PRML ASSIGNMENT I

18307130126　Zihang Lin

Part I

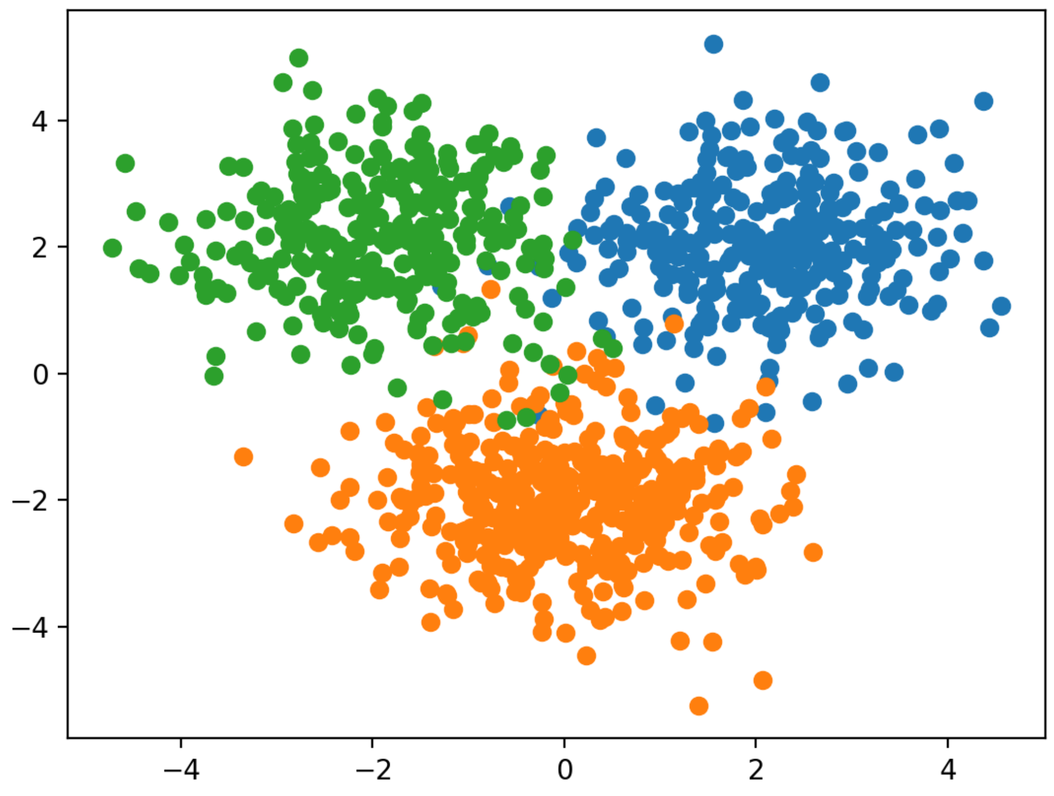

Figure 1 Generate Data

In the "generate_data()" function, we generate multivariate normal distribution using **np.random.multivariate_normal()**.

The data are generated in a two-dimensional space, with consideration of the identity matrix as the default covariance matrix and default **mean** values of (2,2), (0,-2), (-2,2). We also set **prior probabilities** for the Gaussian distribution of three different labels, which are 0.3, 0.4, 0.3, respectively.

The data distribution is shown in Figure 1.

Part II

Discriminative Model

The linear model is $f(\mathbf{x}) = \mathbf{W^T x + b}$, where $\mathbf{W} \in \mathbf{R}^{c \times d}$ is a weighted matrix, $\mathbf{x} \in \mathbf{R}^{d \times 1}$ is a vector representing single sample data.

For multi-class label classification problems, the conditional probability of softmax regression prediction belonging to category c is:

$$\begin{aligned} p(y = c|\boldsymbol{x}) &= softmax(\boldsymbol{w_c^T x}) \\ &= \frac{exp(\boldsymbol{w_c^T x})}{\sum_{c'=1}^{C} exp(\boldsymbol{w_{c'}^T x})} \end{aligned} \tag{2.1}$$

Where w_c is the weight vector of class c.

