Hand Written Digit Recognition Based on Support Vector Matrix (Chen-Yin Yu)

Preliminary Work

1) Store and Visualize the Dataset

In order to read and visualize the Mnist data, I chose to read it as numpy and save it as csv format in the entire project.

The load-mnist function returns two arrays. The first one is a n x m-dimensional NumPy array(images), where n is the number of samples (number of rows) and m is the number of features (number of columns). The training data set contains 60,000 samples and the test data set contains 10,000 samples. Each picture in the MNIST data set consists of 28 x 28 pixels, each pixel being represented by a gray value. Here, we expand the 28 x 28 pixels into a one-dimensional row vector, which is the row in the image array (784 values per line, or each row represents a single image). The second array (tag) returned by the load-mnist function contains the corresponding target variable, which is the class label of the handwritten number (integer 0-9).[1]

Once the dataset is saved as a CSV file, we can reload them into the program using NumPy's 'genfromtxt' function.

2) Read the Dataset

Use the " model-selection -train-test-split" statement when randomly dividing the training set and the test set. It commonly used in cross-validation, the function is to randomly select train data and test data from the sample.[2]

3) Save the Print Statement in a Log file

Python's logging module provides a common logging system that can be used by third-party modules or applications. This module provides different log levels and can log in different ways, such as files, HTTP GET/POST, SMTP, Socket, etc., and even implement specific logging methods.[3]

4) Pickle

The pickle module is a module used to persist objects in Python. The so-called persistence of the object, that is, all the information such as the data type, storage structure, and storage content of the object is saved as a file for the next use.

For example, if you save an array as a file by pickle, then when you read the file by pickle next time, you still read an array instead of an array that looks like an array.

Save data to the data1.pkl file via the dump function in the pickle module.

The first parameter is the name of the object to save. The second parameter is the class file object file to which it is written. File must have a write() interface. file can be a file opened in 'w' mode. If protocol>=1, the file object needs to be opened in binary mode. The third parameter is the version of the protocol used for serialization, 0: ASCII protocol, the serialized object is represented by printable ASCII code; (1: old-fashioned binary protocol; new binary protocol introduced in 2:2.3 version, more than before More efficient; -1: Use the highest protocol supported by the current version. Protocols 0 and 1 are compatible with older versions of python. The default value of protocol is 0.) [3]

Core algorithm

1) Linear Support Vector Matrix

Svm is a two-class classifier. So that, it only answers questions that belong to a positive or negative class. The problems to be solved in reality are often multiple types of problems, such as text classification, such as digital recognition.

The learning of svm is actually to find the separation hyperplane, which is to solve the Lagrange factor in the above formula. The ai is the Lagrange factor and N is the number of samples.

$$w^* = \sum_{i=1}^{N} a_i^* y_i x_i$$

$$b^* = y_j - \sum_{i=1}^{N} a_i^* y_i (x_i \cdot x_j)$$

$$\sum_{i=1}^{N} a_i^* y_i (x_i \cdot x_j) + b^* = 0$$

$$f(x) = sign(\sum_{i=1}^{N} a_i^* y_i (x_i \cdot x_j) + b^*)$$

Figure. The Simple SVM model Function

The dual problem of introducing the slack variable is very different from the dual problem that is not introduced a slack variable. The range of the Lagrange factor ai is different. Ai $\geqslant$ 0 is not introduced into the slack variable, and 0 $\leqslant$ ai $\leqslant$ C is introduced The slack variable is eliminated in the mathematical transformation. [4]

