Report

> this is report of PRML, assignment 1

Table of Contents

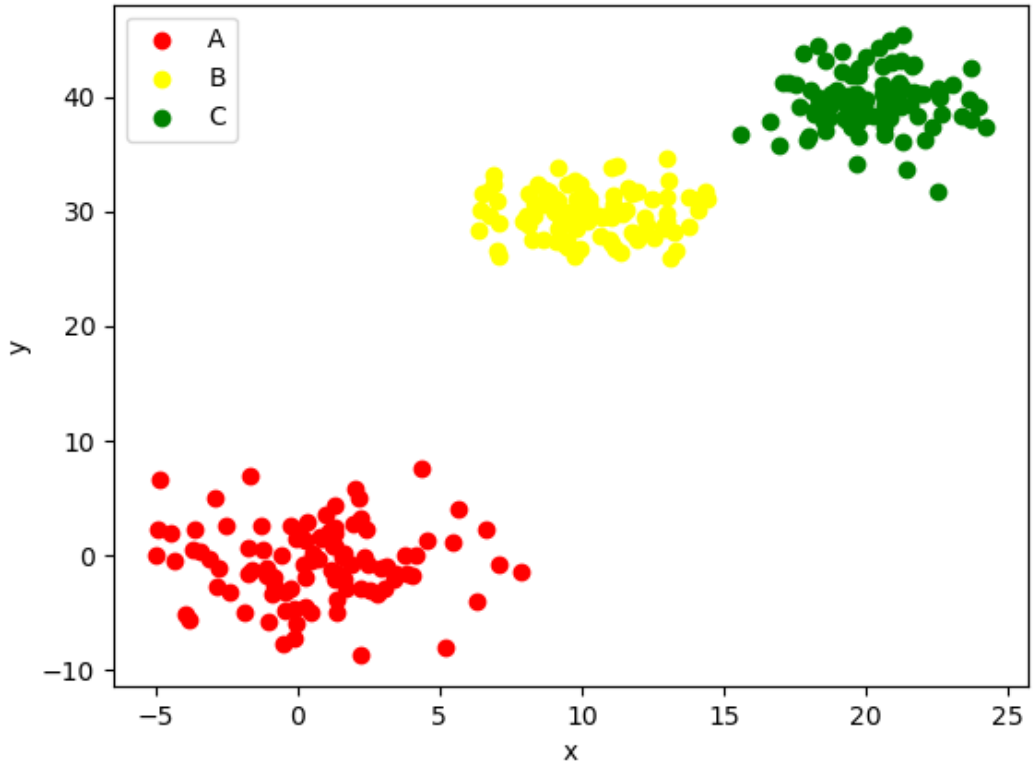
Part I. Generate Gaussian Distribution

> In this part, you are going to design 3 gaussian distribution, each of distribution indicates label A, label B, and label C, respectively. Then construct a dataset sampling from these distributions.

调用`numpy`的方法便可直接生成高斯分布，为了方便画图，这里选择生成二维高斯分布。`source.py`文件中`generate_gaussian_distribution`函数可以直接生成并返回三个高斯分布的数据和标签。

输入分别是三个高斯分布的均值向量、协方差矩阵和标签名称、点的数目；输出是点的坐标值和标签的`list`。

生成效果如下：



Part II.

> In this part, you are required to construct 2 linear classification models: a generative model and a discriminative model. Meanwhile, you are required to compare their differences.

2.1 生成模型

生成模型的原理是利用先验概率和最大似然估计得到的参数计算不同y值下的后验概率，选择使得后验概率最大的y值。

$$\arg\max_y p(y|x) = \arg\max_y \frac{p(x|y)p(y)}{p(x)}$$

在已经知道数据集服从高斯分布的情况下，可以"假设"数据集服从高斯分布，然后利用最大似然估计得到高斯分布的参数。

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-u)\}$$
$$L(\mu,\Sigma) = \prod_{i=1}^{i=N} p(x_i)$$
$$\mu^*,\Sigma^* = \underset{\mu,\Sigma}{arg\,max} L(\mu,\Sigma)$$
$$\mu^* = \frac{1}{N}\sum_{i=1}^{i=N} x_i$$
$$\Sigma^* = \frac{1}{N}\sum_{i=1}^{i=N}(x_i-\mu^*)(x_i-\mu^*)^T$$

得到高斯分布的估计参数后，再分别计算各个点属于哪个类别的概率，取得最大值的那个认为是该点的分类。

$$p(y=C_1|x)=\frac{p(x|y=C_1)p(C_1)}{p(x|y=C_1)p(C_1)+p(x|y=C_2)p(C_2)+p(x|y=C_3)p(C_3)}$$

$$p(y=C_2|x)=\frac{p(x|y=C_2)p(C_2)}{p(x|y=C_1)p(C_1)+p(x|y=C_2)p(C_2)+p(x|y=C_3)p(C_3)}$$

$$p(y=C_3|x)=\frac{p(x|y=C_3)p(C_3)}{p(x|y=C_1)p(C_1)+p(x|y=C_2)p(C_2)+p(x|y=C_3)p(C_3)}$$

$$\text{其中，}p(x|y=C_i)=f_{\mu_i^*,\Sigma_i^*}(x)\text{、}p(y=C_i)=\text{每个高斯分布所占据的比例}$$

