Assignment1

> To run the project
>
> Anaconda Prompt run
>
> >python source.py

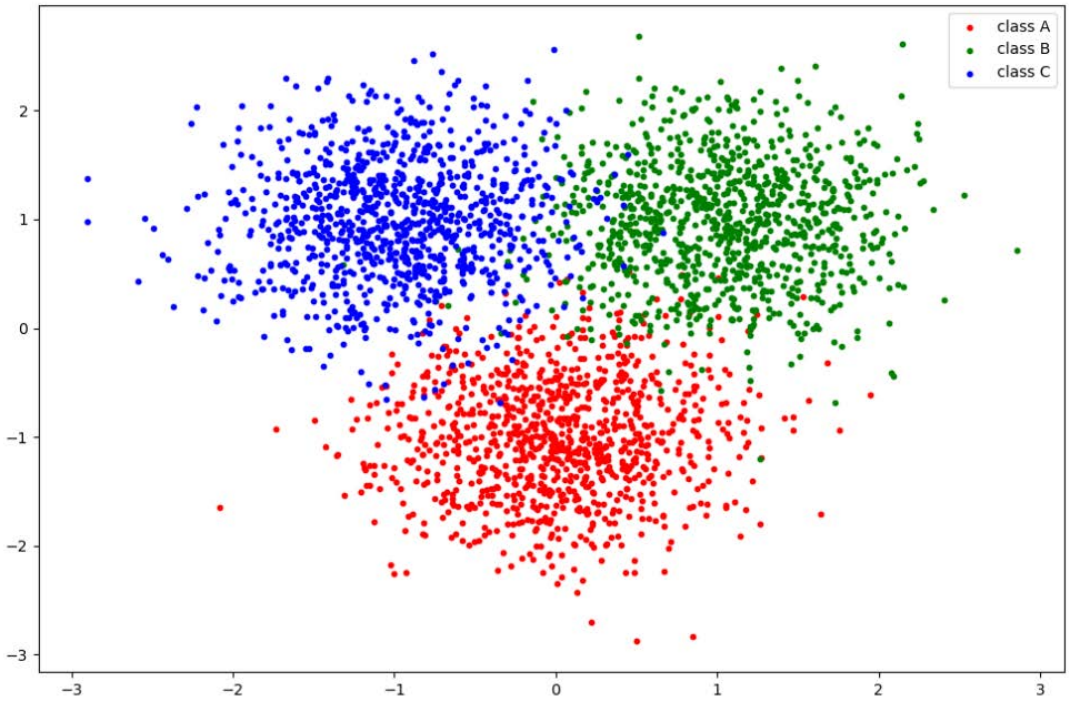
```
(base) D:\Projects\PRML-Spring20-FDU\assignment-1\18307100008>python source.py
Start training Linear Generative Model
Train completed.
Linear Generative Model accuracy: 0.9883333333333333

Start training Linear Discriminative Model
Epoch 100 ,Loss: 0.14123815019429334
Epoch 200 ,Loss: 0.08742848675428105
Epoch 300 ,Loss: 0.08340663967579365
Epoch 400 ,Loss: 0.07120531272539715
Epoch 500 ,Loss: 0.05791362697418894
Epoch 600 ,Loss: 0.08174349891508226
Epoch 700 ,Loss: 0.06066599404460264
Epoch 800 ,Loss: 0.07398129075589727
Epoch 900 ,Loss: 0.06438874461510355
Epoch 1000 ,Loss: 0.07732737900651754
Train completed.
Linear Discriminative Model accuracy: 0.9883333333333333
```

Part1

> **Dataset** is available via Datasets.

Python function `createDataset()` calls `numpy.random.multicariate_normal()` to build 3 sets of points with dimension 2 (by default) drawn from gaussian distribution of separately specified covariance and a common covariance matrix. The scale of datasets is controlled by the default parameter `scale` , which is 1000 each set by default.

