ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

BÀI THI MÔN: Machine Learning

Hình thức thi: Bài tập lớn

Đề tài:08

Giảng viên hướng dẫn : Cao Văn Chung

Sinh viên thực hiện : Nguyễn Quang Quân

Nguyễn Văn Thắng

Phạm Văn Tuấn

Lớp : K65A2 – Toán Tin

HÀ NỘI 2022

Lời nói đầu

Machine Learning là môn học rất quan trọng đối với sinh viên, đặc biệt là sinh viên ngành Toán Tin. Đây là môn học mang lại những định hướng cần thiết và quan trọng cho sinh viên khi còn trên ghế giảng đường đại học cũng như khi ra trường. Dựa vào các nguyên tắc và tư duy sẽ giúp cho mỗi sinh viên khi đứng trước một vấn đề nghiên cứu sẽ nhanh chóng tìm ra được giải pháp để giải quyết vấn đề.

Bài tập lớn này do nhóm em viết nhắm tổng kết, rút ra những bài học và để hiểu hơn những lý thuyết được thầy giảng dạy trên lớp. Qua đây nhóm em xin gửi lời cảm ơn chân thành đến thầy Cao Văn Chung đã tận tình dạy bảo chúng em.

Nội dung tiểu luận có thể không tránh khỏi một vài lỗi. Rất mong thầy thông cảm và cho những nhận xét và đánh giá để bài tiểu luận của chúng em được hoàn chỉnh hơn.

Chúng em xin chân thành cảm ơn!

I. Giới thiệu đề tài và dữ liệu

Cho một tập dữ liệu về hình ảnh chữ số viết tay (gồm 60000 mẫu trong tập training và 10000 trong tập test).

Dữ liệu được lấy từ: http://yann.lecun.com/exdb/mnist/ , các tệp tin được nén và được để trong 4 tệp với thông tin như dưới đây:

train-images-idx3-ubyte: training set images (dữ liệu ảnh train)

train-labels-idx1-ubyte: training set labels (dữ liệu nhãn ứng với ảnh train)

t10k-images-idx3-ubyte: test set images (dữ liệu ảnh test)

t10k-labels-idx1-ubyte: test set labels (dữ liệu nhãn ứng với ảnh test)

Dữ liệu ảnh được các chữ số viết tay ở đây được lưu liên tiếp nhau và không theo định dạng ảnh, cụ thể trong cấu trúc file như sau:

Cấu trúc file **train-images-idx3-ubyte** chứ dữ liệu ảnh training:

[thứ tự byte]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images (số ảnh = 60000)
0008	32 bit integer	28	number of rows (số dòng mỗi ảnh)
0012	32 bit integer	28	number of columns (số cột mỗi ảnh)
0016	unsigned byte	??	cường độ pixel thứ nhất
0017	unsigned byte	??	cường độ pixel thứ hai
....			
xxxx	unsigned byte	??	cường độ pixel cuối cùng

Cường độ Pixels được sắp xếp cạnh nhau thành dòng. Giá trị cường độ Pixel là từ 0 đến 255 (1byte) nhưng để ngược: 0 là background (trắng),và 255 là foreground (đen), tuy nhiên khi in ảnh ra màn hình thì điều này không quan trọng.

Cấu trúc file **train-labels-idx1-ubyte** chứa nhãn của cách ảnh training:

[thứ tự byte]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items (số nhãn ảnh = 60000)
0008	unsigned byte	??	label cho ảnh 1
0009	unsigned byte	??	label cho ảnh 2
....			
xxxx	unsigned byte	??	label cho ảnh cuối

Ở đây nhãn cho ảnh là số nguyên từ 0 đến 9 (ứng với chữ số trong ảnh).

Cấu trúc của tệp dữ liệu ảnh test (train-images-idx3-ubyte) và nhãn ảnh test

(t10k-labels-idx1-ubyte) tương tự như với dữ liệu training, số lượng ảnh là 10000.

II. Rút gọn số chiều dữ kiệu

1. Lấy dữ liệu

Do tệp dữ liệu ở định dạng nén nên cần đoạn chương trình giải nén.