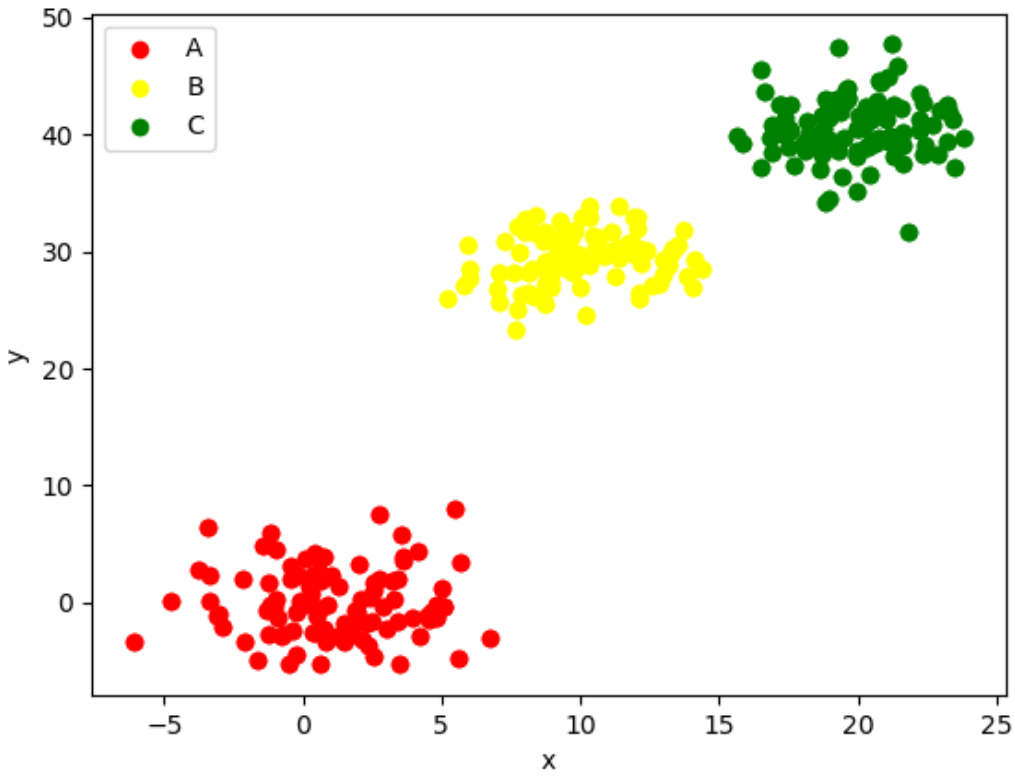
这样计算以后，得到的生成模型准确率仍然不高。此时，为了提高准确率，可以改进模型。将高斯分布的协方差矩阵共享，即：

$$\Sigma=p(y=C_1)*\Sigma_1^*+p(y=C_2)*\Sigma_2^*+p(y=C_3)*\Sigma_3^*$$

将此协方差矩阵作为三个高斯分布的共同协方差矩阵。

`source.py`文件中`GenerativeModel`类中已经实现了相应的代码。

最后的预测结果如下：




```
$ python source.py
----------Generative Model----------
Total:300, predict right:300, accuracy:1.000000
```

显然，对于初始数据不同类别之间比较分散的数据集来说，生成模型的准确度是非常高的。这一点将在part III详细讨论。

2.2 判别模型

对于二分类的判别模型而言，通过不断地调整权重参数和偏置参数，使得损失函数的值达到最小。然后通过`sigmoid`函数对输入x进行预测，得到的y值如果小于0.5，则认为不属于该类，如果大于0.5，则认为属于该类。而面临多分类（3个类别），我们需要做的是针对三个类别的权重参数和偏置参数，分别计算出每个点属于三个类别的概率，取其中最大的一个概率值的类，认为该点属于该类别。

在考虑损失函数的选取的时候，不能使用常用的MSE（均方误差）来作为损失函数，因为MSE一般是非凸函数，容易在使用梯度下降算法的时候得到局部最优解。因此要选择二阶导大于等于0的凸函数，这里选择交叉熵作为损失函数。

$$\hat{y} = \sigma(wx+b) = \frac{1}{1+e^{-(wx+b)}} \in (0,1)$$
$$\text{损失函数} L(w,b) = -ylog(\hat{y}) + (1-y)log(1-\hat{y})$$

确定损失函数以后，采用梯度下降的方法找到修改权重参数和偏置参数的值。

$$L(\hat{y},y) = -ylog(\hat{y}) + (1-y)log(1-\hat{y})$$
$$\frac{\partial L(w,b)}{\partial w} = \frac{\partial L(w,b)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial w} = -(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}})\hat{y}(1-\hat{y})x = (y-\hat{y})x$$
$$\frac{\partial L(w,b)}{\partial b} = \frac{\partial L(w,b)}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial b} = -(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}})\hat{y}(1-\hat{y}) = (y-\hat{y})$$