Data set used in part2 is data.data, visualized below.



For each row in the dataset, points are displayed in form (coordinate_x, coordinate_y, label). The labels are presented in integers in the datasets, where 0 stands for class A, 1 for B and 2 for C.

Part2

linear generative model

The linear generative model is implemented in python class `LinearGenerativeModel`. Given data X of size m and labels y of n classes, we assume the prior probability

$$p(y) = \prod_{i=1}^{n} \phi_i^{1\{y=i\}}$$
$$p(x|y=i) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T \Sigma^{-1}(x-\mu_i)\right) \; for \; i = 1, 2, \ldots, n$$

where μ_i is the mean of the i^{th} gaussian distribution and Σ is the common covariance.

To maximize the log likelihood

$$\begin{aligned}\ell(\phi, \mu, \Sigma) &= \log \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}; \phi, \mu, \Sigma\right) \\ &= \log \prod_{i=1}^{m} p\left(x^{(i)}|y^{(i)}; \mu, \Sigma\right) p\left(y^{(i)}; \phi\right)\end{aligned}$$

In method `train`, parameters ϕ, μ, Σ are calculated explicitly using gradients

$$\begin{aligned}\phi_t &= \frac{1}{m}\sum_{i=1}^{m} 1\left\{y^{(i)} = t\right\} \\ \mu_t &= \frac{\sum_{i=1}^{m} 1\left\{y^{(i)} = t\right\} x^{(i)}}{\sum_{i=1}^{m} 1\left\{y^{(i)} = t\right\}} \\ \Sigma &= \frac{1}{m}\sum_{i=1}^{m}\left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^T\end{aligned}$$

Method `predict` outputs label i which maximize the conditional probability $p(x|y=i)$.

linear discriminative model

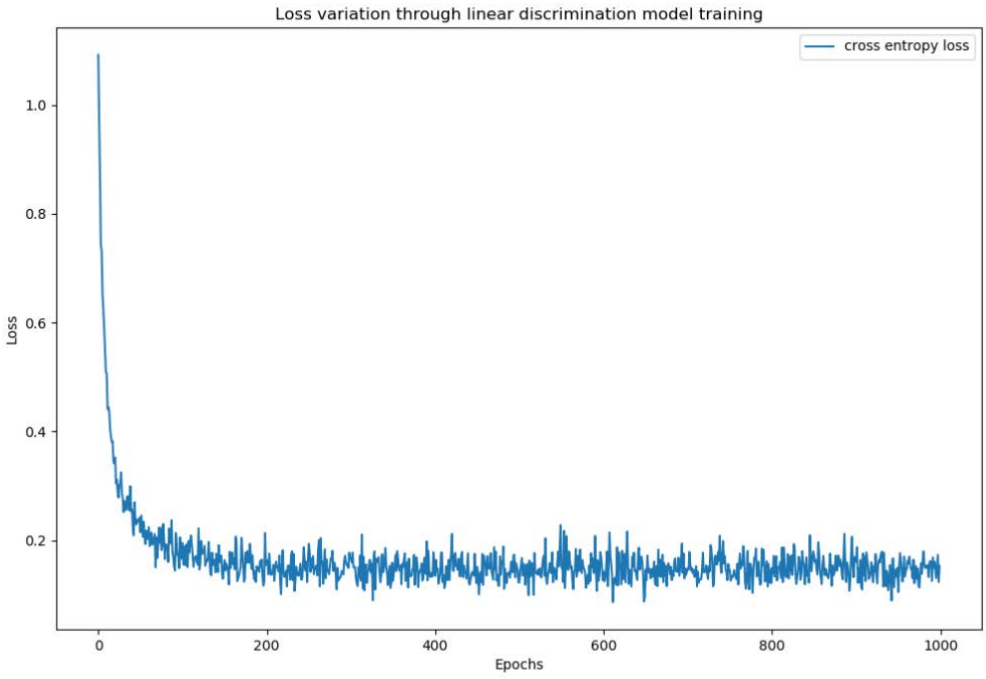
The linear discriminative model is implemented in python class `LinearDiscriminativeModel`, where we uses $\mathrm{softmax}(W\boldsymbol{x} + b)$ for probability of x in each class.