Đầu tiên, ta lấy dữ liệu vào trong chương trình. Đoạn códe dưới đọc tệp dữ liệu.

```
import os
import numpy as np

data_path = '/home/nguyenquan/Desktop/Baitaplon_MachinLearning'

train_images_path = os.path.join(data_path, 'train-images-idx3-ubyte.gz')
train_labels_path = os.path.join(data_path, 'train-labels-idx1-ubyte.gz')

test_images_path = os.path.join(data_path, 't10k-images-idx3-ubyte.gz')
test_labels_path = os.path.join(data_path, 't10k-labels-idx1-ubyte.gz')
```

Xây dựng phương thức đọc dữ liệu từ tệp gzip, giải nén và đưa về định dạng là một dãy ảnh (một dãy ma trận nguyên).

```python
def get_mnist_data (images_path, labels_path, num_images, shuffle = False, _is=True, image_size= 28):

    #read data
    import gzip       # to decompress gz (zip) file

    # open file training to read training data
    f_images = gzip.open(images_path, 'r')

    # skip 16 first bytes because these are not data, only deader infor
    f_images.read(16)


    # general: read num_images data samples if this parameter is set;
    # if not, read all (60000 training or 10000 test)
    real_num = num_images if not shuffle else (60000 if _is else 10000)

    # read all data to buf_images (28x28xreal_num)
    buf_images = f_images.read(image_size * image_size * real_num)

    # images
    images = np.frombuffer(buf_images, dtype=np.uint8).astype(np.float32)
    images = images.reshape(real_num, image_size, image_size,)

    # Read labels
    f_labels = gzip.open(labels_path,'r')
    f_labels.read(8)

    labels = np.zeros((real_num)).astype(np.int64)

    # rearrange to correspond the images and labels
    for i in range(0, real_num):
        buf_labels = f_labels.read(1)
        labels[i] = np.frombuffer(buf_labels, dtype=np.uint8).astype(np.int64)


    # shuffle to get random images data
    if shuffle is True:
        rand_id = np.random.randint(real_num, size=num_images)

        images = images[rand_id, :]
        labels = labels[rand_id,]

    # change images data to type of vector 28x28 dimentional
    images = images.reshape(num_images, image_size * image_size)
    return images, labels
```

Hàm xây dựng đổ dữ liệu vào biến ***train_images*** (ma trận có 60000 hàng và 28x28 cột ứng với kính thước của ảnh) và biến ***train_labels*** (1 mảng có các giá trị từ 0-9)

Gọi phương thức đọc dữ liệu để kiểm tra xem đọc đúng ko.

```
train_images, train_labels = get_mnist_data(train_images_path, train_labels_path, 60000)
test_images, test_labels = get_mnist_data(test_images_path, test_labels_path, 10000)
print(train_images.shape, train_labels.shape)
print(test_images.shape, test_labels.shape)
```

Kết quả nhận được.

```
[Running] python -u "/home/nguyenquan/Desktop/Baitaplon_MachinLearning/de_tai_8.py"
(60000, 784) (60000,)
(10000, 784) (10000,)
```

2. Rút gọn tập dữ liệu và hiểu thị trực quan

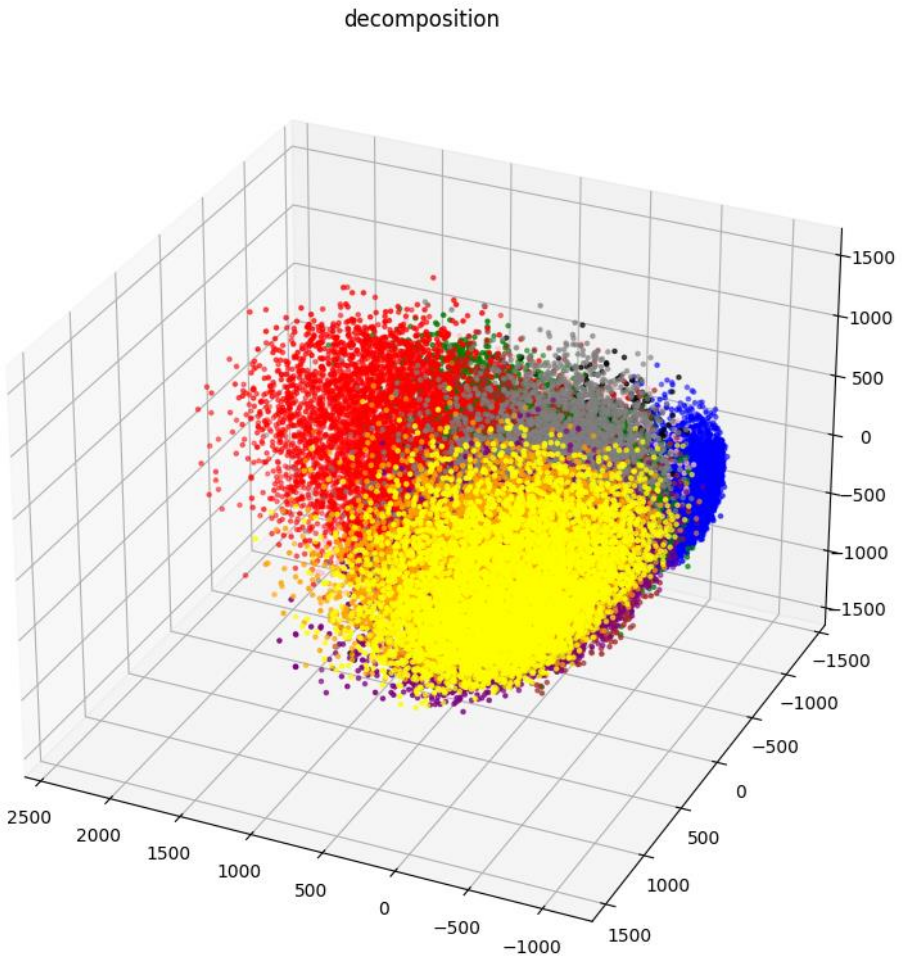
Để rút gọn số chiều dữ liệu, ta sử dụng PCA trong **sklearn.**

