PRML Assignment1

Instruction for Source Code

`source.py` has defaulted parameters, the easiest way to configure the code is : `python source.py`.

Then dataset, plot of generated data and accuracy of

`source.py` also can be configuerd with params

- **n**[integer]: size of the datasize, then the datasize will be separated into train data, test data and valid data with ratio of 0.6: 0.2: 0.2
- **iter**[integer]: number of interations in training stage of discriminative model;
- **lr**[float]: learning rate in training stage of discriminative model
- **mean**[str]: if you want t`o assign A with 3-d mean (v_A^0, v_A^1, v_A^2), B with 3-d mean (v_B^0, v_B^1, v_B^2), C with 3-d mean (v_C^0, v_C^1, v_C^2), then the input shoud be ' $v_A^0, v_A^1, v_A^2, v_B^0, v_B^1, v_B^2, v_C^0, v_C^1, v_C^2$ '
- **cov**[str]:f you want to assign A with 3x3 cov $[[a_{00}, a_{01}, a_{02}], [a_{10}, a_{11}, a_{12}], [a_{20}, a_{21}, a_{22}]]$, B with 3x3 cov $[[b_{00}, b_{01}, b_{02}], [b_{10}, b_{11}, b_{12}], [b_{20}, b_{21}, b_{22}]]$, C with 3x3 cov $[][a_{00}, a_{01}, a_{02}], [c_{10}, c_{11}, c_{12}], [c_{20}, c_{21}, c_{22}]]$, then the input should bea_{00}, a_{01}, a_{02},a_{10},a_{11},a_{12}, a_{20}, a_{21}, a_{22}, b_{00}, b_{01}, b_{02}, b_{10}, b_{11},b_{12}, b_{20}, b_{21}, b_{22}, c_{00}, c_{01}, c_{02}, c_{10}, c_{11}, c_{12}, c_{20}, c_{21}, c_{22}'

  ```
  python source.py --n=1000 --iter=1000 --lr=0.1 --mean='-1, -1, -0, 1, 1, 3, 2, 4, 2' --cov='1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 3, 2, 0, 0, 0, 1, 0, 0, 0, 1'
  ```

Dataset

- I constructed 3-dimensional data in this assignment. The data is stored in format `x, y, z, label`;
- Label $[A, B, C]$ is transformed to $[0, 1, 2]$ accordingly. To be noticed, in linear discriminative model, label encoder will be transformed into one-hot encoder.
- Setting for the before-mentioned data set is：

label	mean	covariance
A	$[-1, -1, -0]$	$[[1, 0, 0], [0, 1, 0], [0, 0, 1]]$
B	$[1, 1, 3]$	$[[1, 0, 0], [0, 1, 0], [0, 0, 3]]$
C	$[2, 4, 2]$	$[[2, 0, 0], [0, 1, 0], [0, 0, 1]]$

New customized dataset can also be configured by `source.py`, the method is listed in next section

Linear Generative Model

Since the data is generated by gaussian distribution, the probabilistic generative model based on **gaussian class-conditional densities** is chosen for classifying the data.

- The density for class C_k is given by

$$p(x|C_k) = \frac{1}{(2\pi)^{D/2} \cdot |\Sigma|^{1/2}} exp\{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\}$$

- Since we have 3 classes, The posterior probability for class C_k can be presented as

$$\begin{aligned} p(C_k|x) &= \frac{p(x|Ck)p(C_k)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2) + p(x|C_3)p(C_3)} \\ &= \frac{\exp \alpha_k}{\sum_k \exp\{\alpha_k\}} = \sigma\{\alpha\} \\ \alpha_k &= \ln p(x|C_k)p(C_k) = w_k^T x + b_k \end{aligned}$$

  Plug into the distribution of x, then we can get

$$\begin{aligned} w_k &= \Sigma^{-1}\mu_k \\ b_k &= -\frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \ln p(C_k) \end{aligned}$$

  Data from different classes is assumed to share covariance matrix Σ, which can be approximated as $p_1 \cdot \Sigma_1 + p_2 \cdot \Sigma_2 + p_3 \cdot \Sigma_3$. Σ_k can be achieved through `np.cov()` and μ_k can be achieved through `np.mean()` on each axis.