代码实现在`source.py`文件的`DiscriminativeModel`部分，其中，为了使得训练效果更客观，我将随机打乱后的数据集按照（默认）80%和20%的比例划分为训练集和测试集，训练结束以后使用测试集对模型进行测试。

训练函数的输入是batch大小和训练的epoch，以及学习率。

实现过程大致如下：

（1）每个epoch开始，随机打乱训练集；

（2）选取batch个训练点，进行一次iter训练

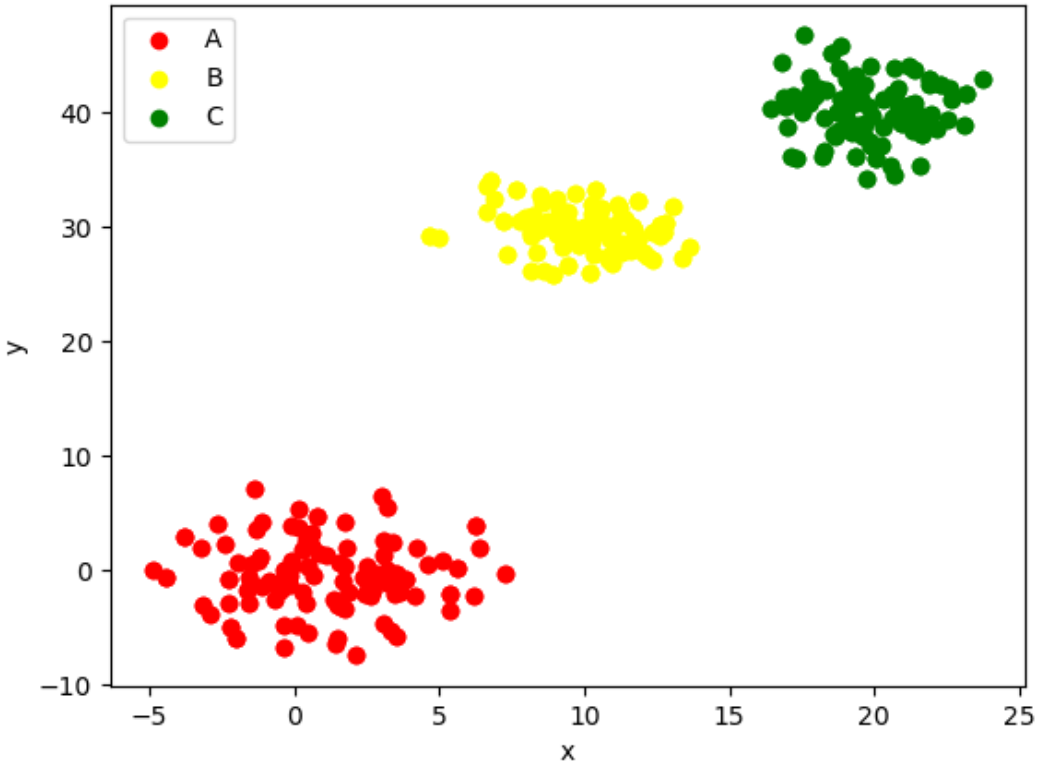
（3）不断计算损失函数和梯度，修改权重参数和偏置参数。

为了提高训练的准确度，本可以在每个epoch训练结束后对测试集进行预测，选取效果最好的w和b。

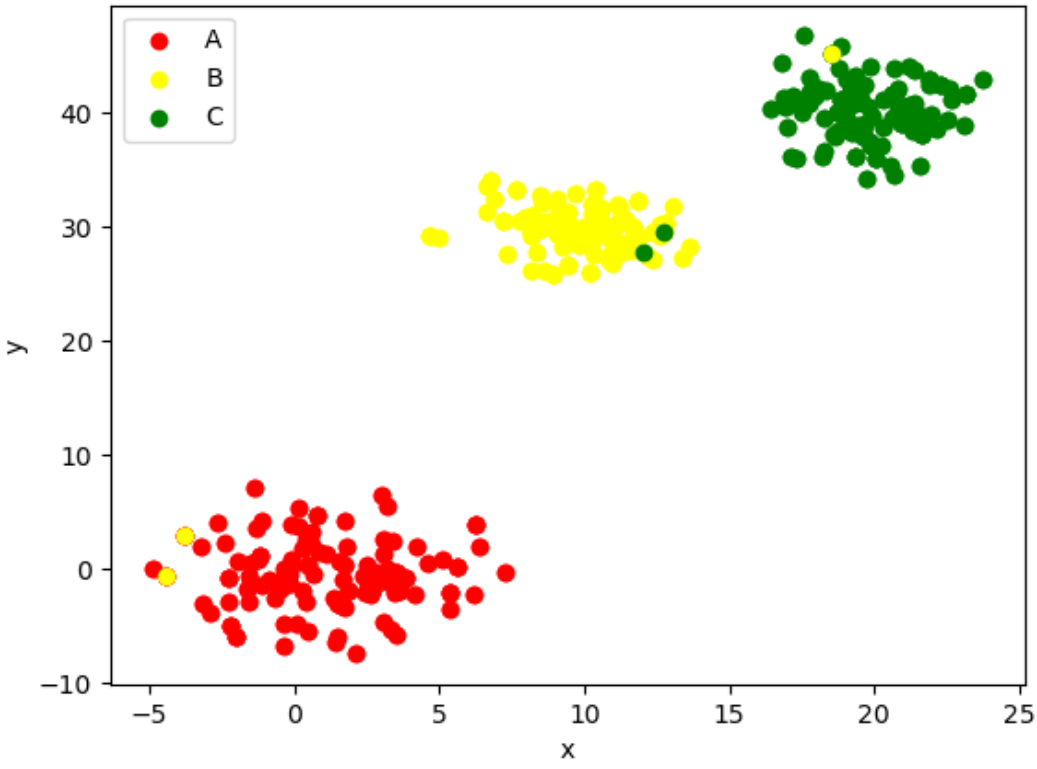
最后训练结果如下：（batch=20, epoch=250, learning_rate=1）

```
Epoch: 249
Train:  Total:240, right:206, accuracy:0.858333
Test:   Total:60, right:55, accuracy:0.916667
```