Để hiện thị trực quan các phân lớp dữ liệu dạng 3D và 2D ta rút gọn số chiều của dữ liệu từ 784 về 3 chiều và 2 chiều. Việc rút gọn dữ liệu và hiển trị trực quan dữ liệu được thực hiện qua đoạn code bên dưới.

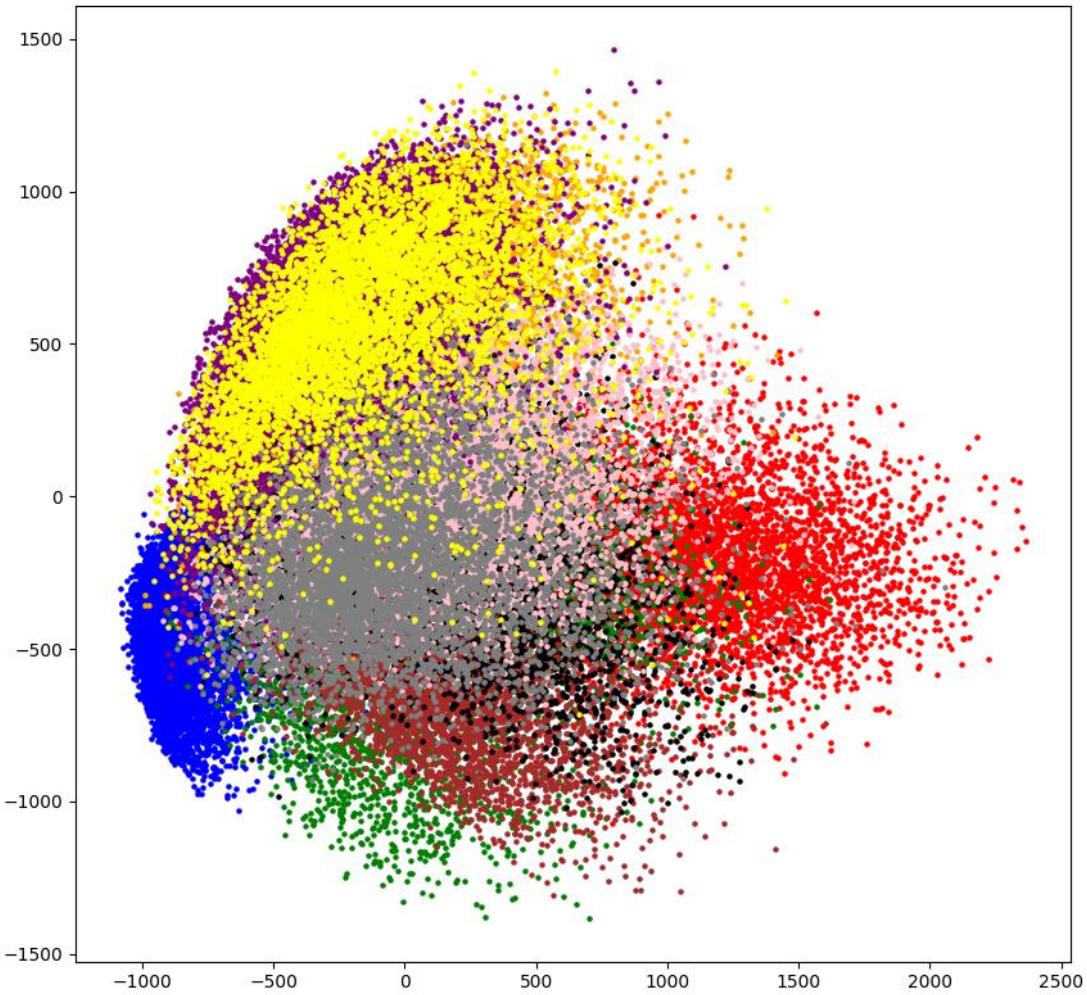
```python
from sklearn.decomposition import PCA

pca3D = PCA(n_components=3)
pca3D.fit(train_images)
pac_transform3D = pca3D.transform(train_images)

fig3 = plt.figure()
fig3.set_size_inches(10,10)
ax = plt.axes(projection='3d')
colors = ['red','blue','green','brown', 'orange', 'black', 'pink','purple','gray','yellow']

for label in range(10):
    ax.scatter(pac_transform3D[train_labels==label,0],
               pac_transform3D[train_labels==label,1],
               pac_transform3D[train_labels==label, 2],
               s= 5, c = colors[label])

ax.set_title('decomposition')
plt.show()


pca2D = PCA(n_components=2)
pca2D.fit(train_images)
pac_transform2D = pca2D.transform(train_images)

fig2 = plt.figure()
fig2.set_size_inches(10,10)
for label in range(10):
    plt.scatter(pac_trans
               pac_trans
               s=5, c = colors[label])
plt.show()
```

```
(variable) colors: list[str]
```