- By computing $w_k, b_k (k = 1, 2, 3)$ from training data, we can classify data by computing $p(C_k|x), k = 1, 2, 3$ and find class k which produces maximal $p(C_k|x)$.

Linear Discriminative Model

Softmax regression can be used to do multiple-label classification.

- The dataset can be expressed as $(x^{(i)}, y^{(i)}), i = 1, 2, \dots, n$, and in this assignment, I set $x^{(i)}$ as 3-dimensional variant, $y^{(i)}$ is expressed in one-hot encoder, which has:

$$y_k^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \in \text{class } k \\ 0 & otherwise \end{cases}$$

- For each class k, we have $val_k = w_k^T x + b_k$, while b is the shared bias
- Then we can get the value of softmax function $p(y_k^{(i)} = 1) = \frac{\exp(val_k(x))}{\sum_i (val_i(x))}$, the predict label of softmax regression is $k, s.t. s_k = \max_i s_i$
- Use cross entropy function to compute the loss:

$$J(w,b) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{3} y_k^{(i)} \log(p(y_k^{(i)} = 1))$$
$$\delta_{w_k} J = -\frac{1}{n}\sum_{i=1}^{n} x^{(i)} [p(y_k^{(i)} = 1) - y_k^{(i)}]$$
$$\delta_{b_k} J = -\frac{1}{n}\sum_{i=1}^{n} [p(y_k^{(i)} = 1) - y_k^{(i)}]$$

- Use gradient descent to train the weight and the bias, α is the learning rate.

$$w_k = w_k + \alpha \cdot \delta_{w_k}$$
$$b_k = b_k + \alpha \cdot \delta_{b_k}$$

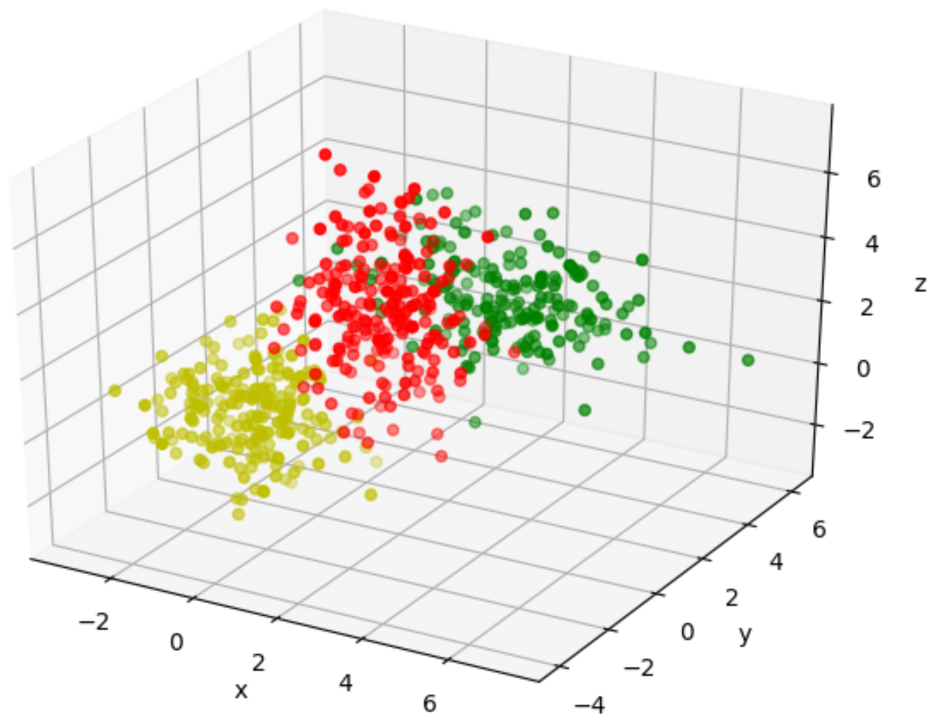
- The model will be trained with n iterations to get less loss on training data and valid data.

