数据挖掘期末实验报告

作者：方佳谋

学号：15331074

1. 题目

- Data Mining 2018 Final-exam, SDCS, SYSU

2. 实验

2.1 数据分析

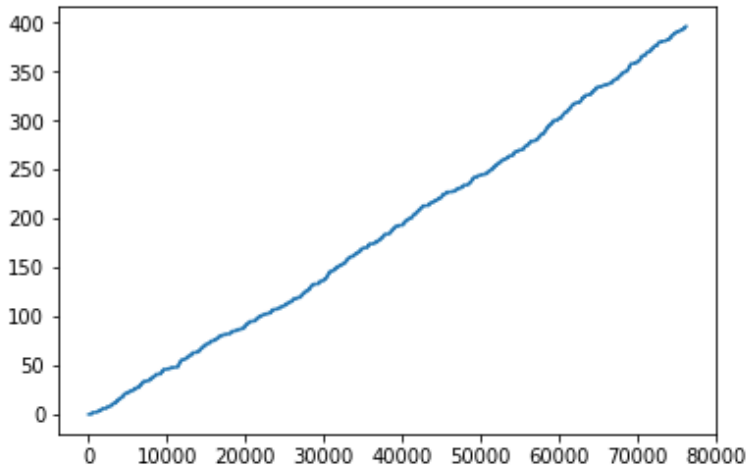
- 题目所给训练数据集大小为76240X6812

```
In [6]: print("————————————————1. 加载训练数据————————————————")
        training_feat, training_label = loadData(file_train)
```

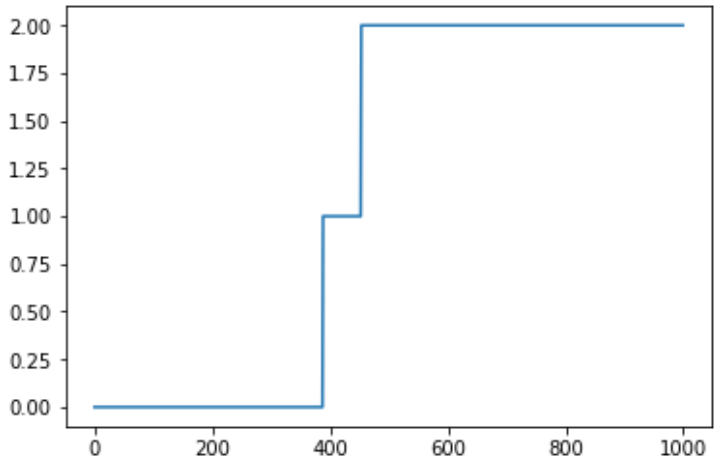
```
————————————————1. 加载训练数据————————————————
(76240, 1)
(76240, 6812)
```

- 分析数据集中的数据分布情况

```
In [7]: plt.plot(training_label)
        plt.show()
```

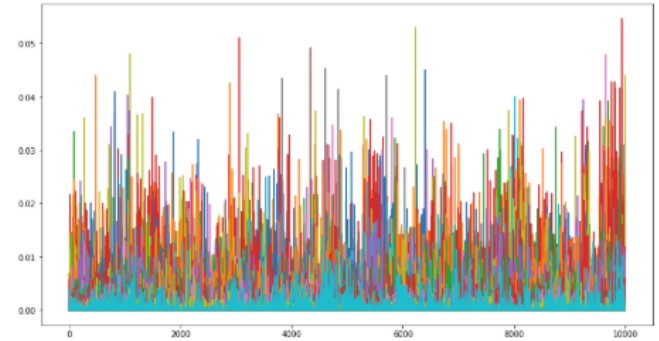


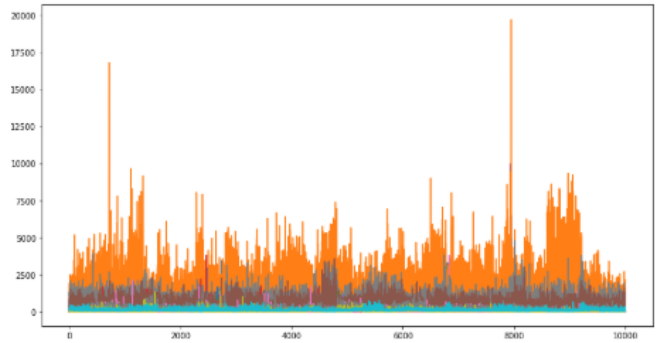
```
In [8]: plt.plot(training_label[0:1000])
        plt.show()
```