测试集的真实数据分类：



对测试集的预测结果分类：



3. 模型对比

对于线性生成模型来说，已经知道数据集的真实标签，并且根据这个标签，将不同类别的数据分别用于各自分布参数的估计（最大似然估计）；经过最大似然估计，属于同一个高斯分布的数据已经足以将本高斯分布的参数（均值和协方差）估计出来，由此得到该高斯分布的似然概率密度，然后根据贝叶斯公式，计算每个点属于各个高斯分布的概率，选择概率最大的所对应的高斯分布作为对该点的预测。

而对于线性判别模型来说，并不在意你是高斯分布还是什么分布。判别模型的任务是直接根据x和初始权重，学习概率$p(\hat{y}|x)$，并且利用梯度下降方法找到最快修正权重参数和偏置参数的方向。通过不断的fit，直到找到模型最合适的参数。在寻找的过程中，需要注意可能陷入局部最优值。因此需要选取凸函数交叉熵作为损失函数。

在所需参数方面，生成模型需要更多的参数，而判别模型的参数更少，并且参数的自我调整来源于每个batch的训练，并不是直接求得。

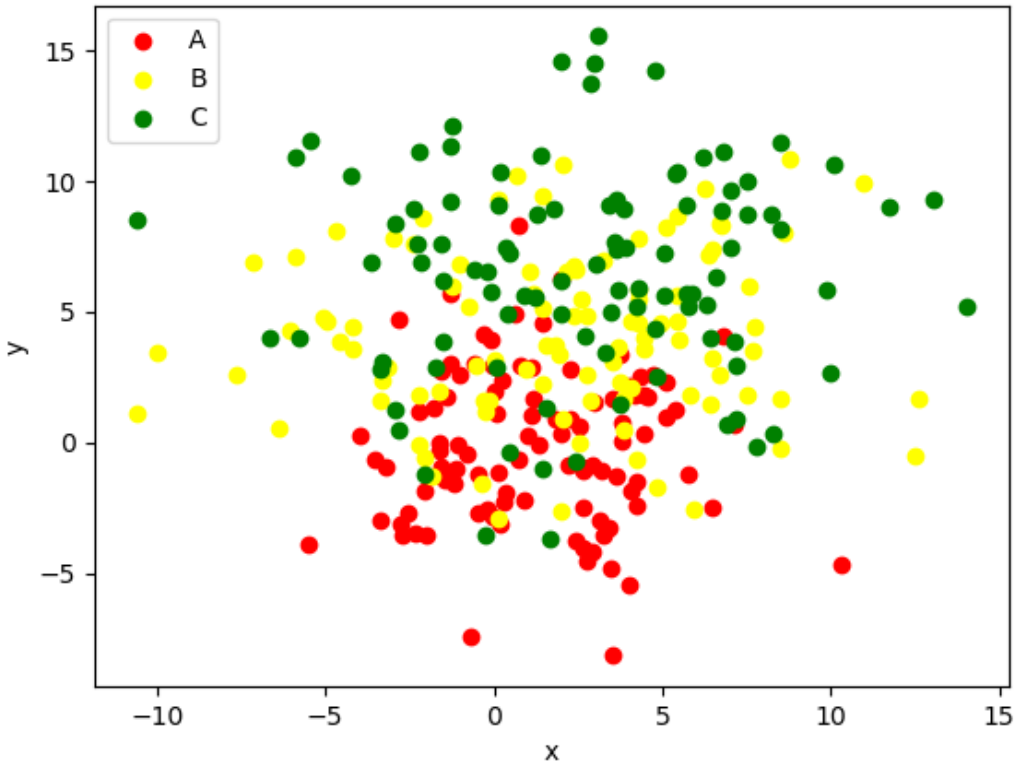
此外，生成模型可以对数据的分布进行估计并且给出相应的标签，而判别模型只能利用线性函数对数据进行分类。生成模型可以用来直接生成数据，而判别模型只能分类。但是如果数据集并不是服从严格的高斯分布或者其他分布，生成模型就没办法给出合适的分布预测。而判别模型则并不关心数据的真实分布如何，只利用梯度下降等方法将数据进行分类。

Part III.