Experiment

01 First Experiment

```
mean = [[-1, -1, -0],
        [1, 1, 3],
        [2, 4, 2]]
cov  = [[[1, 0, 0], [0, 1, 0], [0, 0, 1]],
        [[1, 0, 0], [0, 1, 0], [0, 0, 3]],
        [[2, 0, 0], [0, 1, 0], [0, 0, 1]]]
```

The data generated can be ploted as follows, with label A colored yellow, label B colored red and label C colored green. class A, B, C have similar size of data while the size of the training data is 600 (the ratio of training data: valid data : test data is $0.6 : 0.2 : 0.2$). We can see that class **B** has overlap with both A and C:

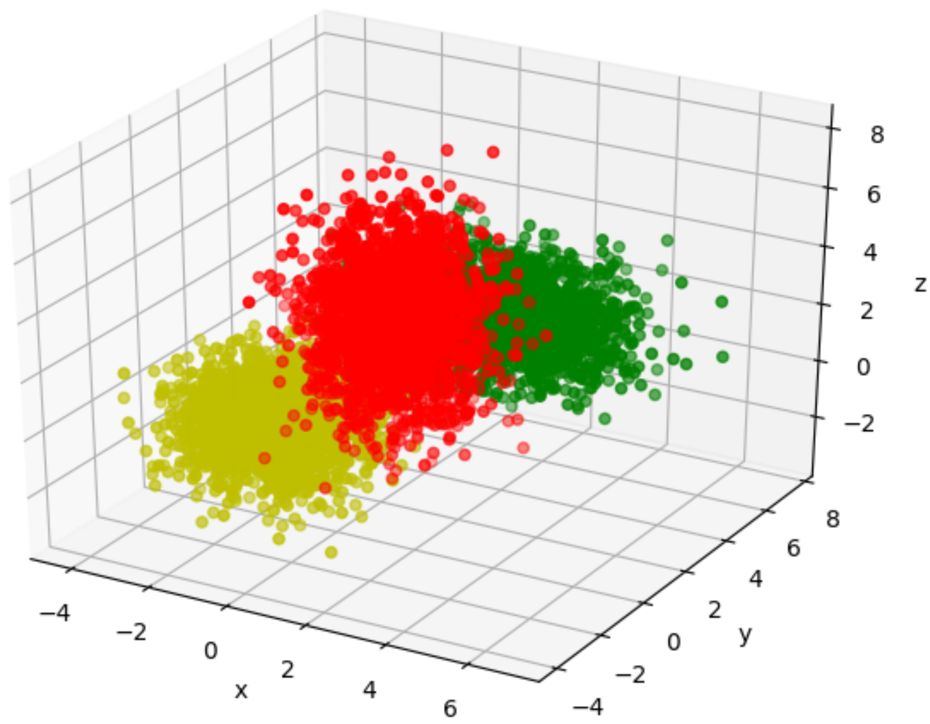



```
generative model: Accuracy: 0.8700
Recall of class 0: 0.9437
Recall of class 1: 0.7692
Recall of class 2: 0.8701
Iteration:[0], loss:[1.2741]
...
Iteration:[9000], loss:[0.1319]
discriminative model: Accuracy: 0.9300
Recall of class 0: 0.9296
Recall of class 1: 0.9423
Recall of class 2: 0.9221
```

- The accuracy of generative model is **lower** than discriminative model.
- Generative model **performs bad in discriminating class B** since class B has overlaps with other 2 classes of data.

02 Enlarging the Size

10000 samples are generated with same distribution, 6000 samples are used for training, while 2000 samples are used for validation and the rest for testing.

