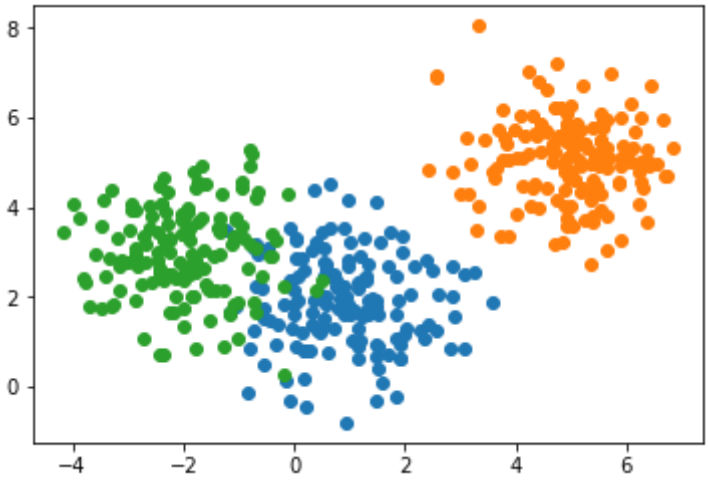
Part I

生成数据：

source.py中getdata()生成数据，三种数据的mean,cov,num分别是数据的平均值，协方差和这种数据的点数。

调用numpy.random.multicariate_normal()生成数据，以x y label的格式输出到dataset.data中，每行一个数据点。并且输出绘制的散点图。

默认数据形如下图：



直接运行即可生成数据集。

Part II

生成模型：

source.py中前半部分为生成模型。cal()函数中从数据x，标签y中计算出三种分布的概率pi，均值mu，协方差sigma，并且返回(pi,mu,sigma)。

因为生成模型的最大似然估计有解析解，可以直接计算出最佳的(pi,mu,sigma)。然后运行testgen()函数，传入数据和学习好的三个分布进行预测（没有区分训练集和测试集）。对于每个数据点，调用predictgen()得到预测标签 。predictgen()中分别使用三个分布进行分类，选出概率最大的作为预测值。

最后输出预测准确率，判断结果的图形，判断错误的数据点。

判别模型：

source.py后半部分为判别模型。因为判别模型下，使用了$y = \sigma(\mathbf{w}\mathbf{x} + w_0)$作为预测值，所以梯度不为0，需要使用梯度下降等方法来学习$\mathbf{w}$. w_0的值。代码中sgd()函数对参数$\mathbf{w}, w_0$进行学习。这里执行的过程大概如下：

- 随机选出batch个数据点
- 计算这些点的loss和梯度，修改参数$\mathbf{w}, w_0$
- 重复上面的过程epoch次

并且在开始调试时因为学习速率alpha设置的太小了，就不断加大学习速率，并且增加epoch。所以在训练过程中加入了alpha每次减少alpha/epoch的设定，但是实际看来和不修改区别不大。因为模型比较简单，最后都能够达到饱和状态。

最后的参数是epoch=2000,batch=10,alpha=1

之后和生成模型类似，调用testdis()对数据进行预测。

两个模型的比较

在默认数据下，生成模型准确率为96.9%，判别模型为94.9%，二者相差不大，因为模型比较简单，并且数据的分布相对分散，只有少数outlier会分布在大量的其他数据中，造成预测错误。

生成模型学习的是对于数据分布的预测，所以除了给出一个数据预测标签外，还可以用来生成数据。但是判别模型只学习了判断数据种类的线性函数，所以只能对数据分类。

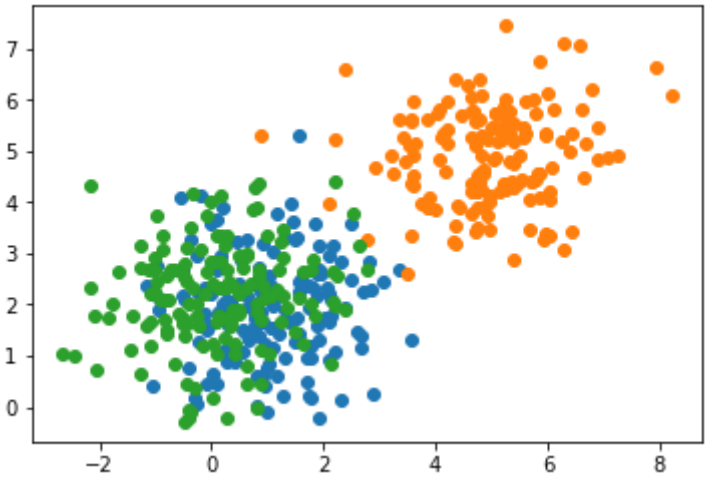
因此，生成模型需要学习更多的参数（13个）来预测数据的分布，而判别模型的参数相对少一些（9个）。因为数据是二维的，所以差距不大，但是当数据的维度变大之后，生成模型的参数数量是$O(n^2)$的。