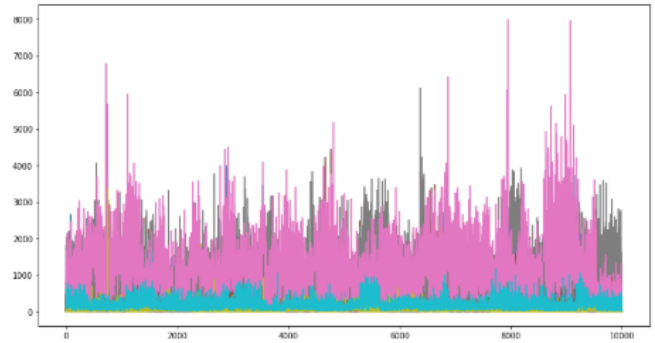


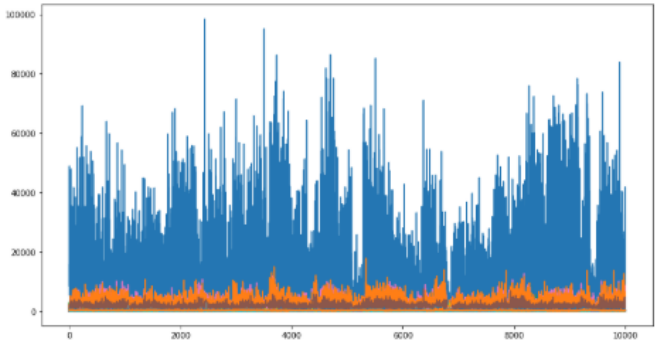
发现给定训练集的数据标签是成簇分布的，每一类标的数据聚在一块。对于训练，随机分布的数据样本会更好。

- 训练集特征的取值范围情况
  通过打印分析训练集特征的取值范围情况，发现特征列[6300:6820]的取值范围远远大于[0:6300]之间的取值。







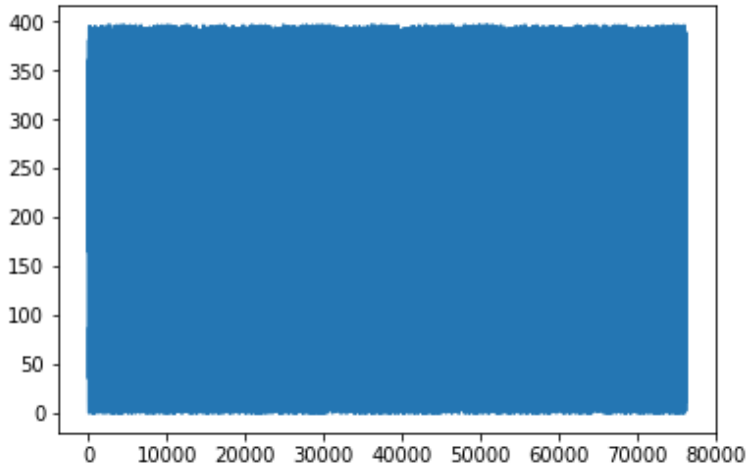


2.2 数据预处理

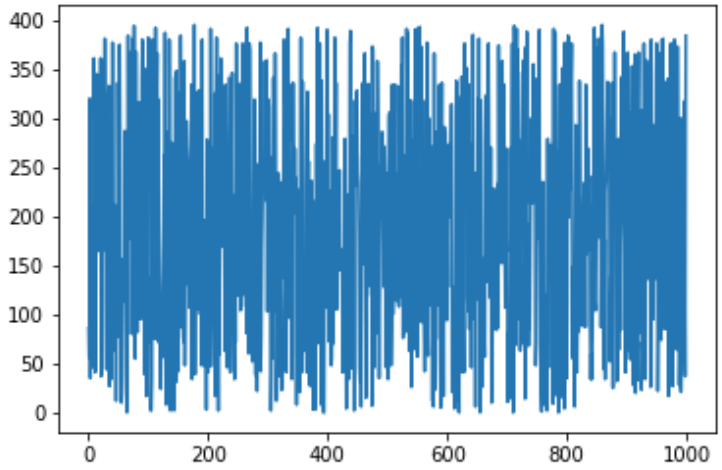
- Shuffle训练数据集

```
print("------------------2. 打乱训练数据---------------")
training_data = np.hstack([training_feat, training_label])
np.random.shuffle(training_data)
X_train = training_data[:, :-1]
y_train = training_data[:, -1]
```

