Homework 1

该报告分为三个部分：

- 生成数据
- 判别模型和生成模型
- 实验

本实验中的数据直接可在运行过程中产生

使用方法：

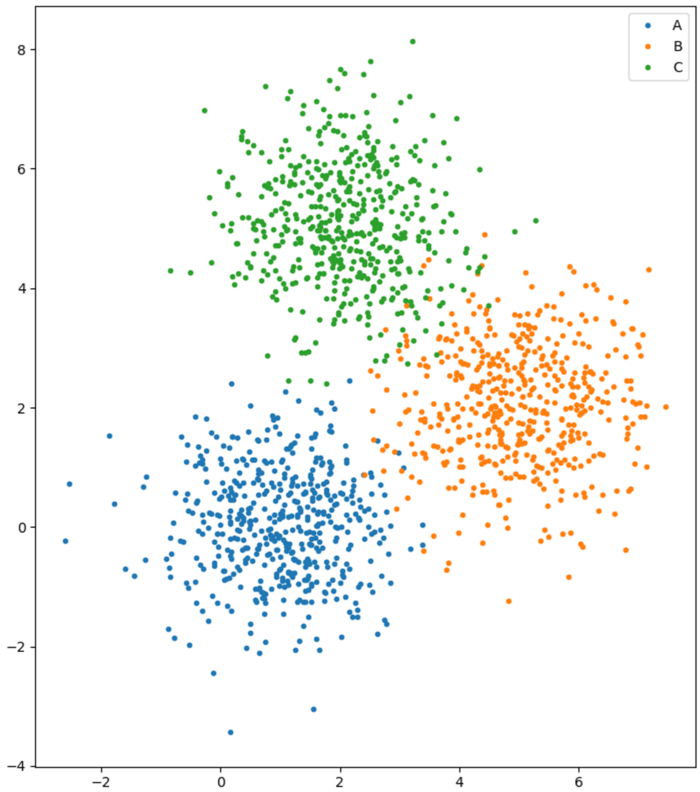
```
python source.py
```

1. 生成数据

1.1 生成结果展示

下图为三个二维高斯分布的生成数据，每个高斯分布的采样个数为500。高斯分布参数如下。

```
mean = np.array([[1,0],[5,2],[2,5]])
cov = np.array([[1,0],[0,1]])
```



1.2 生成过程

为了结果展示更加美观，采用了二维高斯分布的数据，根据修改三个高斯分布的参数决定数据点分布。可变参数有`mean`，`cov`，`sample`，均存放在`gaussian`类中。

生成二维高斯分布的方法是：

```
G = np.random.multivariate_normal(mean,cov,sample)
```

再总体分别调用高斯类生成附有标签的三个高斯分布数据：

```
gaussian_A = gaussian(mean[0],cov,sample,label[0])
gaussian_B = gaussian(mean[1],cov,sample,label[1])
gaussian_C = gaussian(mean[2],cov,sample,label[2])
```

最后**将生成的数据打乱**，存放在`dataset.data`中（也可以直接运行时保存进缓存供后续使用），下面是数据样例：

```
[[0.574497243381892, -0.7619553963463503, 'A'], [1.1583332125233643,
6.329496391598353, 'C'], [2.4675921801423817, 5.711035455657666, 'C'],
[5.356381752478349, 1.8233317513288656, 'B'], [5.647717278495129,
2.3167081475324847, 'B']]
```

2. 判别模型和生成模型

先**数据预处理**，将1500个数据采样点打乱，并将其1/5分为测试集，4/5分为训练集：

```
def data_preprocessing(dataset):
    random.shuffle(dataset)
    size = len(dataset)
    test_data, training_data = dataset[:size//5], dataset[size//5:]
    return test_data, training_data
```