Then The decision function of Softmax regression can be expressed as

$$\hat{y} = \arg\max_c p(y = c|\boldsymbol{x}) \tag{2.2}$$

Using the cross-entropy loss function, the risk function of the Softmax regression model is

$$L(\boldsymbol{W}) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} \boldsymbol{y_c^{(n)}} log \boldsymbol{\hat{y}_c^{(n)}} \tag{2.3}$$

The gradient of the risk function $L(\mathbf{W})$ with respect to $\mathbf{W}$ is

$$\frac{\partial L^{(n)}(\mathbf{W})}{\partial \mathbf{w}_c} = -\mathbf{x}^{(n)}(I(\mathbf{y}^{(n)} = c) - \hat{\mathbf{y}}^{(n)}) \tag{2.4}$$

Then we use **Mini-Batch Stochastic Gradient Descent** and iteratively update by the following expression

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \alpha(\sum_{n=1}^{k} \mathbf{x}^{(n)}(\mathbf{y}^{(n)} - \hat{\mathbf{y}}_{\mathbf{W}_t}^{(n)})^T) \tag{2.5}$$

Where α is the learning rate, k is the batch size, and $\mathbf{x}^{(n)}$ is chosen randomly. Here we set α = 0.01, k = 10 and epoch = 1000. As a result, we achieve good performance with an accuracy of **0.97**.

Generative Model

Inspired by Bayesian estimation, we achieve the goal of predicting the probability of each label of input data $\mathbf{x}$ with the following formula

$$p(C_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_k)p(C_k)}{\Sigma_j p(\boldsymbol{x}|C_j)p(C_j)} \tag{2.6}$$

Where $p(C_k)$ is estimated by the frequency of occurrence in the given data, while assuming that the class-conditional densities $p(\mathbf{x}|C_k)$ are Gaussians, and have the same covariance matrix $\mathbf{\Sigma}$:

$$p(\boldsymbol{x}|C_k) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\} \quad (2.7)$$

Then we can fit the parameters $\boldsymbol{\mu}_k, p(C_k), \boldsymbol{\Sigma}$ to this model using maximum likelihood learning. Given i.i.d. data $\mathbf{X} = (x_1, \ldots, x_N)^T$, the log likelihood function is given by

$$ln\, p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{ND}{2}ln(2\pi) - \frac{N}{2}ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{x}_n-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n-\boldsymbol{\mu}) \quad (2.8)$$

Set the derivative of the log likelihood function to zero,

$$\frac{\partial}{\partial\boldsymbol{\mu}}ln\, p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n-\boldsymbol{\mu}) = 0 \quad (2.9)$$

and solve it to obtain

$$\boldsymbol{\mu}_{ML} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n \quad (2.10)$$

Similarly,

$$\boldsymbol{\Sigma}_{ML} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n-\boldsymbol{\mu}_{ML})(\boldsymbol{x}_n-\boldsymbol{\mu}_{ML})^T \quad (2.11)$$

So we can use the mathematical analysis method to directly calculate the parameters, and make prediction by choosing the label with the highest posterior probability $p(C_k|\boldsymbol{x})$. As a result, we achieve good performance with an accuracy of **0.967**.

Part III

First, we reorganize the scale of our dataset, setting the number of nodes from 1000 to 10000.

# of nodes	1000	2000	5000	8000	10000
Discriminative Model	**0.970**	0.952	0.963	0.913	0.956
Generative Model	0.967	**0.973**	**0.973**	**0.972**	**0.970**

Table 1: The ACC in the two models with different # of nodes

As the scale of dataset increases, the Generative Model seems to perform better than the Discriminative Model. Intuitively, since we generate the data using a Gaussian distribution, and the Generative Model also assumes the Gaussian distribution, it is not surprising the generative model performs better than Discriminative Model when dataset scale becomes large here. However, as the number of parameters learned in the Generative Model is $O(n^2)$, when the dataset scale is quite large, the Generative Model is no longer applicable. Besides, when the true distribution of the data is not Gaussian, the performance of our Generative Model is also limited.

