

# Tutorial of Zero One MNIST (ZOM)

---

## 1 Motivation

---

The ZOM project is funded to help you with your homework HW3. You may applicate this project as a data loader in your Logistic Regression program for the last question.

## 2 Introduction of MNIST & ZOM

---

The MNIST database of handwritten digits 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. The digits have been size-normalized and centered in a fixed-size image.

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.

Web of MNIST is here: [MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges](#)

Only two classis of data in the MNIST dataset are selected for the ZOM project, data with label 0 and label 1. To be convenience, these data have be reformed as python list and saved as pickle file.

Some examples are shown in the section 4.

## 3 Requirement and Contents

---

### 3.1 Requirement

- Python3
- Numpy
- Opencv
- scikit-learn

### 3.2 Contents

- `readme.md`: readme file
- `readme.pdf`: readme file
- `train_dataset.pkl`: train dataset, 12665 data is involved
- `test_dataset.pkl`: test dataset, 2115 data is involved
- `image0.png`: image example of MNIST data with label 0
- `image1.png`: image example of MNIST data with label 1

## 4 Tutorial

---

- Install related python package.

**Attention: This tutorial encourages you to use Pypi to maintain the python package!**

```
# opencv: an image processing package
pip install opencv-python

# numpy
pip install numpy

# sklearn
pip install scikit-learn # you need a good net condition
```

- Save python object as a pickle file.

```
import pickle
# pickle helps to save and load your python object
x = 1
with open('x.pkl', 'wb+') as file:
    pickle.dump(x, file)
```

- Read python object from a pickle file.

```
import pickle
# pickle helps to save and load your python object
with open('x.pkl', 'rb+') as file:
    x = pickle.load(file)
print(x)
```

- Read one data in the train dataset, print its label and show its image.

```
import pickle
import cv2 as cv # import opencv

with open('train_dataset.pkl', 'rb+') as file:
    train_dataset = pickle.load(file)

index = 6
image, label = train_dataset[index]
print(label)
cv.imshow('image', image) # image of MNIST dataset is very small(28*28), find it!
cv.waitKey() # press one key on your keyboard to finish the program,
              # we use it to keep the image showing
```

image like this will be shown



- Flatten the train data and create the train matrix with size  $M \times N = (28 * 28) \times 12665$ .

(This code may take some minutes to run.)

```
import pickle
```

```

import numpy as np

with open('train_dataset.pkl', 'rb+') as file:
    train_dataset = pickle.load(file)

train_matrix = None
for img, label in train_dataset:
    if train_matrix is None:
        train_matrix = np.reshape(img, [-1, 1])
    else:
        data = np.reshape(img, [-1, 1])
        train_matrix = np.concatenate([train_matrix, data], axis=-1)

print(np.shape(train_matrix))
# (784, 12665)

```

- Logistic regression with scikit-learn(sklearn) package

```

import pickle
import numpy as np
from sklearn import linear_model

# 1. read train dataset and reformed it as matrix(image) and vector(label)
with open('train_dataset.pkl', 'rb+') as file:
    train_dataset = pickle.load(file)

train_matrix = None
train_label = []
for img, label in train_dataset:
    if train_matrix is None:
        train_matrix = np.reshape(img, [-1, 1])
    else:
        data = np.reshape(img, [-1, 1])
        train_matrix = np.concatenate([train_matrix, data], axis=-1)
    train_label.append(label)

# 2. train the LogReg model
model = linear_model.LogisticRegression()
model.fit(train_matrix.T, train_label)

# 3. read test dataset and reformed it as matrix(image) and vector(label)
with open('test_dataset.pkl', 'rb+') as file:
    test_dataset = pickle.load(file)

test_matrix = None
test_label = []
for img, label in test_dataset:
    if test_matrix is None:
        test_matrix = np.reshape(img, [-1, 1])
    else:
        data = np.reshape(img, [-1, 1])
        test_matrix = np.concatenate([test_matrix, data], axis=-1)
    test_label.append(label)

# 4. predict with trained model
predict = model.predict_proba(test_matrix.T)
acc = model.score(test_matrix.T, test_label)

```

```
print('accuracy is {:.2f} %'.format(acc*100))
```

```
# accuracy is 99.76 %
```