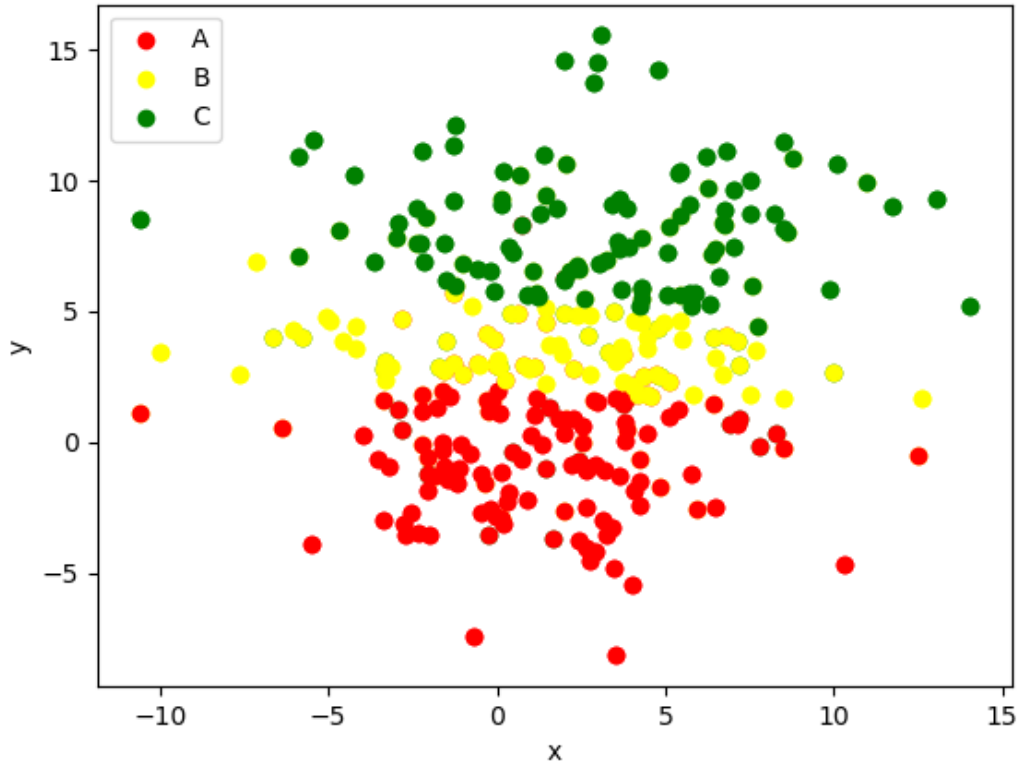
> In this part, you can reorganize the scale of your dataset or the adjust the overlap between different gaussian distributions.