For convenience, we append 1 after each x and neglect b.

The loss of each data point x is defined $l = -y^T \log \mathrm{softmax}(Wx)$, where y is the one-hot vector of the true label of x. In method `train` , parameter W is optimized by stochastic gradient descent while the gradient for data point x is $\frac{\partial l}{\partial W} = (\mathrm{softmax}(Wx) - y)x^T$.

Method `predict` outputs label i which maximize the i^{th} probability in $\mathrm{softmax}(Wx)$.

Loss fluctuates while epochs increases, this is mainly caused by the randomicity of the mini-batch, as learning rate reduction can't ease the fluctuation.



model comparison

- The final accuracy of the linear generative model is 96.67%.
- The final accuracy of the linear discriminative model is 96.5%.

The two models varies in **modeling** and **training step**.

Linear generative model fits the parameters of **joint probability** $p\left(x^{(i)}, y^{(i)}; \theta\right)$ while discriminative model fits the parameters of **conditional probability** $p\left(y^{(i)} | x^{(i)}; \theta\right)$. To solve the parameter estimation, we can use **maximum likelihood estimation**: in generative model, we can get analytical solution by calculation derivatives; in discriminative model, we derives the SoftMax cross entropy loss and can use iterative methods like SGD or Newton's method to solve.

For classification of data X with size m and dimension d into C classes, discriminative models requires $C \cdot (d+1)$ parameter (1 for bias) while generative models requires $C + d \cdot C + d^2$ parameter (C for multinomial distribution $p(y=i)$, $d \cdot C$ for means of gaussian distribution $p(x|y)$ and d^2 for covariance matrix).

Using **prior knowledge** and solves for the joint distribution, generative models uses more parameters and is more capable while handling **latent variables**, shows the inner relationship between data. Discriminative models, however, simply solves for the conditional distribution, simplifies the structure of data and determines a **decision boundary**. Hence, discriminative models requires less parameters.