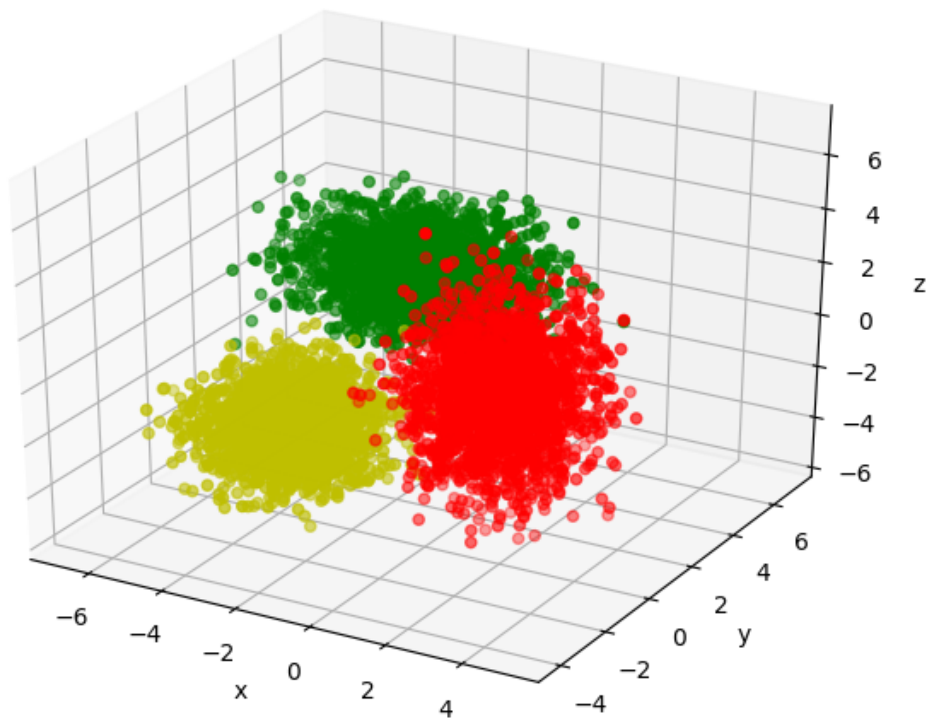



```
generative model: Accuracy: 0.8745
Recall of class 0: 0.9578
Recall of class 1: 0.7031
Recall of class 2: 0.9626
Iteration:[0], loss:[2.3145]
Iteration:[1000], loss:[0.1725]
Iteration:[2000], loss:[0.1571]
Iteration:[3000], loss:[0.1515]
...
Iteration:[19000], loss:[0.1452]
discriminative model: Accuracy: 0.9350
Recall of class 0: 0.9714
Recall of class 1: 0.8857
Recall of class 2: 0.9478
```

- Both generative model and discriminative **show better results** than training on smaller dataset.
- However, when data increases, generative model **doesn't perform better in discriminating class B** since when the size of the data is larger, the mean value of the data is closer to the mean of the distribution. However, the 'outliers' in class B still divagate from others and even fall in the space where probability of other classes‘ data s appears is higher. Since generative model assumes the prior distribution of data, it will have trouble in areas where distributions overlap (which is linear inseparable).

03 Changing the Overlap

Setting $\mu = [[-3, -1, -2], [2, 0, 0], [-2, 3, 2]]$, the data generated are plotted as follows, the size of the whole dataset is 10000.