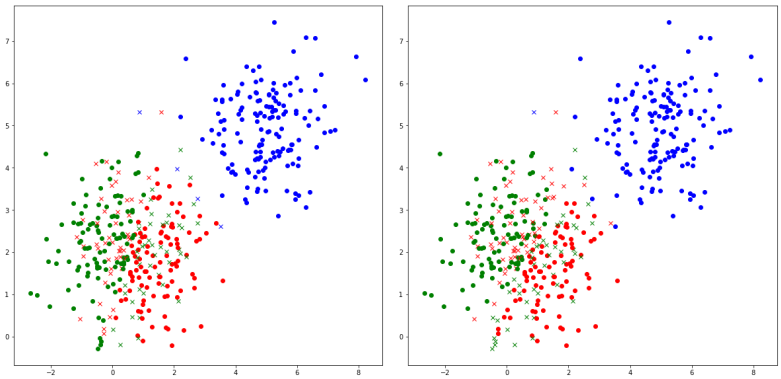
生成模型预设了数据的分布类型，比如高斯分布，所以如果数据的真实分布情况不是高斯分布，或者不是简单的分布时，生成模型的能力就会非常受限。但是判别模型不考虑数据的分布类型，只要数据是可分的，就能够将进行学习。

在这个模型下，生成模型有解析解，可以直接计算出最优的分布。但是判别模型没有解析解，需要使用梯度下降，或者其他迭代方式来学习参数。

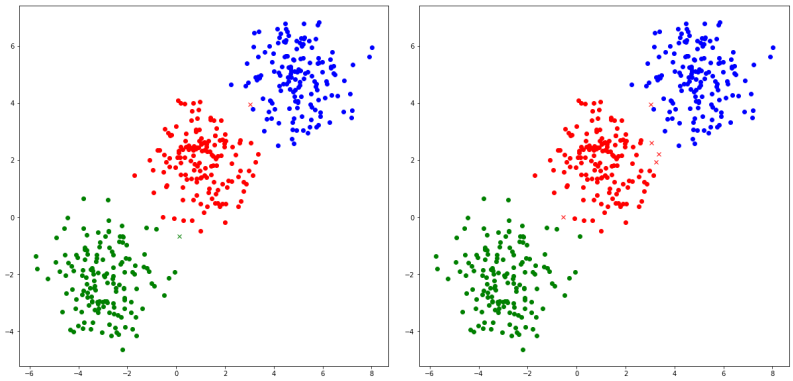
Part III

1.将其中两个标签的数据贴合紧密，导致两个模型的准确率下降到78.6%,76.2%，不能分辨出两种标签，但是可以看出，在两个模型下，重叠部分有一条明显的直线作为边界，以直线为分界预测标签。并且由于输入可供学习的信息不足，所以无法区分这两种类型。图中左侧为生成模型，右侧为判别模型。

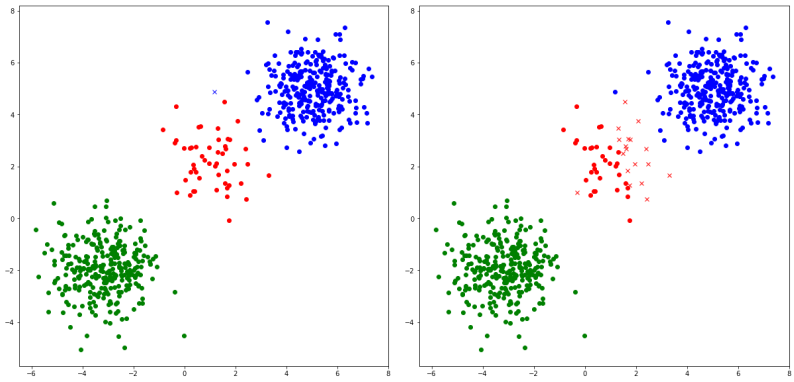




2.相比之下，如果三组数据线性可分，那么两种模型的正确率都会很高，分别是99.5%和98.9%



3.但是如果数据分布不变，把其中一种类型的数据采样数量显著下降（采样数为300，50，300），那么生成模型的表现依然很好，为99.8%。判别模型的表现很差，虽然正确率为97.2%，但是红色的点有三分之一被判断错误（参数为epoch=2000,batch=10,alpha=1）将batch改为100之后表现依然很差，将epoch改为20000后正确率变为99%。推测原因是因为我写的梯度下降并不是正常版本的随机梯度下降，在这样的写法下，如果样本数量很少，会导致这种样本的学习机会非常少，学习次数不够时没有收敛，所以当epoch变大之后有明显变化。



代码运行方式

```
python source.py
```