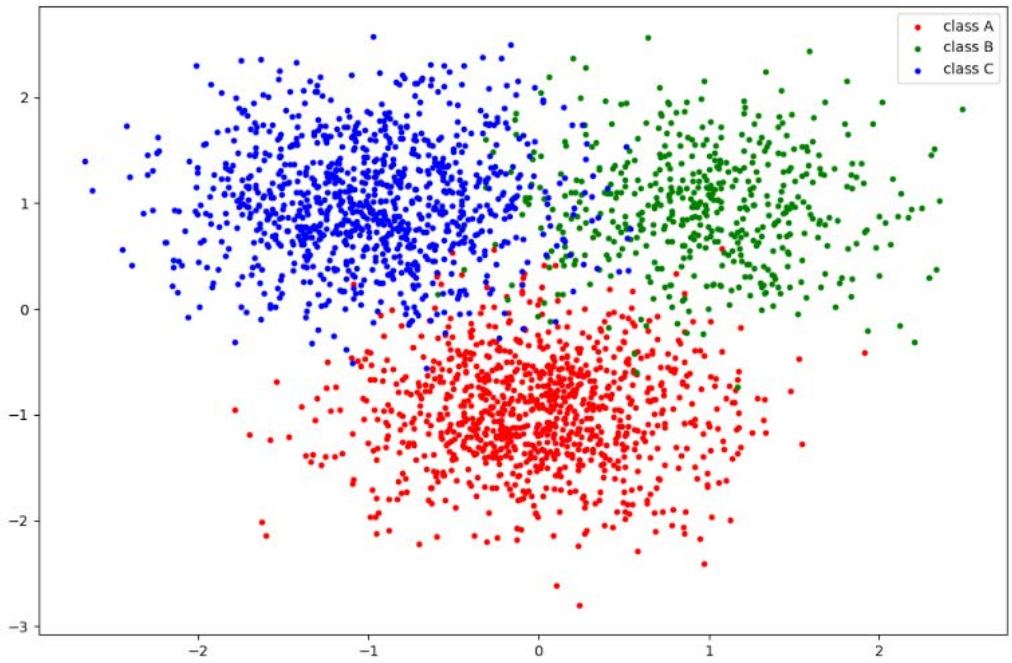
Part 3

scale

When we increases the scale of all 3 sets of data points, the training epochs of linear discriminative model should increase (when the batch size remains), and therefore, the time consumption of discriminative model increases more comparing to the generative model.

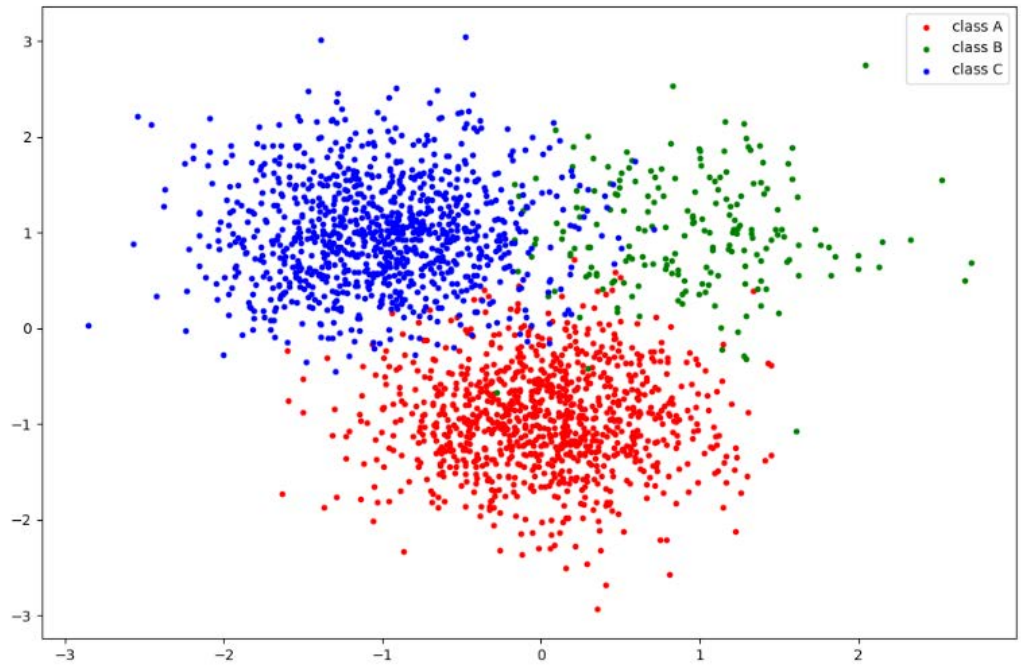
When the scale of one set of points is far less than the other two sets, generative models is affected more than the discriminative model (generative models need **more** data points to fit the precise distribution of data):

1. scale: (1000, 500, 1000)