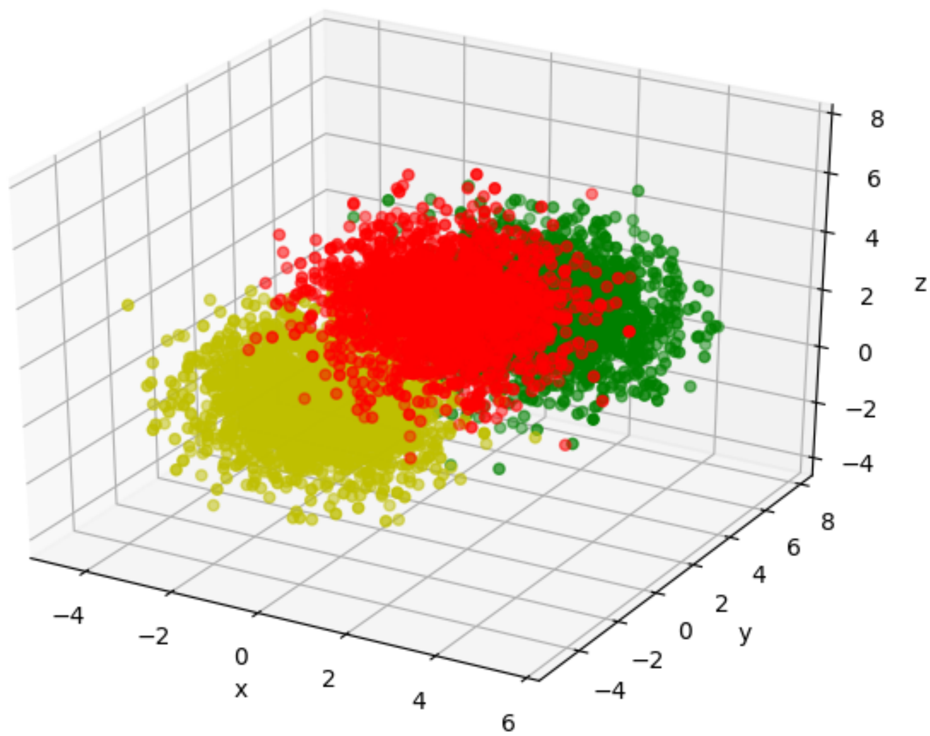

```
generative model: Accuracy: 0.9860
Recall of class 0: 0.9986
Recall of class 1: 0.9731
Recall of class 2: 0.9858
Iteration:[0], loss:[0.4389]
...
Iteration:[19000], loss:[0.0337]
discriminative model: Accuracy: 0.9920
Recall of class 0: 0.9957
Recall of class 1: 0.9880
Recall of class 2: 0.9921
```

- With the overlap smaller, the accuracy of generative model **rises** significantly.
- The accuracy of discriminative model **also rises**.
- Apparently, this dataset introduces less errors in using 2-d hyperplane to seperate the 3-d space.

Other Findings

04 Same Covariance

Since in generative model, assumption that the distribution of different classes**shares the same covariance matrix** is made in order to make the model linear. However, in the experiment above, the covariance matrices of different classes differ. By setting each matrix as $cov = [[1.5, 0, 0], [0, 1.5, 0], [0, 0, 1.5]]$, data generated are plotted as below

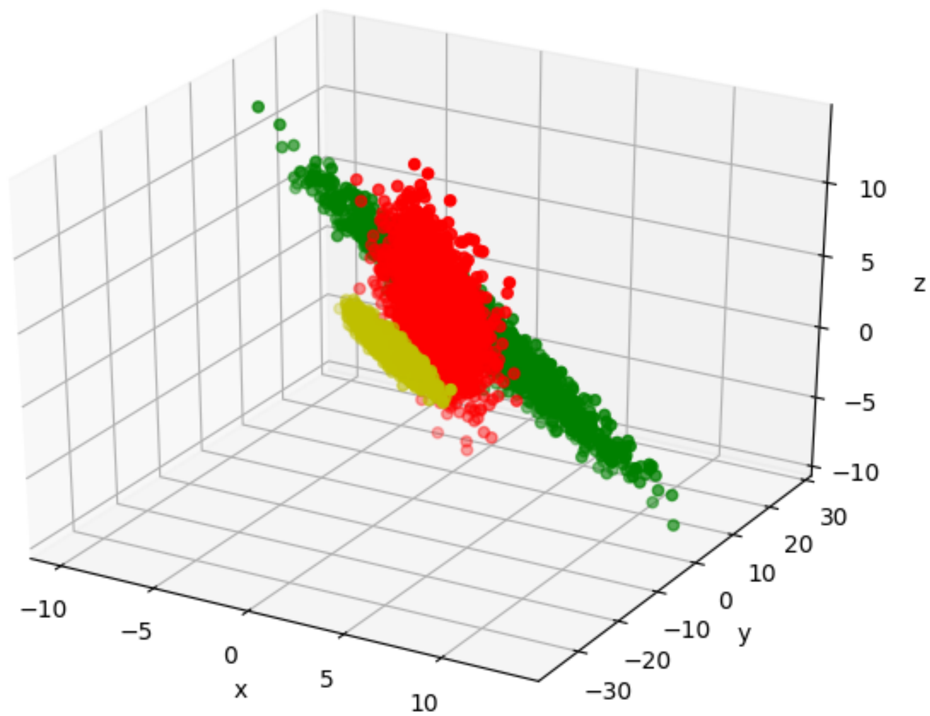



```
generative model: Accuracy: 0.9160
Recall of class 0: 0.9434
Recall of class 1: 0.8924
Recall of class 2: 0.9137
```