实验证明，初始数据的离散程度和彼此之间的分散程度会影响两个模型的准确率。比如，当数据彼此之间相隔比较近或者协方差比较大时，就会出现如下情况：

真实分布：



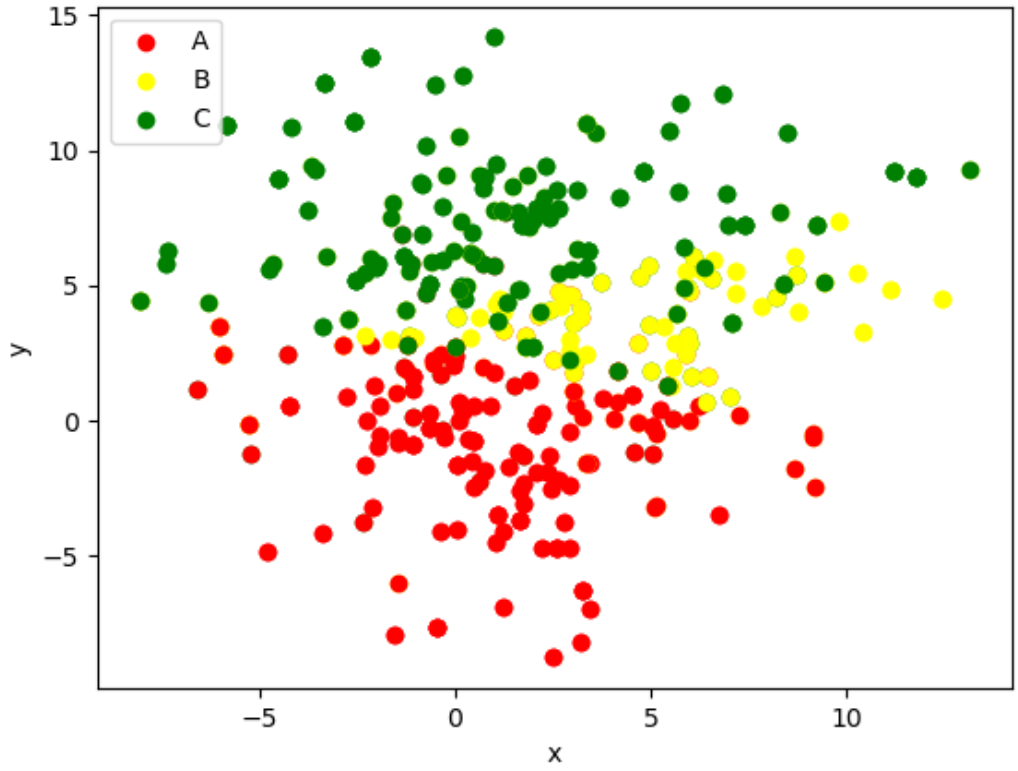
生成模型预测：



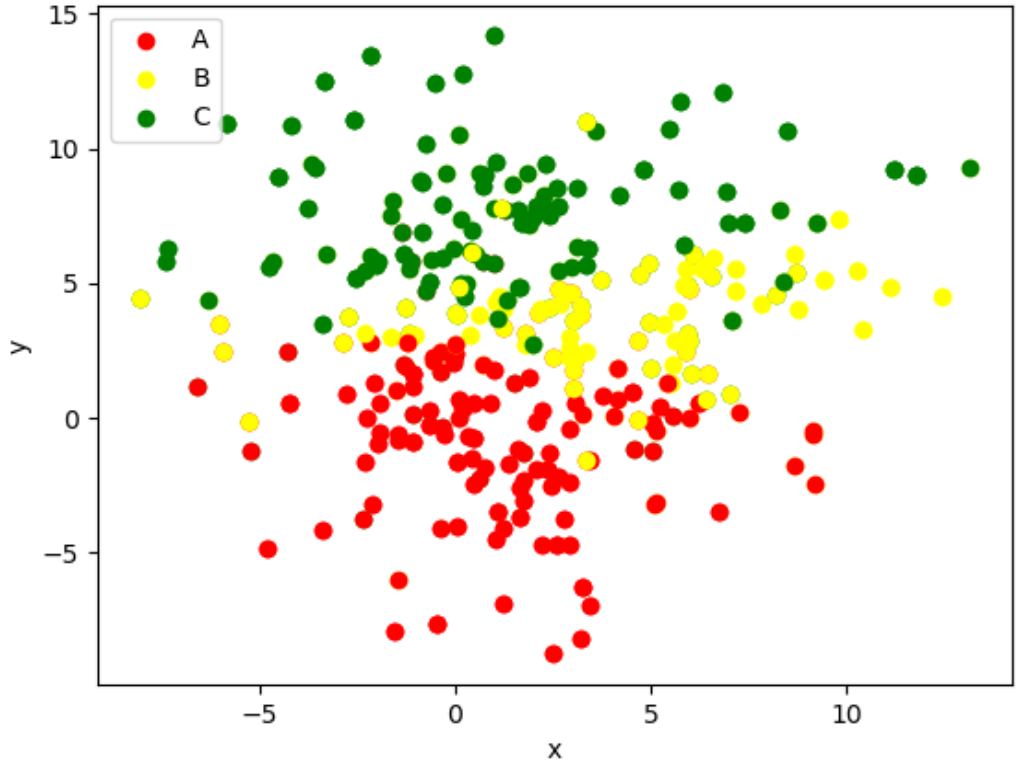

```
$ python source.py
----------Generative Model----------
Total:300, predict right:183, accuracy:0.610000
```

分布模型预测： （batch=20, epoch=250, learning_rate=1）

```
Epoch: 249
Train:  Total:240, right:122, accuracy:0.508333
Test:   Total:60, right:35, accuracy:0.583333
```