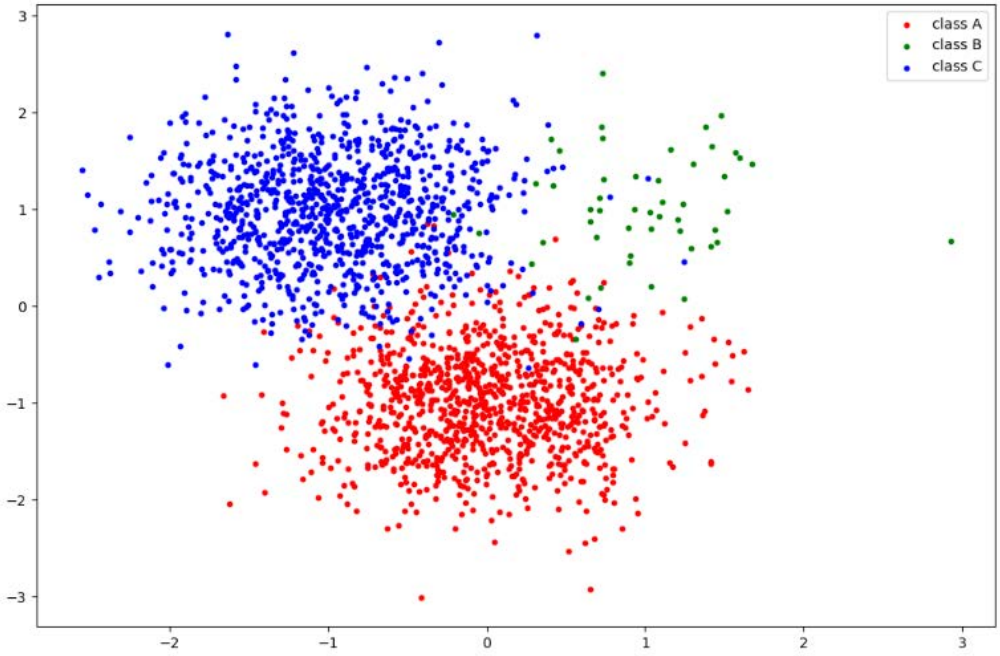
- The final accuracy of the linear generative model is 94.6%.
- The final accuracy of the linear discriminative model is 95.2%.

2. scale: (1000, 200, 1000)



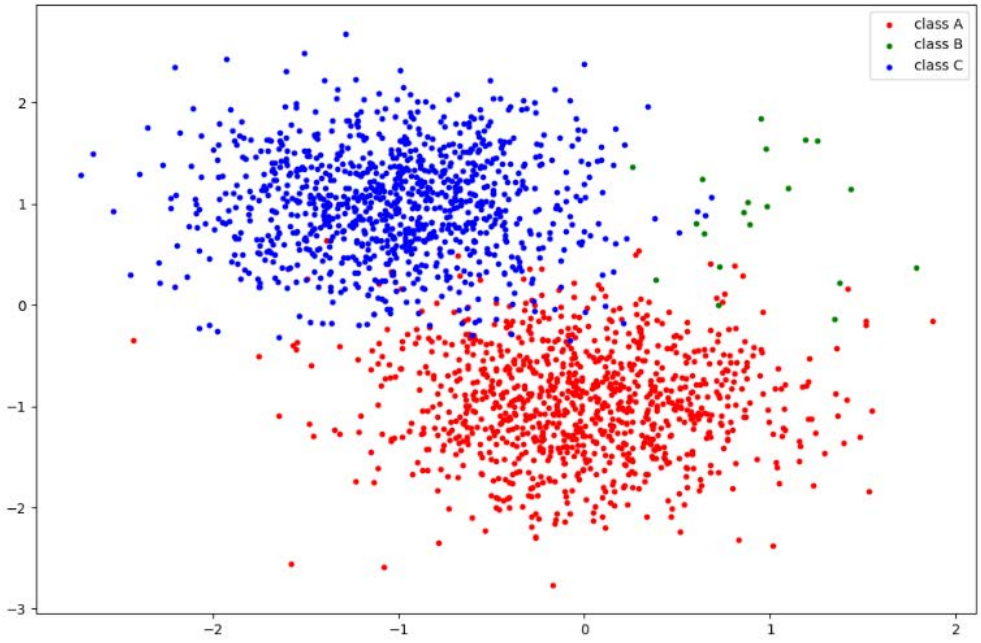
- The final accuracy of the linear generative model is 94.55%.
- The final accuracy of the linear discriminative model is 95.68%.

