TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN MÔN HỌC

Môn: Phân tích thống kê dữ liệu nhiều biến

Học kỳ II (2021-2022)

ĐỀ TÀI

CLASSIFICATION

Giảng viên hướng dẫn: PGS.TS Lý Quốc Ngọc

Nhóm đồ án: Warriors

Thành viên: 19120456 – Nguyễn Phan Quốc Bảo (nhóm trưởng)

19120656 – Phan Văn Thắng

19120716 – Lê Trọng Việt

Mục lục

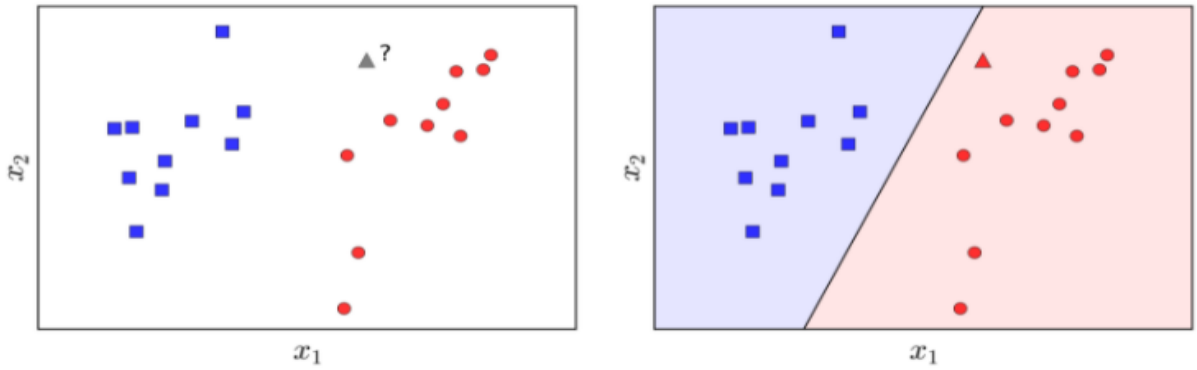
1 Giới thiệu vấn đề

1.1 Khái niệm

Với sự bùng nổ lượng lớn dữ liệu như hiện nay, phân lớp dữ liệu (**classification**) đang là một trong những hướng nghiên cứu chính. Chúng ta cần phân biệt rõ giữa phân lớp dữ liệu (**classification**) và phân biệt dữ liệu (**discrimination**):

- Discrimination (**separation**) nhằm mô tả những đặc trưng khác biệt của dữ liệu quan sát được so với quần thể.
- Classification (**allocation**) sắp xếp dữ liệu quan sát được vào 2 hay nhiều lớp đã được gán nhãn trước.

Hình minh họa bài toán phân lớp:



Nguồn ảnh: https://machinelearningcoban.com/2017/01/21/perceptron/

Ở hình minh họa trên, số quần thể là 2. Tại hình 1, tam giác màu xám vẫn chưa thuộc quần thể nào. Đến hình 2, tam giác đã được phân vào quần thể màu đỏ.

Dù phân lớp (**classification**) và hồi quy tuyến tính (**linear regression**) đều là bài toán học có giám sát (**supervised learning**), nhưng khác với dự đoán dùng để xây dựng mô hình với các giá trị liên tục (**continous value**), thì phân lớp xây dựng mô hình với các giá trị rời rạc (**discrete value**) hay những nhãn xác định (**categorical label**). Có nghĩa là phân lớp thao tác với dữ liệu có giá trị đã được biết trước.

Quá trình phân lớp dữ liệu gồm 2 bước:

- Bước 1: quá trình phân tích phân biệt (discrimination)

Đầu vào là 1 tập dữ liệu training gồm các bộ dữ liệu (*data tuple*) đã được gán nhãn. Sau đó, mô hình sẽ mô tả đặc trưng khác biệt của bộ dữ liệu mình đang quan sát so với các bộ còn lại. Đầu ra bước này là quy tắc để phân lớp (mô hình phân lớp).

- Bước 2: quá trình phân lớp (classification)

Ở giai đoạn này, chúng ta sử dụng tập dữ liệu mới chưa được gán nhãn, sử dụng quy tắc phân lớp đã xây dựng, tiến hành đánh nhãn cho tập dữ liệu mới.

1.2 Ứng dụng

Trong dự báo thời tiết, dựa vào các thông số độ ẩm, sức gió, nhiệt độ,... của ngày hôm nay và những ngày trước đó để dự báo ngày mai nắng hay mưa.

Trong y khoa, dựa vào các đặc điểm kích thước, tốc độ phát triển,... của khối u để dự đoán khối u lành tính hay ác tính.

2 Phân lớp bằng cực tiểu ECM

2.1 Phân 2 lớp bằng cực tiểu ECM

2.1.1 Phát biểu bài toán

Cho $X^T = [X_1, X_2, \dots, X_n]$ là một vector ngẫu nhiên.

$f_1(x)$, $f_2(x)$ lần lượt là hàm mật độ xác suất của 2 quần thể π_1 và π_2.