In [10]:
```
plt.plot(y_train)
plt.show()
```



In [12]:
```
plt.plot(y_train[0:1000])
plt.show()
```



- 均值归一化
  对训练集的每个特征进行均值归一化，降低部分特征取值范围远大于其他特征从而对训练带来的影响。

```
print("------------------3. 数据均值归一化---------------")
x_mean = X_train.mean(axis=0)
x_std = X_train.std(axis=0)
X_train = (X_train-x_mean)/x_std
```

2.3 分类算法

2.3.1 MLP

鉴于数据的维度较大，首先考虑神经网络模型，采用多层感知机（MLP）

实验采用sklearn提供的MLPClassifier

MLP的优缺点

- 优点：
  a. 可以学习得到非线性模型
  b. 使用 `partial_fit` 可以学习得到实时模型（在线学习）
- 缺点：
  a. 具有隐藏层的 MLP 具有非凸的损失函数，它有不止一个的局部最小值。 因此不同的随机权重初始化会导致不同的验证集准确率
  b. MLP 需要调试一些超参数，例如隐藏层神经元的数量、层数和迭代轮数
  c. MLP 对特征归一化很敏感.

2.3.2 MLP超参数设置

到目前为止的超参数调优过程，显示出最佳分类效果的超参数组合是：

- solver：权重优化采用随机梯度下降
- hidden_layer_sizes:采用单隐层，该层具有1600个神经元
- activation:激活函数采用 `relu` ，实验证明 `logistic` 的效果略输于 `relu`
- batch_size:随机梯度下降过程，分批大小设置为128，实验证明设置为258的效果与128差不多，设置为512的效果则下降
- alpha: L2正则项的参数，实验证明 `alpha` 为默认值（0.0001）的效果不如 `alpha=0.001` 且继续增大 `alpha` 将导致分类效果变差
- learning_rate_init：初始学习率，实验证明 `learning_rate_init=0.1` 的效果不如 `learning_rate_init=0.01` 且继续减小 `learning_rate_init` 不会得到提升。
- max_iter：实验证明 `max_iter=60` 已经足够学习到所有特征

```
mlp = MLPClassifier(solver='sgd',learning_rate_init  = 0.01,
        alpha=0.001,learning_rate='adaptive',max_iter=60,
        shuffle = True,activation='relu',
        batch_size = 128,verbose=True,
        hidden_layer_sizes=(1600,))
```

2.3.3 分批训练（或在线学习）

```
    max_iter = 50
    num_classes = 397
    split_size = 4765
    split_times = 16
    classes = np.arange(num_classes)
    # mlp.classes_ = classes
    for i in range(max_iter):
        for index in range(split_times):
            X_part_train, y_part_train = readPartialData(X_train,y_train,m_x, index,split_size)
            if index == 0:
                mlp = mlp.partial_fit(X_part_train,y_part_train,classes)
            else:
                mlp = mlp.partial_fit(X_part_train,y_part_train)
            print("iter :" + str(i) + ",split time: " + str(index) + ", accuracy",
                mlp.score(X_part_train,y_part_train))
```

2.3.4 算法复杂度

- 训练样本数: n
- 样本特征数: m
- 隐藏层个数: k
- 每层包含神经元个数:h
- 输出神经元个数：o
- 迭代次数: i

反向传播的时间复杂度是

$$bt = O(n * m * h^k * o * i)$$

其中 i 是迭代次数。根据2.3.3中的训练过程，训练过程总的时间复杂度是

$$total = O(max\_iter * split\_times * split\_size * bt)$$

2.3.5 内存需求

在本次实验过程中，因为给定的训练集（矩阵）规模较为庞大，若是将数据全部加载到内存，则一般的笔记本电脑甚至台式机都会因为资源占用过多而导致训练过程极为缓慢，因此，在2.3.3中，给出了在线学习的例子，每次只使用部分的数据样本，多次迭代训练，这样可以缓解内存紧张。一般$8G$的内存可以无障碍地进行训练模型。

附

- 源代码