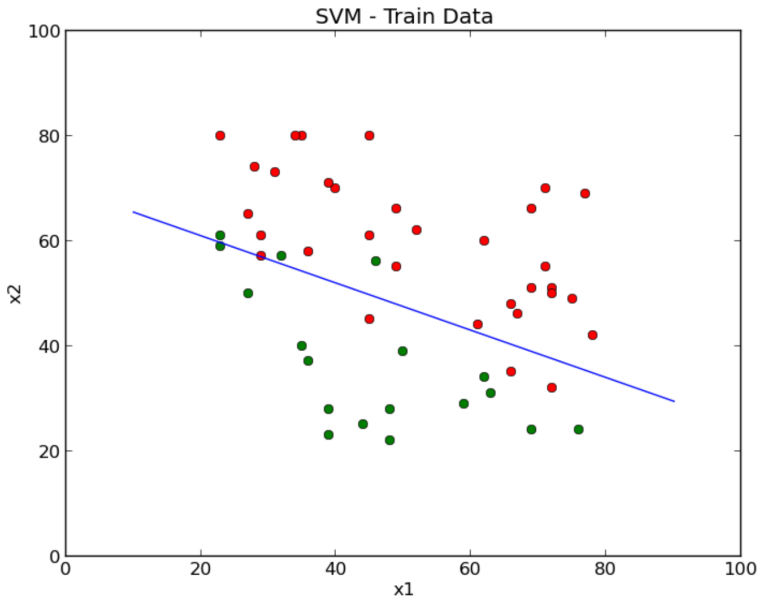

Figure. The two types of samples cannot be completely separated

$$\begin{aligned} \max_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{s.t.} \quad & y_i(w\cdot x_i + b) \geq 1 - \xi_i, \qquad i = 1, 2, \cdots, N \\ & \xi_i \geq 0, \qquad i = 1, 2, \cdots, N \end{aligned}$$

$$\begin{aligned} \min_{a} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N} a_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} a_i y_i = 0 \\ & 0 \leq a_i \leq C, \qquad i = 1, 2, \cdots, N \end{aligned}$$

$$b^* = y_j - \sum_{i=1}^{N} a_i^* y_i (x_i \cdot x_j)$$

Figure. Relaxation variable make the Function Change

2）Nonlinear SVM

Sometimes our data cannot be separated by linear functions, but nonlinearity can be divided. So after a certain mathematical mapping, the data becomes linearly divided.

Let χ be the input space, specifying a mapping function φ(x). If for all x, z∈χ, the kernel function K(x, z) is satisfied. K(x,z)=φ(x)·φ(z). Where φ(x) · φ(z) is the inner

product. With a kernel function, you can talk about data xi mapping to higher dimensions, even infinite dimensions. [5]

$$\min_a \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N} a_i$$

$$\min_a \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i a_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N} a_i$$

$$b^* = y_j - \sum_{i=1}^{N} a_i^* y_i K(x_i \cdot x_j)$$

$$f(x) = sign(\sum_{i=1}^{N} a_i^* y_i K(x_i, x_j) + b^*)$$

Figure. Replace the minimized objective function in the dual problem with a kernel function

3) Parameters meaning

According to "**clf = svm.SVC**" the parameters are meaning: [6]

C: The penalty coefficient C, C of the objective function is larger, the higher the accuracy rate is tested for the training set, but the weaker the generalization ability is, the smaller the C value is, the less the penalty for misclassification is, the fault tolerance is allowed, and the generalization ability is stronger.

Shrinking: helps to calculate the effect

Verbose: show details... is to allow redundant output

Coef0: constant term of the kernel function, useful for poly and sigmoid kernel functions

Degree: the dimension of the polynomial poly kernel function, the default is 3, other kernel functions will ignore

Gamma: not clear, but the effect is obvious after tuning; kernel function parameters of rbf, poly and sigmoid