对于上述三个二维高斯分布（分别500个采样点）的数据而言，测验三次（三次数据不同/分布相同），生成模型和判别模型的**准确度**如下：

```
Generative model, accuracy:  0.98
Discriminative model, accuracy:  0.9766666666666667
```

```
Generative model, accuracy:  0.98
Discriminative model, accuracy:  0.9766666666666667
```

```
Generative model, accuracy:  0.9866666666666667
Discriminative model, accuracy:  0.9866666666666667
```

2.1 生成模型

对于上述三个二维高斯分布（分别500个采样点）的数据而言，生成模型的原理即先假定有三个高斯分布在数据点中，使用训练集训练，可以不使用随机梯度下降，直接使用多类问题的解。下面是二类问题的解法：

$$\mu_1 = \frac{1}{N_1}\sum_{i=1}^{N} t_n \mathbf{x}_n$$

$$\mu_2 = \frac{1}{N_2}\sum_{i=1}^{N} (1-t_n) \mathbf{x}_n$$

$$\Sigma = \mathbf{S} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2$$

$$\mathbf{S}_i = \frac{1}{N_i}\sum_{n\in C_i} (\mathbf{x}_n - \mu_1)(\mathbf{x}_n - \mu_1)^T$$

$$a_k(\boldsymbol{x}) = \boldsymbol{w}_k^{\top}\boldsymbol{x} + w_{k0}$$

其中，$\boldsymbol{w}_k = \Sigma^{-1}\boldsymbol{\mu}_k$，$w_{k0} = -\frac{1}{2}\boldsymbol{\mu}_k^{\top}\Sigma^{-1}\boldsymbol{\mu}_k + \ln p(C_k)$

2.2 判别模型

之前上过`cs231n`的课，并**独立完成**了`assignment1`中的`linear classifier`，所以直接使用了其中的`linear_classifier`函数作为构造随机梯度下降法的线性分类器的基础。主要分为`train`、`loss`、`predict`三个函数：

- `train`：使用随机梯度下降法训练，并且采用了`batch_size`方法
- `loss`：使用**最小均方误差** $E(\boldsymbol{W}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}(y_k(\boldsymbol{x}_n) - t_{nk})^2$ 作为损失函数，梯度类似最大似然估计的梯度 $\nabla\ell(\boldsymbol{w}) = \sum_{n=1}^{N}(t_n - y_n)\boldsymbol{x}_n$
- `predict`：仿射变换后使用`sigmoid`函数映射到`[0,1]`，最大取值的`index`为预测标签

`learning_rate`和`regularization_strength`均使用默认值并没有调节过程：

```
learning_rate=1e-3, reg=1e-5, num_iters=2000, batch_size=200
```

2.3 比较

这个数据集上，两者的准确率几乎相同：

- 问题很简单并且原数据集边界清晰
- 判别模型微小逊色是由于在理想高斯分布采样，生成模型本来预测就更容易，而且并没有对学习率和正则化系数进行调节；但差距微乎其微，可以忽略

