

学校代码: 10246

学 号: 20210860084

復旦大學

Machine Learning Fundamentals

Week 7 Experimental Class

**Application of SVM in Handwritten Digit Recognition**

院 系: 工程与应用技术研究院

姓 名: 杨仲伟

学 号: 20210860084

指 导 教 师: 范佳媛老师

完 成 日 期: 2020 年 11 月 20 日

# Contents

Application of SVM in Handwritten Digit Recognition .....	1
I. Experiment Purpose .....	3
II. Data Set Source.....	3
III. Implementation Tools.....	3
IV. Experiment Procedure .....	3
V. Data Processing Method.....	4
VI. Algorithm .....	6
1. Algorithm.....	6
2. SVC.....	7
3. Kernel.....	8
VII. Experiment Results.....	10
VIII. Performance.....	11
1. Running time .....	11
2. Memory space .....	11
3. Data set standardization .....	12
IX. References .....	12

# I. Experiment Purpose

Realize the application of SVM classifier in handwritten digit recognition.

# II. Data Set Source

Use the training set in the Mnist-image database to train the model and the test set to evaluate the model.

# III. Implementation Tools

Platform and Notebook: Anaconda Navigator & JupyterLab

Programming Language: python 3.8

# IV. Experiment Procedure

Step 1.

Use python's file reading method to get the training set and test set by reading the binary file downloaded from Mnist-image database website.

Step 2.

Store the data set in four arrays which include train images, training labels, test images, test labels.

Step 3.

For tags 0-9, three test pictures of each label are randomly selected for output. The random number here is selected at random intervals within 20.

Step 4.

Standardize the data set. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data.

Step 5.

Use training set to train the SVC model with the linear kernel, Radial Basis Function (RBF) kernel and custom kernel function. The linear kernel and RBF kernel use the parameters in the scikit-learn library, and the custom kernel function is a new kernel function constructed using the properties of the kernel function.

Step 6.

Use the test set to verify, obtain these models' recognition accuracy score and output.

## **V. Data Processing Method**

1. Download the mnist 784 data set from the official website and save it to the local path: D:\Projects\mnistDataSet









D:\Projects\mnistDataSet			
名称	修改日期	类型	大小
 train-images-idx3-ubyte.gz	2020/11/4 20:34	GZ 压缩文件	9,681 KB
 t10k-images-idx3-ubyte.gz	2020/11/4 20:33	GZ 压缩文件	1,611 KB
 t10k-labels-idx1-ubyte.gz	2020/11/4 20:33	GZ 压缩文件	5 KB
 train-labels-idx1-ubyte.gz	2020/11/4 20:33	GZ 压缩文件	29 KB
 t10k-labels.idx1-ubyte	1998/1/26 23:07	IDX1-UBYTE 文件	10 KB
 t10k-images.idx3-ubyte	1998/1/26 23:07	IDX3-UBYTE 文件	7,657 KB
 train-labels.idx1-ubyte	1996/11/18 23:36	IDX1-UBYTE 文件	59 KB
 train-images.idx3-ubyte	1996/11/18 23:36	IDX3-UBYTE 文件	45,938 KB

FIG. 1 Local path of data set binary files

## 2. Read binary files.

```
def load_images(file_name):
    # 在读取或写入一个文件之前, 必须使用 Python 内置open() 函数来打开它。
    # file object = open(file_name [, access_mode][, buffering])
    # file_name是包含要访问的文件名的字符串值。
    # access_mode指定该文件已被打开, 即读, 写, 追加等方式。
    # 0表示不使用缓冲, 1表示在访问一个文件时进行缓冲。
    # 这里rb表示只能以二进制读取的方式打开一个文件
    binfile = open(file_name, 'rb')
    # 从一个打开的文件读取数据
    buffers = binfile.read()
```

FIG. 2 Python API of reading files

3. According to the data format given by the official website, segment the data to obtain the number of images, rows and columns. Find image information based on offset, and store the image pixel information in an array called images.

### TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803 (2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

FIG. 3 Data Set Type From Website

4. Machine learning estimators might behave badly if the individual features do not more or less look like standard normally distributed data . So standardization processing is necessary.

By calculating the relevant statistical information of the samples in the

training set, independently centering and scaling each feature, and then storing the average and standard deviation to use the transformation on the future data (test set).

## VI. Algorithm

### 1. Algorithm

**Input:**

Binary file of Mnist-image database

**process:**

1. Functions load\_images and load\_labels parse binary files
2. Get training set images, training set labels, test set images, test set labels
3. Randomly extract test set images and print them
4. Standardized processing of training set and test set images
5. Construct a linear kernel vector machine
6. Use the training set to train the model
7. Obtain recognition accuracy through the test set
8. Construct RBF kernel vector machine
9. Use the training set to train the model
10. Obtain recognition accuracy through the test set
11. Take a subset of the data set

12. The function `my_kernel` defines a custom kernel function
13. Construct a support vector machine through a custom kernel function
14. Use a subset of the training set to train the model
15. Use a subset of the test set to obtain recognition accuracy

**Output:**

30 test set images;

Recognition accuracy scores under three kernel functions.

## 2. SVC

In this experiment, support vector machine is used to solve classification problems. The purpose is to find the partition hyperplane with the largest interval. The basic type of SVM is

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad i = 1, 2, \dots, m. \end{aligned}$$

When it comes to high-dimensional feature space, the optimal solution of the model can be expanded by the kernel function of the training sample to obtain the support vector expansion

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T \phi(\mathbf{x}) + b \\ &= \sum_{i=1}^m \alpha_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b \\ &= \sum_{i=1}^m \alpha_i y_i \kappa(\mathbf{x}, \mathbf{x}_i) + b. \end{aligned}$$

Python's scikit-learn library has SVC toolkit, and the following parameters can be customized.