Tol: the error value of the stop training, the default is 1e-3

Result Analysis

1) Confusion Matrix

When it comes to classification problems, we often need to analyze the experimental results by visualizing the confusion matrix to get the reference ideas. The row is true value and the column is the predicted value.[7]

```
Confusion Matrix:
 [[546   0   4   2   0  13  15   0  11   3]
 [  0 644   2   2   1   5   3   2   8   2]
 [  7  13 508  11   3   4  21   2  37   1]
 [  6   3  18 502   1  26   9   7  35   2]
 [  3   2   2   1 526   5  12   2  22  26]
 [  5   1   3  22   4 426  21   0  32   6]
 [  1   0   2   0   2  12 577   0   5   0]
 [  1   1  15   7   5   3   0 560  10  26]
 [  3  13   7   7   4  36  15   1 480   9]
 [  2   4   2  13  27   9   1  21  19 500]]

Confusion Matrix for Test Data:
 [[ 921    0    4    2    0   11   28    3    9    2]
 [   0 1120    3    1    0    1    5    0    4    1]
 [   7   17  844   14    5   10   34   10   87    4]
 [  12    2   22  813    1   68   10   10   67    5]
 [   3    5    6    3  859    5   31    4   19   47]
 [   7    4    3   21    2  751   40    9   49    6]
 [   5    3    2    0    5   16  921    0    6    0]
 [   0    8   37    9    9    8    1  910    9   37]
 [   4   16    9   11    9   61   18    7  831    8]
 [   6    7    2   17   45   13    0   28   69  822]]
```

Figure. The confusion function

2) GGplot

Ggplot is a landscaping of matplot graphics. From the visual confusion function, we can visually see that 1 has the highest recognition accuracy rate and 5 has the lowest recognition accuracy rate.

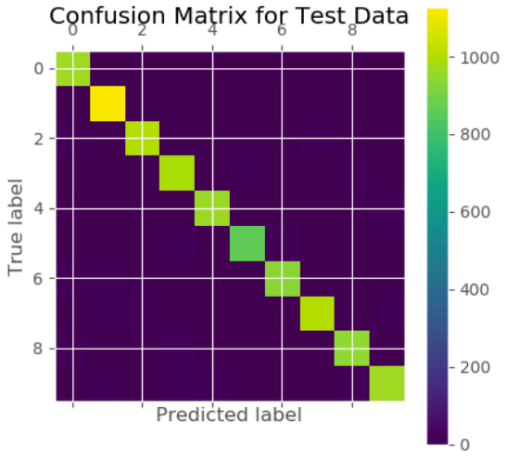

Figure. The Visualization confusion function

3）Conclusion:

	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9
C	0.5	0.1	0.1	0.01	0.01	0.01	0.01	0.8	0.01
gamma	0.05	auto	0.1	0.2	0.3	0.4	0.05	0.05	0.05
tol	0.05	0.01	0.01	0.05	0.05	0.05	0.05	0.1	0.1
Train Accuracy	0.981	0.981	0.9773	0.981	0.981	0.981	0.981	0.98116	0.98116
Test Accuracy	0.9785	0.9783	0.9771	0.9785	0.9785	0.9785	0.9785	0.9785	0.9785

From1. 9 Test of different parameters

Test8 and 9 are the best parameter sets. From Test4 to Test8, we can know that the size of gamma does not affect the correct rate of SVM when the data is stable.

We can conclude by comparing Test2 to Test8. gama=0.1 is an interference data.

And by comparing Test7 and Test9, you can get the conclusion that increasing the value of tol can increase the correct rate slightly.

The Linear Support Vector Classification Accuracy is 0.869. This is as low as we expected.

Reference

[1] “详解 MNIST 数据集 - Liu-Cheng Xu - CSDN 博客.” [Online]. Available: https://blog.csdn.net/simple_the_best/article/details/75267863. [Accessed: 19-Aug-2019].

[2] “sklearn.model_selection.train_test_split 划分训练集和测试集 - I am what i am - CSDN 博客.” [Online]. Available: https://blog.csdn.net/liuxiao214/article/details/79019901. [Accessed: 19-Aug-2019].

[3] “log 输出日志 - 浩子的博客 - CSDN 博客.” [Online]. Available: https://blog.csdn.net/u013851082/article/details/72782943. [Accessed: 19-Aug-2019].