真实测试集：

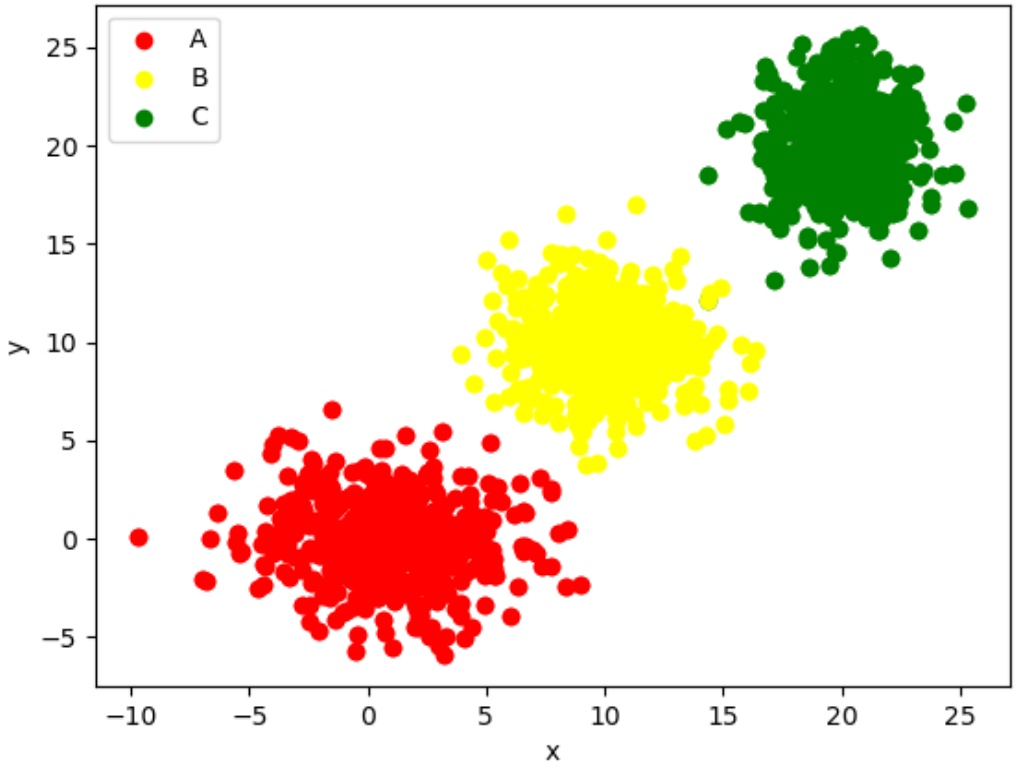


注：上述极其“惨烈”的预测结果是故意为之，只是想说明两个线性模型在面对相似程度较高的三个高斯分布时的无能为力。

当数据分布彼此之间相隔较远时，两个模型的预测准确率均非常高。

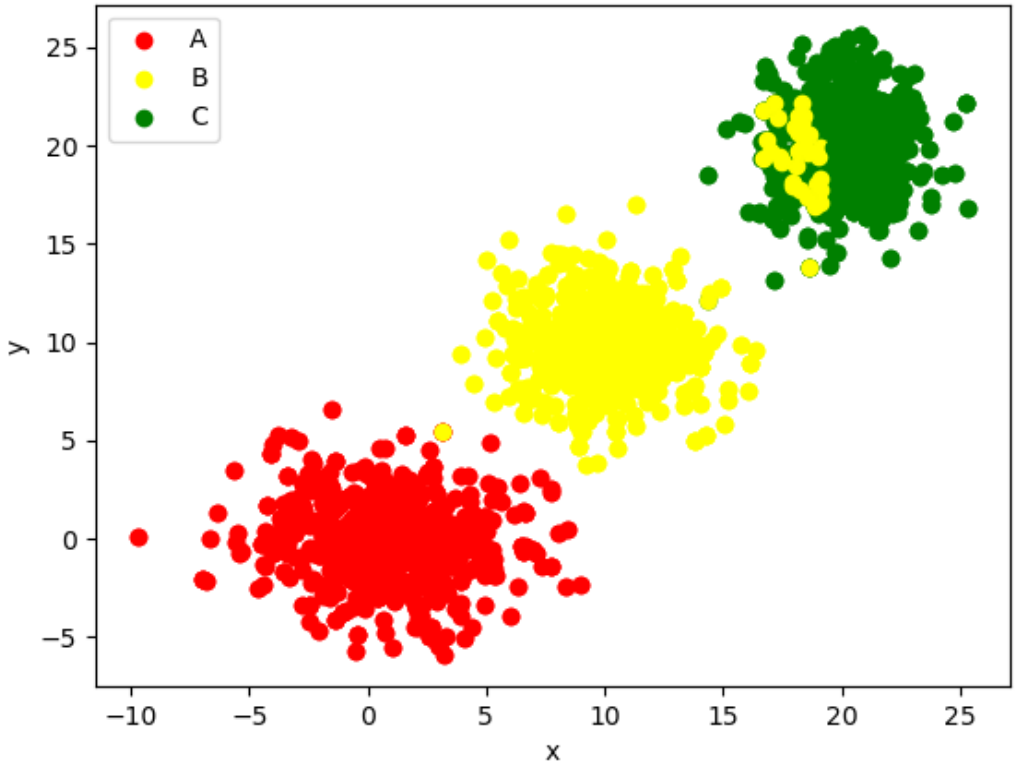
```
$ python source.py
----------Generative Model----------
Total:1500, predict right:1499, accuracy:0.999333
```

生成模型预测：



判别模型预测：

```
Epoch: 249
Train:  Total:1200, right:978, accuracy:0.815000
Test:   Total:300, right:262, accuracy:0.873333
```

