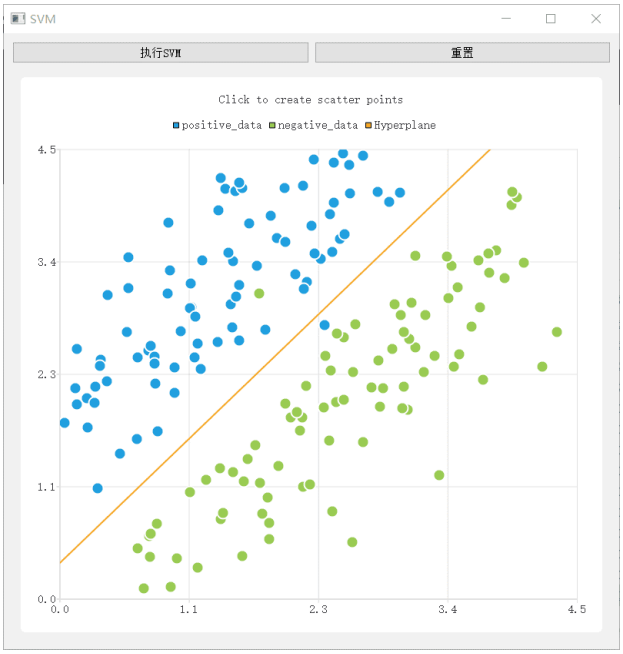

Figure 3: a 2D SVM demo

3.2 K-Nearest Neighbor(k-nn)

k-nn algorithm is a concept of Pattern Recognition,it's a non-parametric method used for classification and regression. In the feature space, the input consists of k closet training samples. When k-nn is used for classification, the output is a class membership, an object is classified by a majority vote of its neighbors, with the object being assigned to class most common among its k nearest neighbors,and k is a positive integer,typically small. When k is 1, the object is simply assigned to the class of which single nearest neighbor.
In k-nn, when we calculate the distance between the training sample and the input sample, we several ways, e.g, Euclidean Distance, Manhattan Distance ,etc. Here I list the formula of the 2 kinds of distance metric.

$$Eudiean(X, Y) = ||X - Y|| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (13)$$

$$Manhattan(X, Y) = |X - Y| = \sum_{i=1}^{n} |x_i - y_i| \quad (14)$$

where Manhattan Distance is also called L_1 norm, Euclidean Distance is called L_2 norm. In this work, we decided to use both the two kinds of distance metrics. Here I list the process of k-nn on MNIST dataset as algorithm 5. Obviously, different distance metrics and different k will have a great influence on the classification results.

Algorithm 5 K-Nearest Neighbors

Require: D, the set of the training samples,and z,the test sample, a vector of attribute values, L, the set of classes used to label the samples.
Ensure: $C_z \in L$, the class of z.
1: for each sample $(x^t, y^t) \in D$ do
2: Compute $d(z, x^t)$, the distance is determined by what metric is used.
3: end for
4: Sort the training samples by the distances increase.
5: Select the first k sorted samples
6: Calculate the frequency of the k samples‘ label
7: return the label which has the highest frequency.

3.3 Convolutional Neural Network(CNN)

In machine learning areas, a convolutional neural network is a deep ,feed-forward artificial neural networks, most commonly applied to visual imagery.
As a very famous and classical deep learning method with high efficence, CNNs classification on high dimensional samples works well compared with the traditional supervised learning algorithms. We use the extracted dataset to train our CNN model, which extracted the noise pixel and others that has no influence on the classification but will cost time.
Manage Inputtype CNN consists of 3 kinds of layers, the input layer, the hidden layer,the output layer, and the hidden layer typically contains convolutional layers, pooling layers, fully connected layers and normalization layers. Different layers play different roles in the CNN model, their mixture let the classifier has higher accuracy.The dataset we use contains 50000 training samples, each sample is a $28 * 28$ figure, begin training, we flatten each sample into a $28 * 28$ array, and meanwhile, norm each pixel to $[0, 1]$ to avoid the Curse of Dimensionality due to each pixel varies form 0 to 255, which may affect the classification result. Otherwise, we encode each label with $one - hot$ encoding for 10 classes, for example, label 1 is assigned to $[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$.
Model Architecture In our group‘s CNN model, we use a model which is similar with the $LeNet5$ model. Our model consists of three convolutional layer, two pooling layer, one fully connected layer and a ouput layer. In CNN model, there are many parameters we can set, for example, the loss function, the optimizer,etc. In our work, we apply different optimizer and loss function to compare their accuracy and get the optimize parameter. Besides, we use Batch Normalization(BN) technician to improve the accuracy and avoid the overfitting. And we add the dropout steps to avoid the overfitting too.
Optimizer In our model, we apply three different optimizer, that is Adagrad,Adadelta ,adam. Here I simply introduce three kinds of optimizer:

Adagrad :

$$n_t = n_{t-1} + g_t^2 \tag{15}$$

$$\triangle\theta_t = -\frac{\eta}{\sqrt{n_t + \epsilon}} * g_t \tag{16}$$

here g_t create a regulizer from 1 to t, and $\Delta\theta_t = -\frac{1}{\sum_{\tau=1}^{t}(g_t)^2+\epsilon}$, and ϵ is used to keep the denominator be 0. It suit to manage the sparse gradient.
Adadelta : Adadelat is the extension of ***Adagrad***.

adadelta can speed the training in the begining time.

$$n_t = v * n_t - 1 + (1 - v) * g_t^2 \tag{17}$$

$$\Delta\theta_t = -\frac{\eta}{\sqrt{n_t + \epsilon}} * g_t \tag{18}$$

$$\tag{19}$$

with ***newton iteration***

$$E|g^2|_{t-1} + (1 - \rho) * g_t^2 \tag{20}$$

$$\Delta x_t = \frac{\sqrt{\sum_{r=1}^{t-1} \Delta x_r}}{\sqrt{E|g^2|_t + \epsilon}} \tag{21}$$

Adam

$$m_t = \mu_{t-1} + (1 - \mu) * g_t \tag{22}$$

$$n_t = v * n_{t-1} + (1 - v) * g_t^2 \tag{23}$$

$$\hat{m_t} = \frac{m_t}{1 - \mu^t} \tag{24}$$

$$\hat{n_t} = -\frac{n_t}{1 - v_t} \tag{25}$$

$$\Delta\theta_t = -\frac{\hat{m_t}}{\sqrt{\hat{n_t}} + \epsilon} * \eta \tag{26}$$

where $\hat{m_t}, \hat{n_t}$ is the estimation of the gradient, and

$$-\frac{\hat{m_t}}{\sqrt{\hat{n_t}}}$$

is a dynamic constrain of the learning rate.
Loss Function Our goal is to classify multi-class targets, so we adopt the ***cross entropy*** as the loss function for our CNN model after unnormalized log probabilities for each class.

4 Results

In this section, we mainly state the experiment results and analysis the results.Consider the extracted dataset‘s size is $28 * 28$, we need integrate the PCA with the KNN and SVM, so I organise this section with the order KNN, SVM, and the CNN.

4.1 K-Nearest Neighbors

In the experiment of ***knn***, we firstly use PCA to reduce the dimensions, and considering if K is too big, the calculation may cost too much time, so we set the K be 10 in the beginning. The execution result is as below(with Euclidean Distance) , To view more

Table 1: KNN Classification Results 1

(K=10,test samples=10000)

N	Accuracy	ErrorRate	Time
5	69%	31%	01:31
10	88.32%	11.68%	02:00
20	94.39%	5.61%	03:11
30	95.31%	4.69%	04:12
40	95.42%	4.58%	05:39
50	95.01%	4.99%	06:45
60	94.72%	5.28%	07:13
70	93.91%	6.09%	08:44

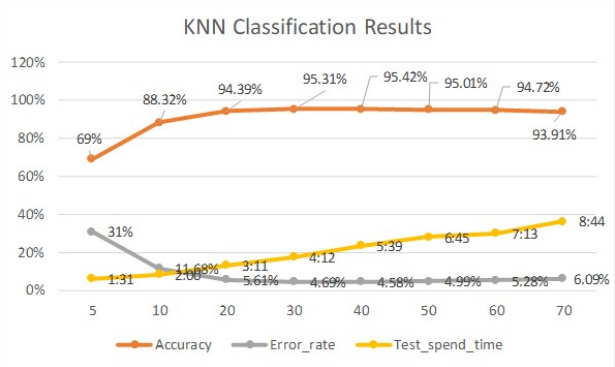

Figure 4: KNN Classification Results 1
(K=10,test samples=10000)

intuitional, we draw a picture. From the table and the figure, we know if the n_components parameter ranges in $[30, 50]$, the accuracy is more than 95%, to find the best n_components, we execute with different n_components with step 2 in range $[30, 50]$, the result is as listed below, in the same, the figure is , Obviously,

Table 2: KNN Classification Results 2

(K=10,test samples=10000)

N	Accuracy	ErrorRate	Time
30	94.38%	5.62%	04:19
32	95.38%	4.62%	04:32
34	95.51%	4.59%	04:38
36	95.36%	4.64%	04:52
38	95.35%	4.65%	05:00
40	95.38%	4.62%	05:15
42	95.31%	4.69%	05:30
44	95.25%	4.75%	05:40
46	95.29%	4.71%	06:37
48	95.08%	4.92%	07:20

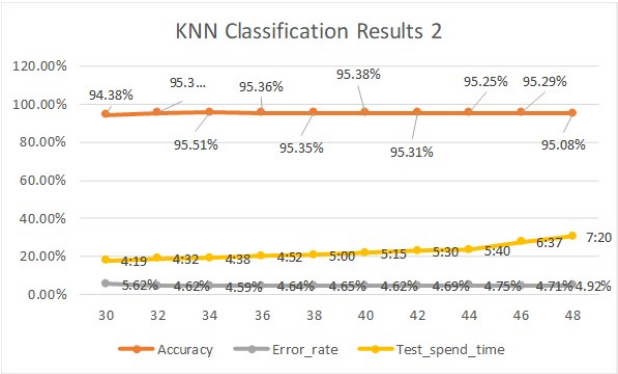

Figure 5: KNN Classification Results 2
(K=10,test samples=10000)

when we reduce the image vector to 34 dimensions, we get the best accuracy, so now we adopt 34 as the best reduced dimensions parameters.
Actually, the k and the distance metrics affect the results too, now in the condition of $n\_components = 34$, we change k and distance metrics, the results is shown as table 3.