Đồ thị nhận được sau khi giảm số chiều về 3.



Đồ thị sau khi giảm số chiều về 2.



IV Xây dựng chương trình sử dụng mô hình Naive Bayes

Ta nhận thấy tệp dữ liệu image có mẫu là các biến rời rạc (giá trị của mỗi điểm ảnh là một số nguyên nằm trong khoảng 0-255) nên ta không thể sử dụng mô hình Gaussian Naïve Bayes (vì Gaussian Naïve Bayes được sử dụng khi các mẫu là các biến liên tục).

Và ta cũng thấy rằng giá trị của các biến không phải là giá trị nhị phân (giá trị nhận là 0,1) nên ta sẽ sử dụng mô hình Multinomial Naive Byes.

Ta sử dụng phương thức MultinomialNB() trong thư viện **sklearn** để giải quyết bài toán.

```
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(train_images, train_labels)


predictTest = clf.predict(test_images)
```

V. So sánh độ chính xác của các mô hình

Ta tính độ chính xác của mô hình qua bốn cách sau:

- Accuracy : tỷ lệ giữa số lượng dự đoán đúng và tổng số mẫu được dự đoán.
- Confusion matrix : là ma trận dùng để đánh giá hiệu suất của thuật toán được sử dụng
- Recall : Recall là tỉ lệ giữa số lượng các trường hợp được dự đoán là positive (dự đoán đúng) và thực tế cũng là positive (ground truth), trên tổng số các trường hợp

  positive trong dữ liệu
- Precision : cho biết tỉ lệ số lượng các trường hợp được phân loại đúng trong số các trường hợp được phân loại là positive.

Ta sử dụng thư viện **sklearn** để tính các mô hình.

Code tính độ chính xác của mô hình Multinomial Naive Bayes.

```
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score

accuracyMultinomial = accuracy_score(test_labels, predictTest)
confusionMatrixMultinomial= confusion_matrix(test_labels, predictTest)
precisionMultinomial = precision_score(test_labels, predictTest, average='macro')
recallMultinomial = recall_score(test_labels, predictTest, average='macro')

print("Accuracy:", accuracyMultinomial)
print("Confusion matrix:\n", confusionMatrixMultinomial)
print("Precision:", precisionMultinomial)
print("Recall:", recallMultinomial)
```

Ta thu được kết quá.

```
Accuracy: 0.8365
Confusion matrix:
 [[ 912    0    2    6    1    8   14    1   36    0]
 [   0 1061    5    9    0    2    6    0   51    1]
 [  15   11  858   24   10    3   33   11   66    1]
 [   4   11   34  851    1   21    7   14   40   27]
 [   2    2    6    0  732    0   25    1   38  176]
 [  23   11    6  107   18  590   17    6   78   36]
 [  17   13   17    1    7   25  860    0   18    0]
 [   1   21   11    5   19    0    1  861   40   69]
 [   6   26   13   54   14   27    8    9  777   40]
 [   6    7    3   10   66   10    0   17   27  863]]
Precision: 0.8433162997126132
Recall: 0.8334531845906966
```