3 实验

进行了以下实验，判断不同因素对两个模型的影响：

- 采样点
- 重叠

3.1 采样点的影响

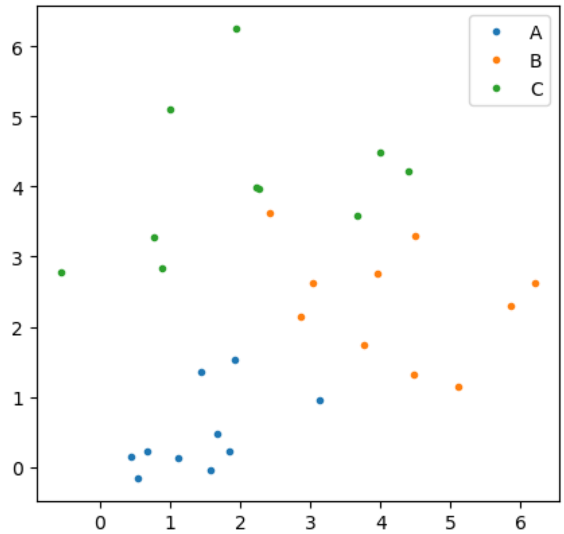
设置了五个不同的sample个数，跑出对应结果：

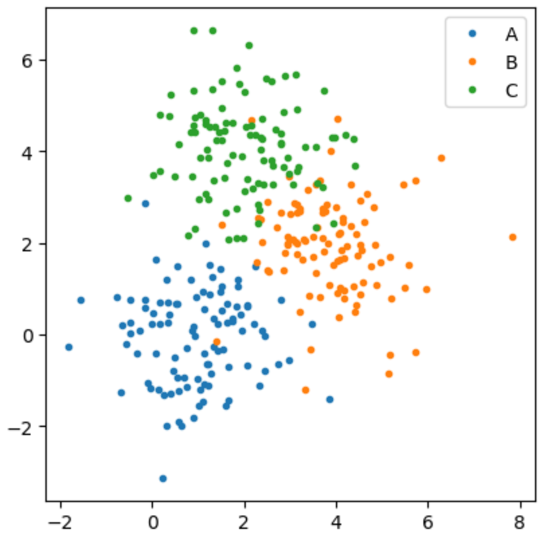
```
sample points: 10
Generative model, accuracy:  1.0
Discriminative model, accuracy:  0.8333333333333334
sample points: 50
Generative model, accuracy:  0.8666666666666667
Discriminative model, accuracy:  0.8333333333333334
sample points: 100
Generative model, accuracy:  0.9166666666666666
Discriminative model, accuracy:  0.9166666666666666
sample points: 500
Generative model, accuracy:  0.95
Discriminative model, accuracy:  0.9566666666666667
sample points: 1000
Generative model, accuracy:  0.9283333333333333
Discriminative model, accuracy:  0.93
```

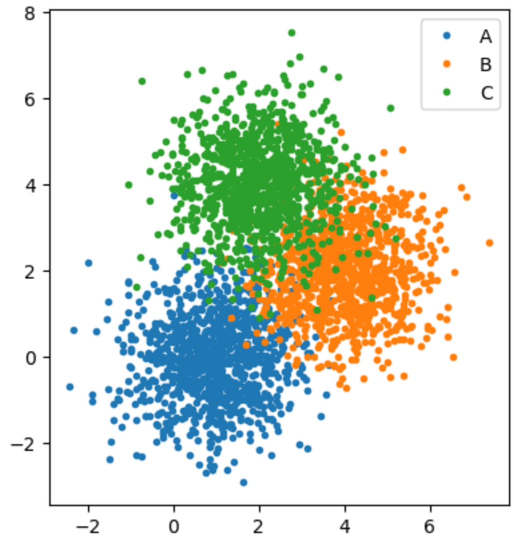
下面三个图片是采样点数为10,100,1000时的数据分布图像。有部分重叠，为了防止两个模型准确率都为1。

可以发现**采样点个数对该情况下的判别式模型影响更大**，推测原因是：

- 此时用的是理想高斯分布，生成模型更占优势
- 判别模型的超参数不好







3.2 重叠的影响

固定采样点数为500，不改变协方差矩阵（为了符合生成模型的实现基础），改变高斯分布的均值，以达到重叠的效果。

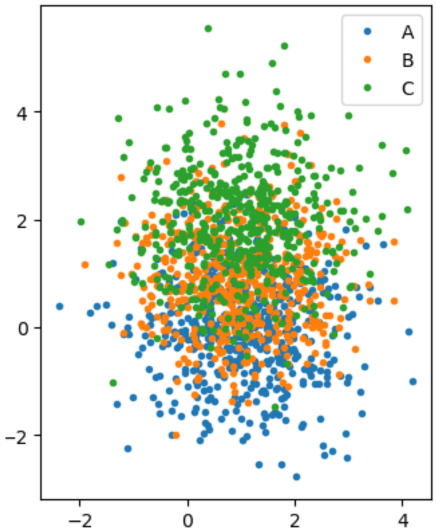
```
[[1 0]
 [1 1]
 [1 2]]
Generative model, accuracy:  0.6133333333333333
Discriminative model, accuracy:  0.61
[[1 0]
 [2 3]
 [3 2]]
Generative model, accuracy:  0.7733333333333333
Discriminative model, accuracy:  0.7666666666666667
[[1 0]
 [4 2]
 [2 4]]
Generative model, accuracy:  0.9266666666666666
Discriminative model, accuracy:  0.92
[[1 0]
 [5 2]
 [2 5]]
Generative model, accuracy:  0.9766666666666667
Discriminative model, accuracy:  0.96
```