Table 3: KNN Classification Results 3

(N=34,test samples=10000)

K	Accuracy(Euclidean)	Accuracy(Manhattan)
1	96.74%	96.5%
2	95.81%	95.35%
3	95.65%	96.29%
4	96.39%	96.21%
5	96.44%	96.06%
7	96%	95.7%
9	95.87%	95.76%
11	95.68%	95.45%
13	95.35%	95.23%
15	95.32%	95.02%
17	95.16%	94.88%
19	94.97%	94.67%

From the results we know that when $k = 1, N = 34$ with *Euclidean Distance* metric, we get a better accuracy on MNIST dataset. Maybe $k = 1$ is strange in KNN, I think it is because that we use PCA in the beginning, and extracted the principal components, then the most similar sample has the sample label compared with the input sample in high probability.

4.2 Multi-class Support Vector Machine

When implementing the SVM algorithm, we use the *Eigen* library in $c++$, and use rbf kernel as the kernel function, use grid search method to find the optimal parameter C and γ, In the depth graph, we find that the best accuracy is 99.21%, a good accuracy but training SVM costs too much time due to the high dimensions.

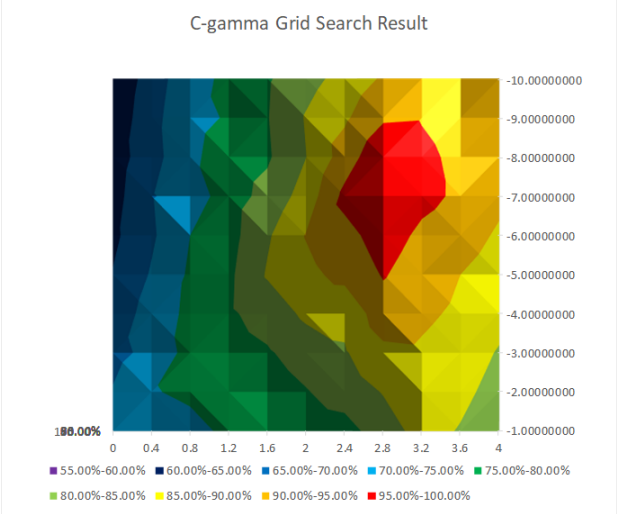

Figure 6: SVM Grid Search Results

4.3 Convolutional Neural Network

Firstly, we use two convolutional layers, one pooling layer without any batch normalization to build the model,and we set the the optimizer adam, adadelta, and adagrad, their accuracy are 98.89%,98.67%,98.48% respectively. The three kinds of optimizer's classification accuracy vary not too much, and adam is better. Although the accuracy is near 99%, we still think it's not good. Therefore we add external one convolutional layer and one pooling layer, and add batch normalization to avoid the over-fitting, and after each Conv layer, we drop 0.25% data to reduce the dimesnions. Here is the results, After several adjustment on our model,

Table 4: CNN Classification Results

Hidden Layers	optimizer	accuracy
3	adam	98.89%
3	Adadelat	98.67%
3	adagrad	98.5%
5	adam	99.29%
5	Adadelat	99.36%
5	adagrad	99.21%

we find that when use adam optimizer as well as 3 convolutional layers and two pooling layers followed by *BatchNormalization*, we get the best accuracy 99.36%.

4.4 Comparison

In our work, we applied three kinds of algorithms to classify the MNIST dataset, each has its own characteristics. KNN is fast than SVM, but its accuracy is not. SVM is actually a deep fully connected neural network in some aspects. CNN contains the correlation of the nearest pixels in a figure, that is why CNN performances well in image recognization. When we want adopt one model to classify a figure in real life, I think the SVM or CNN is better after training due to their high efficiency.

5 Conclusion

Machine Learning is a very attractive subject, sometimes it is quite interesting and sometimes boring. In the process of classifying the MNIST dataset, we implement the algorithms by our own. When we began this report, we have not so much knowledge of machine learning, we study those method by ourselves and benefit a lot in the process. By doing these experiments, we learn to adjust the model to let it performance better, we are more familiar with those algorithms.

6 reference

[1] IEEE Transactions on Systems, Man, and Cybernetics (Volume: 9, Issue: 1, Jan. 1979)
[2] 《计算机辅助设计与图形学学报》2004年第10期|王冰职秦川张仲选耿国华周明全西北大学计算机科学系西安710069
[3] The Journal of Machine Learning Research; Volume 6, 12/1/2005; Pages 1889-1918