- Generative model rises in accurarcy as we expected

05 Different Covariance

In opposite, if we **enlarge the difference between covariance matrices**. The data generated can be plotted as below

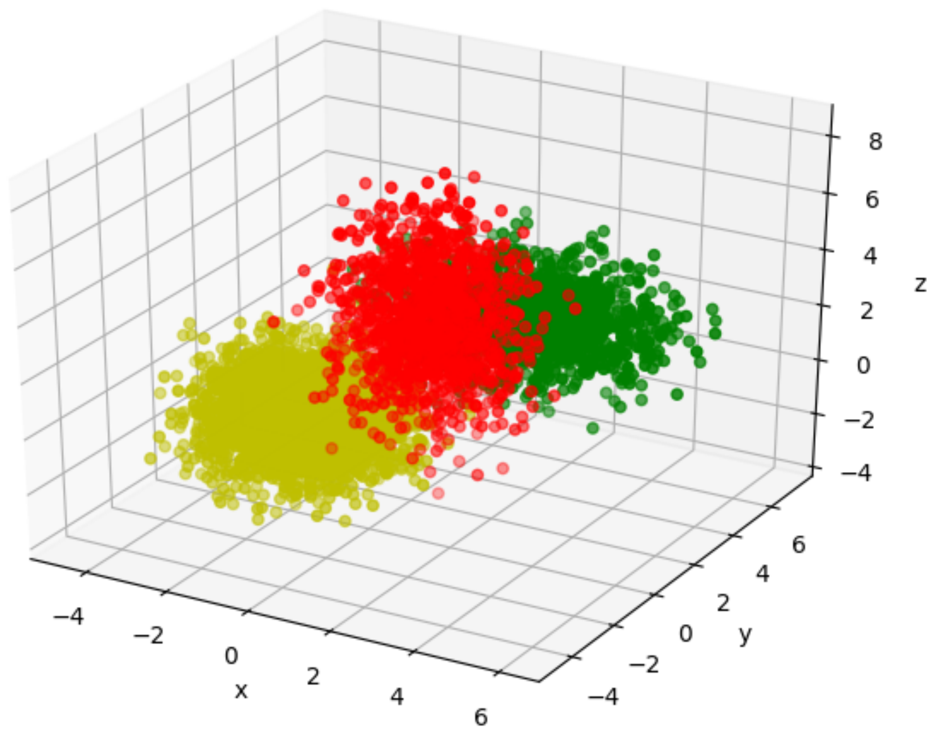



```
generative model: Accuracy: 0.5455
Recall of class 0: 0.6667
Recall of class 1: 0.9230
Recall of class 2: 0.0480
discriminative model: Accuracy: 0.9440
Recall of class 0: 0.9955
Recall of class 1: 0.9577
Recall of class 2: 0.8784
```

- Though it seems that the task can be dealt with linear model more easily. The generative model **behaves bad.**
- Discrminative still have good results.

06 Skew Dataset

By Assigning 6000 samples in class A, 2000 samples in class B and 2000 samples in class C. With exactly same μ and Σ with experiment `02`, data generated are plotted as below.




```
generative model: Accuracy: 0.9270
Recall of class 0: 0.9876
Recall of class 1: 0.7286
Recall of class 2: 0.9418
discriminative model: Accuracy: 0.9570
Recall of class 0: 0.9856
Recall of class 1: 0.8897
Recall of class 2: 0.9424
```

- The discriminative power of generative model and discriminative model **doesn't change a lot** (Only recall is meangingful for comparison). Since 2000 samples is also sufficient to featurize the data.