3. scale: (1000, 100, 1000)



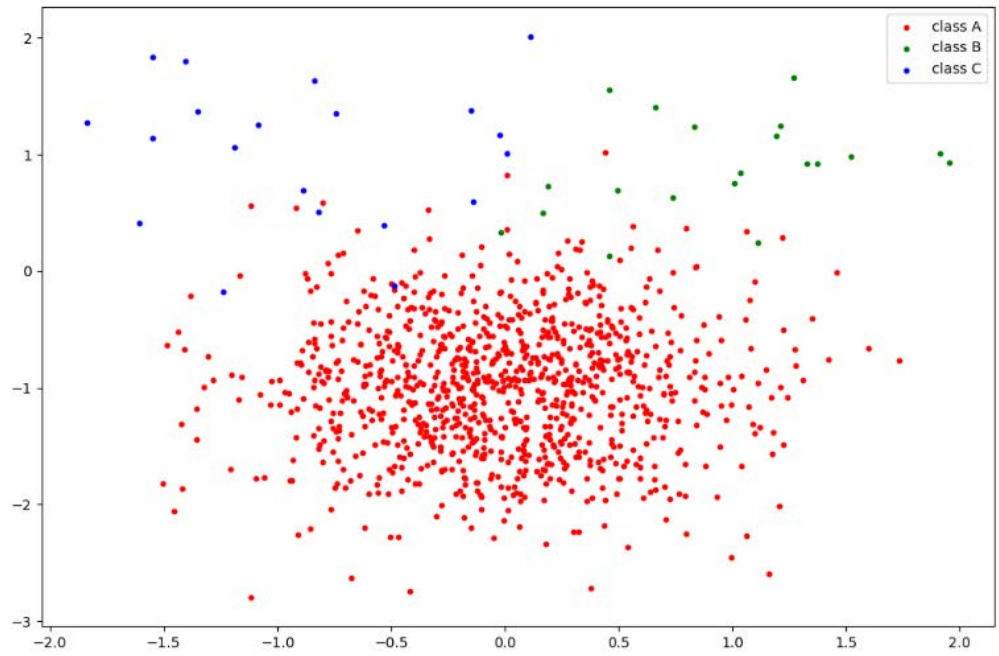
- The final accuracy of the linear generative model is 95.85%.
- The final accuracy of the linear discriminative model is 96.58%.

4. scale: (1000, 20, 1000)



- The final accuracy of the linear generative model is 96.03%.
- The final accuracy of the linear discriminative model is 97.78%.

5. scale: (1000, 20, 20)