**C: *float, default=1.0***

Regularization parameter.

**Kernel *{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'}, default='rbf'***

**Degree: *int, default=3***

Degree of the polynomial kernel function ('poly'). Ignored by all other kernels.

There are more parameters, but in this experiment, only the kernel function is modified, and the other parameters use default values.

### **3. Kernel**

#### **a. Linear Kernel:**

The implementation of the linear kernel uses the default parameters of the SVC function in the sklearn library.

The linear kernel function is

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

#### **b. RBF Kernel:**

The implementation of the RBF kernel also uses the default parameters of the SVC function in the sklearn library.

The RBF kernel function is



$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

### c. Custom Kernel:

The properties of the kernel function are used to construct a new kernel function: the direct product of two kernel functions is also a kernel function.

$K_1$  is Linear Kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

$K_2$  is Laplacian Kernel

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma}\right) \quad \sigma > 0$$

So, the custom kernel is  $K_1 \otimes K_2$ .

$K_1$  and  $K_2$  kernel functions are implemented through python functions written by myself, showing more details in the programming process of radial basis and linear kernels.

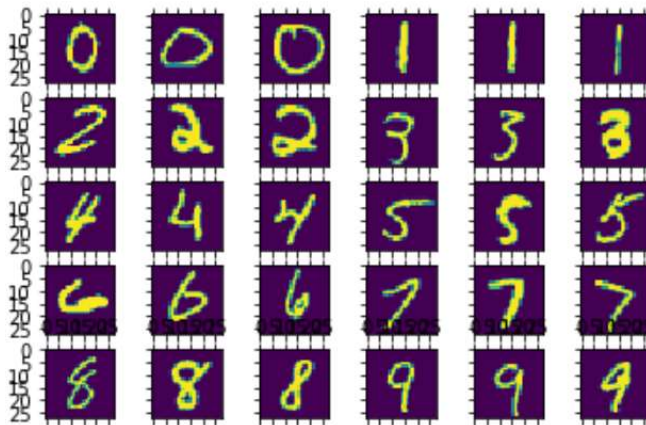
```
# 使用自定义核函数构造向量机，思路采用了这条性质：核函数和另一个核函数的直积也是核函数。
# 其中一个核函数为线性核，另一个核函数为拉普拉斯核函数。
def my_kernel(X,Y):
    # 初始化一个值全为0的数组
    LaplacianK=np.zeros([len(X),len(Y)],dtype=np.float64)
    # 遍历X的每一个值，记为Xi
    for i in range(len(X)):
        # 遍历Y的每一个值，记为Yj
        for j in range(len(Y)):
            # 参数σ设为1
            gamma=1
            # 拉普拉斯核的公式为exp(-||Xi-Yj||/σ)
            LaplacianK[i][j]=np.exp(-np.linalg.norm(X[i]-Y[j])/gamma)
    # 线性核的公式为X*YT
    linearK=np.dot(X,Y.T)
    # 返回自定义核函数：拉普拉斯核和线性核的直积 构成新的核函数
    return LaplacianK * linearK
```

FIG. 4 Custom Kernel Implement

## VII. Experiment Results

Code running results display:

```
Randomly show 3 test images of each label, in order of:  
000111  
222333  
444555  
666777  
888999
```



```
Score of linear kernel SVC:  
0.9293  
Score of rbf kernel SVC:  
0.966  
Score of custom kernel SVC:  
0.13666666666666666
```

FIG. 5 Output of python code in JupyterLab

From the picture, the information shows here:

First, the test images are shown. Each label shows three pictures, output according to the matrix of 000-999, and the corresponding numbers are shown in the figure.

Then the information shows in order:

Use linear kernel function to construct support vector machine, use training set for model training, use test set to measure recognition accuracy, and output the score.

Use Gaussian kernel function to construct support vector machine, use

training set for model training, use test set to measure recognition accuracy, and output the score.

Use a custom kernel function to construct a support vector machine, use the training set for model training, use the test set to measure the recognition accuracy, and output the score.

## VIII. Performance

### 1. Running time

The running time of the program is about 30 minutes, which can be adjusted and optimized by adjusting the SVC parameters.

### 2. Memory space

When using the train set from 60000 samples to perform a custom kernel function, the np array stores float64 type data, occupying 26.8 GiB of space, which exceeds the memory limit of the PC.

```
<ipython-input-2-7152bbaed/a> in my_kernel(X, Y)
    98
    99 def my_kernel(X,Y):
--> 100     return np.dot(X,Y.T)
    101 # 使用自定义核函数构造向量机, SVC的其他参数使用默认值
    102 my_kernel_clf = svm.SVC(kernel=my_kernel)

<__array_function__ internals> in dot(*args, **kwargs)
MemoryError: Unable to allocate 26.8 GiB for an array with shape (60000, 60000) and data type float64
```

FIG. 6 Memory Error when designing custom kernel  
with the whole train set samples

Therefore, in order to make the program run normally, only part of the data is intercepted to test the custom kernel function.

### **3. Data set standardization**

Through the preprocessing of data set standardization, the model training speed is greatly improved. So it is necessary to preprocess data set.

## **IX. References**

1. Mnist-image database download: <http://yann.lecun.com/exdb/mnist/>
2. scikit-learn library installation and API of SVC:  
<https://scikit-learn.org/stable/index.html>
3. Machine Learning-Zhou Zhihua