Then we tune the **learning rate** α , **number of iterations** $epoch$, and the **batch size** k to observe the difference in the performance of the Discirminative Model. (under the default mean values, covariance matrix and data number $n = 1000$)

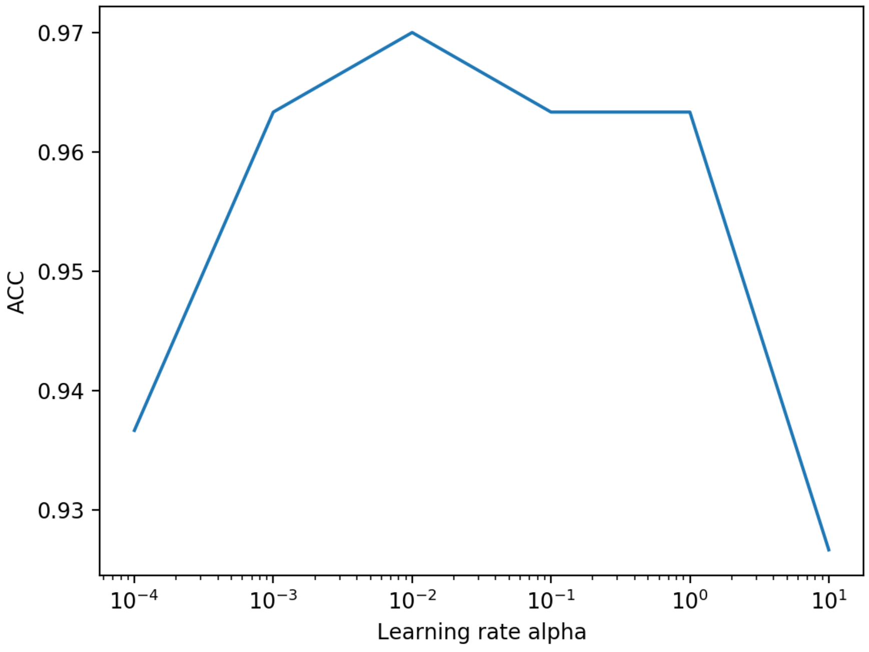

Figure 2 different learning rate of ACC

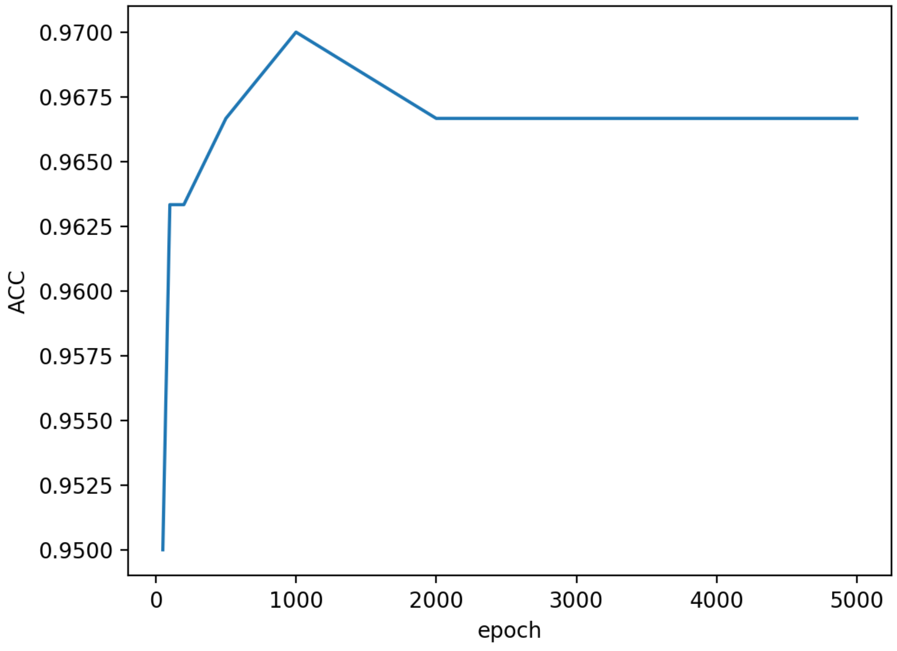

Figure 3 different epoches of ACC

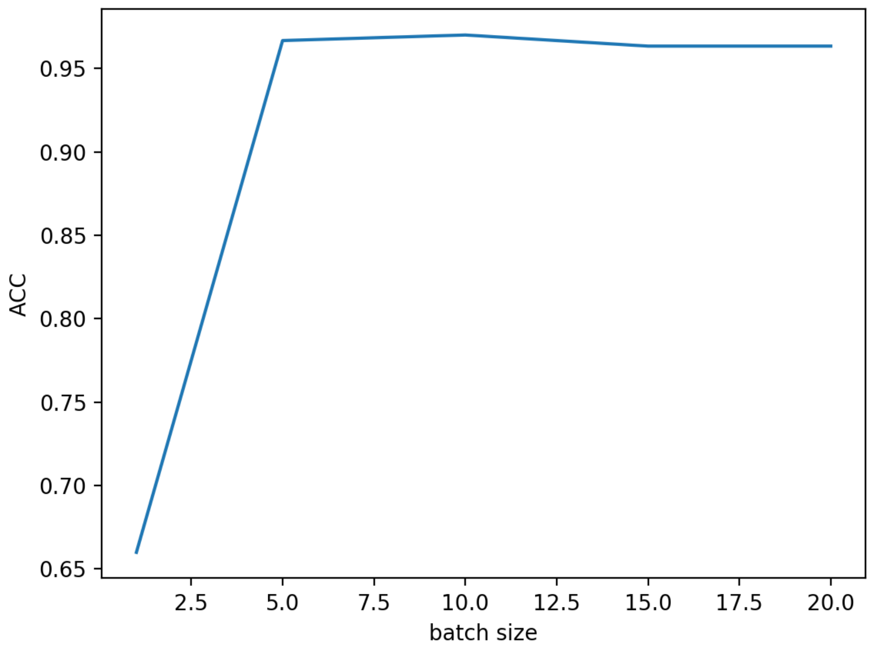

Figure 4 different batch sizes of ACC

So we choose learning rate α as 0.01, $epoch$ as 1000, and batch size k as 10 in our Discriminative Model.

As for adjusting the overlap between different gaussian distributions, we found that when the data distribution overlap becomes larger, the model will gradually lose its accuracy.