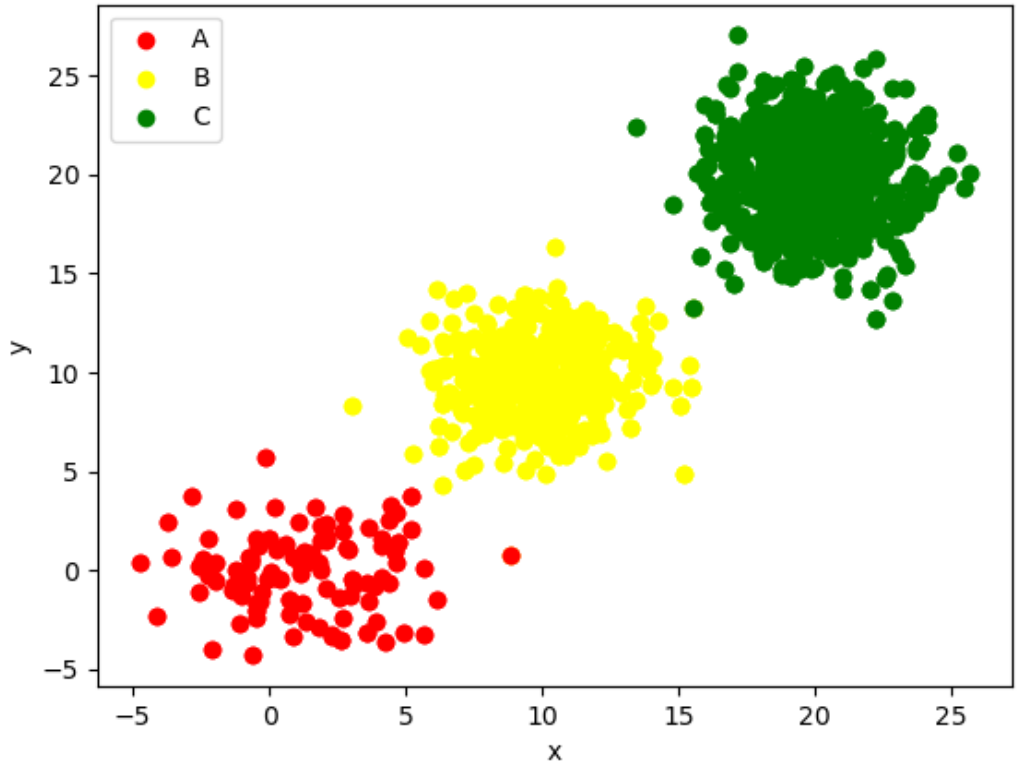


数据集规模大小对模型预测结果的影响小于不同分布之间相似性（均值、协方差）对模型的影响。但是，如果调整三个分布之间的采样率，会出现如下结果：

三个分布采样率调整为`100，400，800`

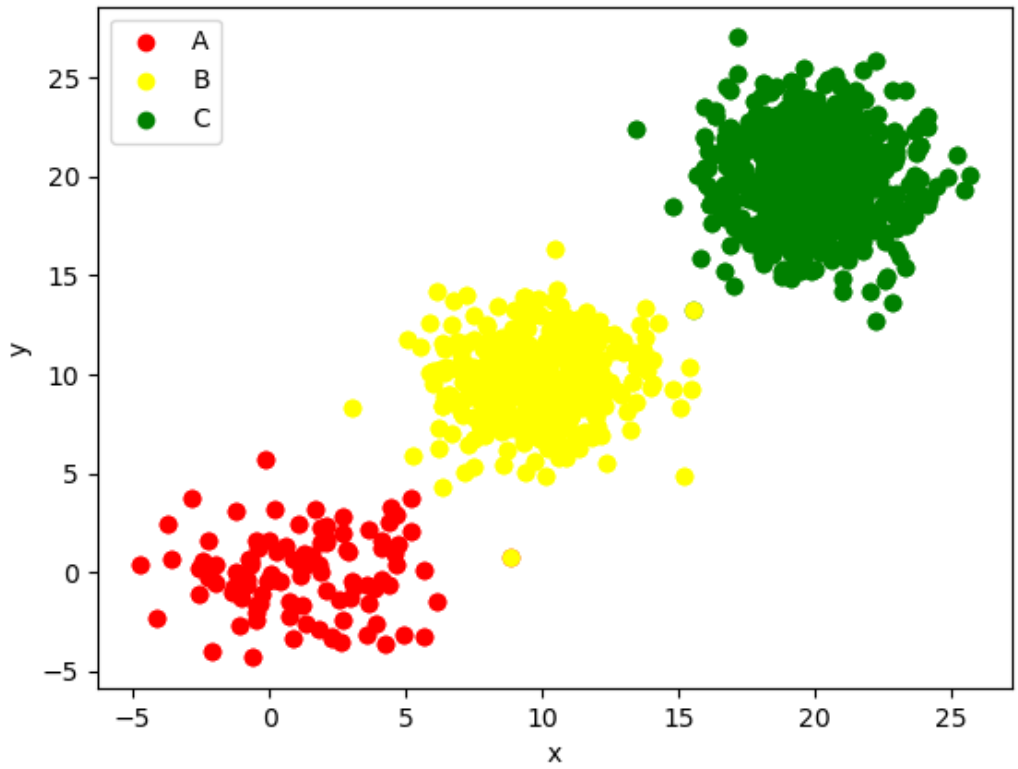
判别模型预测率大幅提高：

```
Epoch: 249
Train:  Total:1040, right:939, accuracy:0.902885
Test:   Total:260, right:251, accuracy:0.965385
```



生成模型预测准确率依旧很高：

```
$ python source.py
----------Generative Model----------
Total:1300, predict right:1298, accuracy:0.998462
```



Run Code

3.1 环境依赖

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as st
import random as rd
```

3.2 运行代码

```
$ python source.py
```