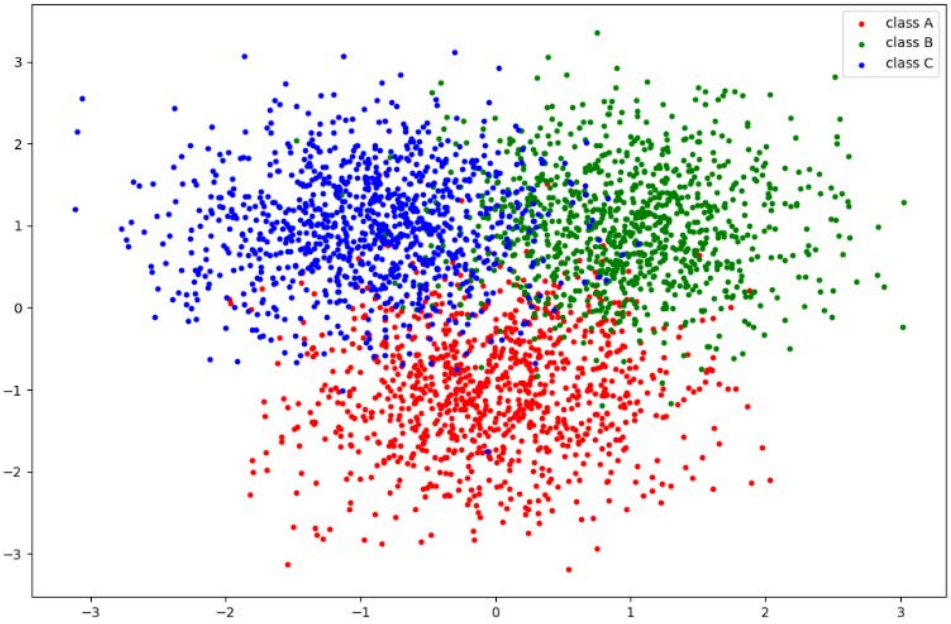


- The final accuracy of the linear generative model is 96.15%.
- The final accuracy of the linear discriminative model is 99.04%.

overlap

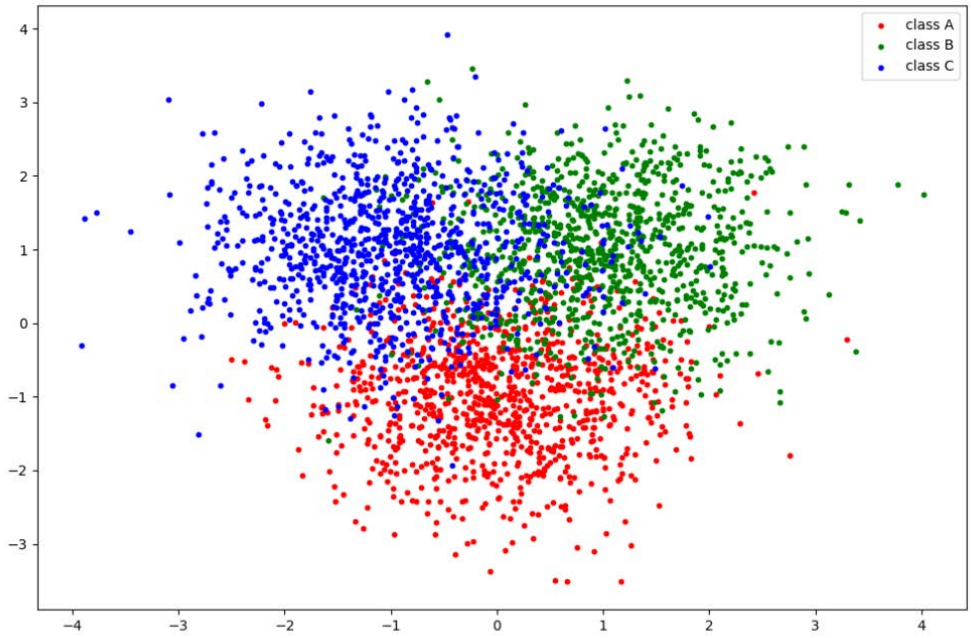
In this part, we adjust the covariance matrix $k \cdot I = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix}$.

1. $k = 0.5$:



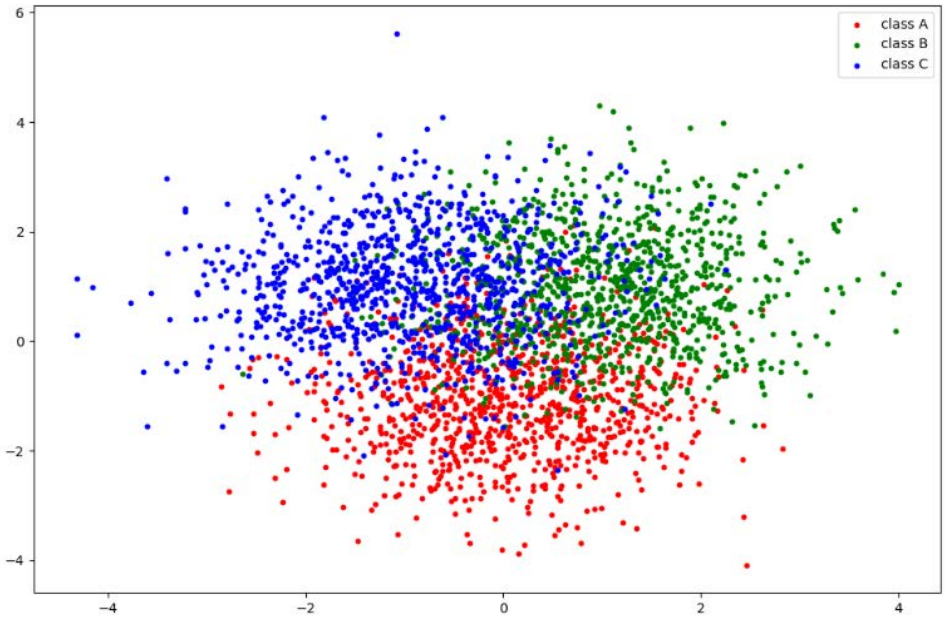
- The final accuracy of the linear generative model is 90.67%.
- The final accuracy of the linear discriminative model is 90.67%.

2. $k = 0.7$:



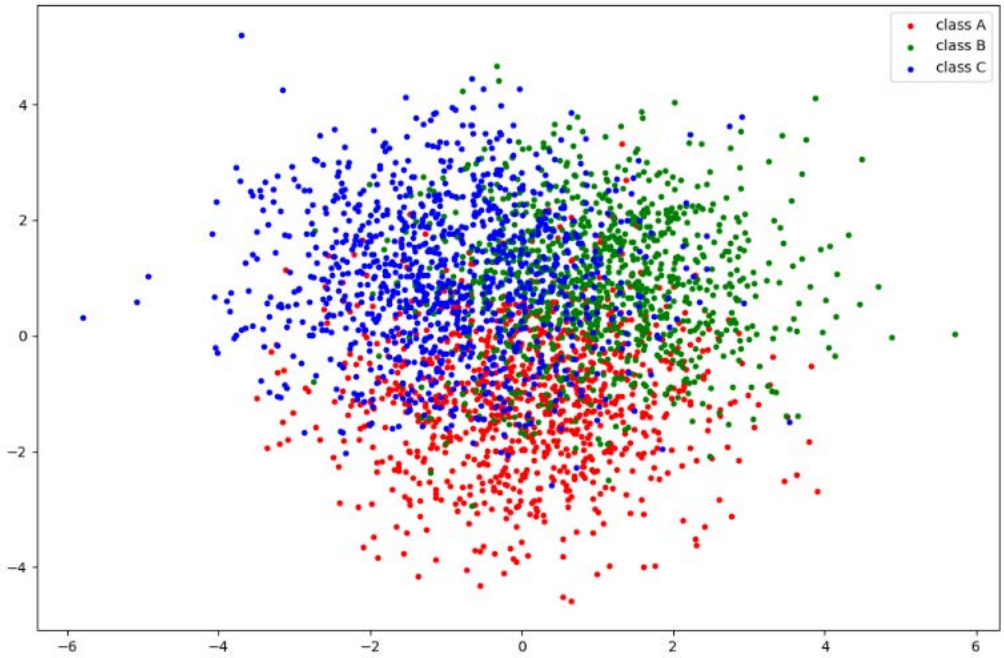
- The final accuracy of the linear generative model is 83.5%.
- The final accuracy of the linear discriminative model is 83.5%.

3. $k = 1$:



- The final accuracy of the linear generative model is 77.17%.
- The final accuracy of the linear discriminative model is 76.67%.

4. $k = 1.5$:



- The final accuracy of the linear generative model is 72%.
- The final accuracy of the linear discriminative model is 71.83%.

We can see that the accuracy of linear generative model and discriminative model is still close as overlap changes.