[4] “SVM 支持向量机 - 刘洪江的流水帐.” [Online]. Available: http://liuhongjiang.github.io/tech/blog/2012/12/26/svm/. [Accessed: 19-Aug-2019].

[5] “1.4. Support Vector Machines — scikit-learn 0.21.3 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#svm-classification. [Accessed: 19-Aug-2019].

[6] “sklearn.svm.SVC — scikit-learn 0.21.3 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed: 19-Aug-2019].

[7] “Confusion matrix — scikit-learn 0.21.3 documentation.” [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html. [Accessed: 19-Aug-2019].

[8] “样式美化(matplotlib.pyplot.style.use) - You_are_my_dream 的博客 - CSDN 博客.” [Online]. Available: https://blog.csdn.net/You_are_my_dream/article/details/53464662. [Accessed: 19-Aug-2019].

Appendix

Test1:
SVM Classifier with C=0.5, cache_size=200, coef0=0.0,gamma=0.05,
kernel=poly,max_iter=-1, probability=False, shrinking=True, tol=0.05
SVM Trained Classifier Accuracy: 0.981
Accuracy of Classifier on Test Images: 0.9785

Test2:
SVM Classifier with C=0.1, cache_size=200, class_weight=None, coef0=0.0,
gamma='auto', kernel='poly',max_iter=-1, probability=False , shrinking=True,
tol=0.01
SVM Trained Classifier Accuracy: 0.981
Accuracy of Classifier on Test Images: 0.9783

Test3:
SVM Classifier with C=0.1, cache_size=200, class_weight=None, coef0=0.0,
gamma=0.1, kernel='poly',max_iter=-1, probability=False , shrinking=True,
tol=0.01
Support Vector Classification Accuracy of Classifier on Train Images:
0.9773333333333334
Support Vector Classification Accuracy of Classifier on Test Images: 0.9771

Test4:
SVM Classifier with C=0.01, cache_size=200, coef0=0.0, gamma=0.2,
kernel=poly,max_iter=-1, probability=False, shrinking=True, tol=0.05
SVM Trained Classifier Accuracy: 0.981
Accuracy of Classifier on Test Images: 0.9785

Test5:
SVM Classifier with C=0.01, cache_size=200, coef0=0.0,gamma=0.3,
kernel=poly,max_iter=-1, probability=False, shrinking=True, tol=0.05
SVM Trained Classifier Accuracy: 0.981
Accuracy of Classifier on Test Images: 0.9785

Test6:
SVM Classifier with C=0.01, cache_size=200, coef0=0.0,gamma=0.4,
kernel=poly,max_iter=-1, probability=False, shrinking=True, tol=0.05
SVM Trained Classifier Accuracy: 0.981
Accuracy of Classifier on Test Images: 0.9785

Test7:
SVM Classifier with C=0.01, cache_size=200, coef0=0.0,gamma=0.05,
kernel=poly,max_iter=-1, probability=False, shrinking=True, tol=0.05
SVM Trained Classifier Accuracy: 0.981
Accuracy of Classifier on Test Images: 0.9785

Test8:
   SVM Classifier with C=0.8, cache_size=200, coef0=0.0,gamma=0.05,
kernel=poly,max_iter=-1, probability=False, shrinking=True, tol=0.1
   SVM Trained Classifier Accuracy: 0.9811666666666666
   Accuracy of Classifier on Test Images: 0.9785

Test9:
   SVM Classifier with C=0.01, cache_size=200, coef0=0.0,gamma=0.05,
kernel=poly,max_iter=-1, probability=False, shrinking=True, tol=0.1
   SVM Trained Classifier Accuracy: 0.9811666666666666
   Accuracy of Classifier on Test Images: 0.9785

Test10:
   Linear Support Vector Classification Accuracy of Classifier on Train Images:
0.869
   Linear Support Vector Classification Accuracy of Classifier on Test Images:
0.8742