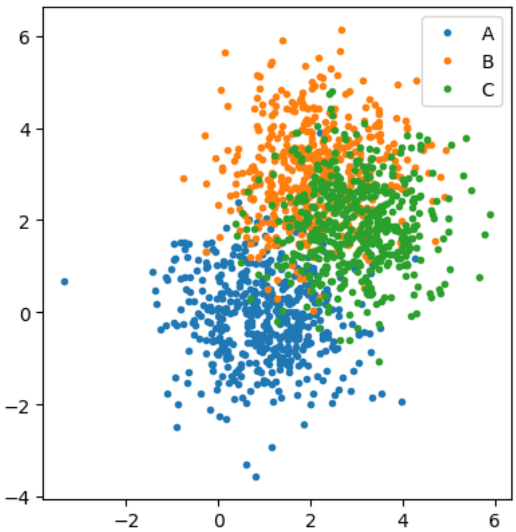
```
[[1 0]
 [6 2]
 [2 6]]
Generative model, accuracy:  0.9866666666666667
Discriminative model, accuracy:  0.98
```

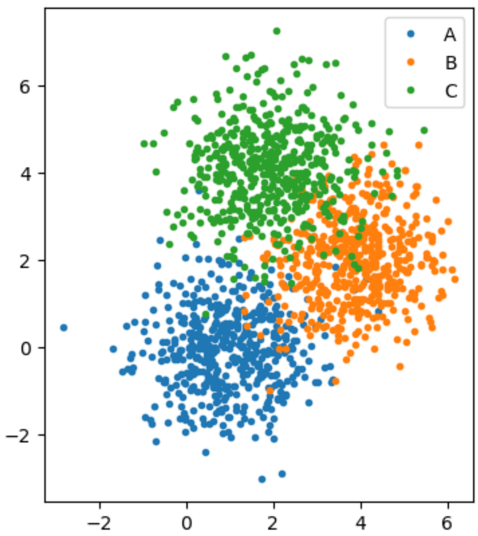
上面是两个模型的输出结果，发现：

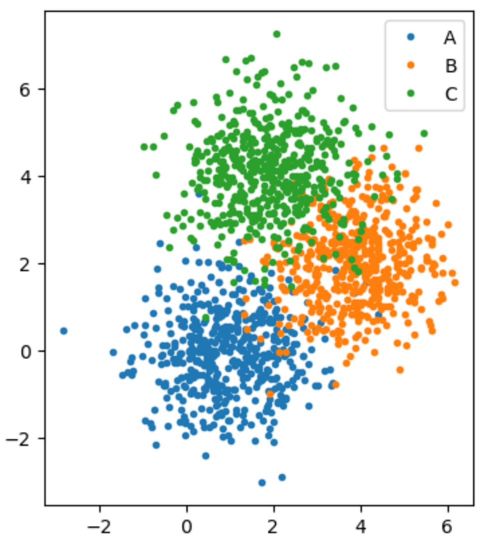
- **重叠更多的时候两个模型的准确率下降，但也高于1/3的随机猜测准确率**
- **几乎不重叠时，两个模型的准确率上升，接近于1**
- **重叠对两者的影响类似**

尤其是第一个图的情况（下图），让我略有震惊，观察图片，点几乎重叠在一起，两个模型都能以60%的准确率分类。






A
B
C
6
4
2
0
−2
−2
0
2
4
6


A
B
C
6
4
2
0
−2
−2
0
2
4
6