When we set the **mean** values as (1,1), (0,-1), (-1,1), the accuracy of Discriminative Model and Generative Model remains 0.703, 0.810, respectively.

When we set the **mean** values as (0.5, 0.5), (0,-0.5), (-0.5,0.5), the accuracy of Discriminative Model and Generative Model remains 0.527, 0.583, respectively.

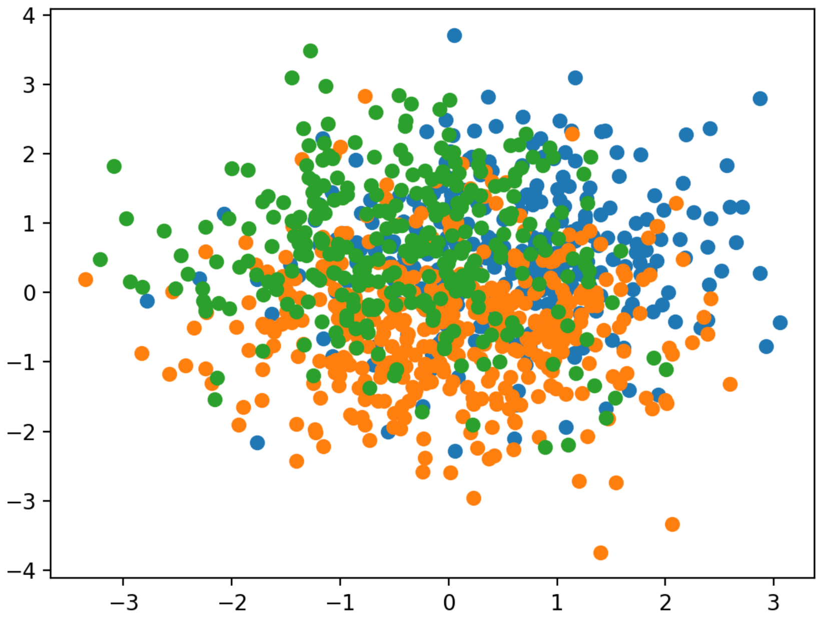

Figure 5 Larger overlap data

Summary

This assignment gave me a deeper understanding of linear discriminative and generative models in machine learning.

To run the code, you can run this instruction directly in the folder where the file is located

```
python source.py
```

References

"Neural Networks and Deep Learning" by Xipeng Qiu

"Pattern Recognition and Machine Learning" by Bishop