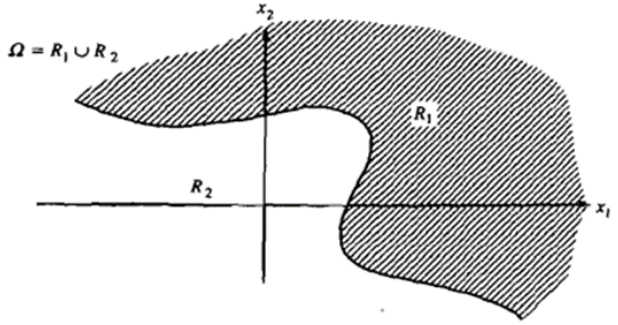
Vấn đề đặt ra: Với bộ giá trị dữ liệu cụ thể $X = x$, ta phải tìm quy tắc phân lớp của từng quần thể để xác định $X = x$ sẽ được gán nhãn π_1 hay π_2 sao cho xác suất phân lớp lỗi là ít nhất có thể.

2.1.2 Phương pháp

- Xét Ω là không gian mẫu các trường hợp gán nhãn có thể cho X.

R_1 là tập con của Ω khi phân loại X vào quần thể $\pi 1$.

$R_2 = \Omega - R_1$ là tập con của Ω mà chúng ta phân loại X vào $\pi 2$.



Hình minh họa không gian mẫu với số chiều p = 2

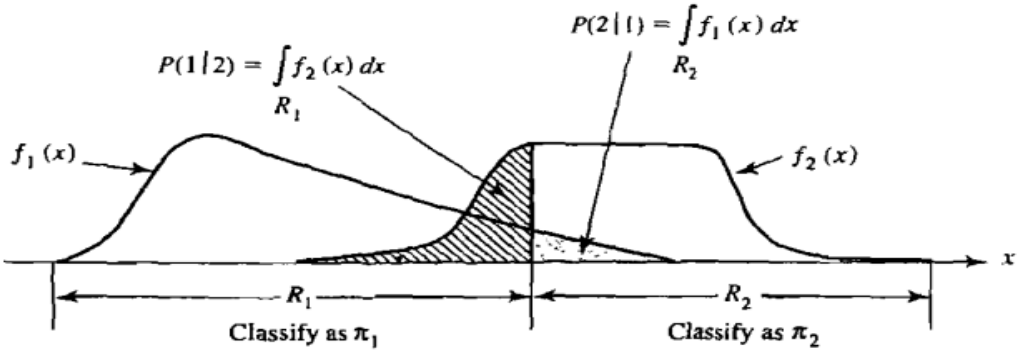
- Xác suất có điều kiện của việc phân lớp X vào π_2 là sai (khi X thuộc về π_1):

$$P(2|1) = P(X \in R_2|\pi_1) = \int_{R_2} f_1(x)dx \qquad (1)$$

- Xác suất có điều kiện của việc phân lớp X vào π_1 là sai (khi X thuộc về π_2):

$$P(1|2) = P(X \in R_1|\pi_2) = \int_{R_1} f_2(x)dx \qquad (2)$$

Dưới đây là hình ảnh minh họa cho xác suất phân lớp sai:



Giải thích cho hình minh họa
Đường thẳng vuông góc với trục x chính là đường phân chia 2 lớp $\boldsymbol{\pi_1}$ và $\boldsymbol{\pi_2}$.
Vùng sọc thể hiện việc phân lớp sai đối tượng thuộc lớp $\boldsymbol{\pi_2}$ vào lớp $\boldsymbol{\pi_1}$
Vùng sọc thể hiện việc phân lớp sai đối tượng thuộc lớp $\boldsymbol{\pi_1}$ vào lớp $\boldsymbol{\pi_2}$

Việc của chúng ta là đi tìm đường thẳng đó, sao cho tổng diện tích của 2 phần phân lớp sai (trong hình là vùng sọc và vùng chấm).

Ta tính toán được: đường thẳng trên đi qua giao điểm của 2 hàm mật độ xác suất $\boldsymbol{f_1(x)}$ và $\boldsymbol{f_2(x)}$ sẽ cho kết quả tốt nhất (tổng diện tích của 2 phần phân lớp sai là nhỏ nhất).

Gọi p_1 và p_2 lần lượt là xác suất của một đối tượng thuộc về quần thể π_1 và π_2, khi đó: $p_1 + p_2 = 1$

Khi đó ta có được các xác suất có điều kiện sau:

$$P(Phân\ lớp\ sai\ X\ vào\ \pi_1) = P(X \in R_1|\pi_2)P(\pi_2) = P(1|2)p_2$$

$$P(Phân\ lớp\ sai\ X\ vào\ \pi_2) = P(X \in R_2|\pi_1)P(\pi_1) = P(2|1)p_1$$

$$P(Phân\ lớp\ đúng\ X\ vào\ \pi_1) = P(X \in R_1|\pi_1)P(\pi_1) = P(1|1)p_1$$

$$P(Phân\ lớp\ đúng\ X\ vào\ \pi_2) = P(X \in R_2|\pi_2)P(\pi_2) = P(2|2)p_2$$

- Gọi $\mathbf{c(i|j)}$ là chi phí phân lớp sai đối tượng vào $\boldsymbol{i}$ khi đối tượng đó thuộc $\boldsymbol{j}$, với $\boldsymbol{i}$, $\boldsymbol{j} = 1, 2$. Ta có bảng chi phí phân loại sai như hình dưới.

		Classify as	
		π_1	π_2
Truth	π_1	0	c(2|1)
	π_2	c(1|2)	0

- Từ đó, ta có hàm tính độ lỗi ECM (***excepted cost of misclassification***) của việc phân lớp sai:

$$ECM = c(2|1)P(2|1)p_1 + c(1|2)P(1|2)p_2 \qquad (3)$$

- Để việc phân lớp đạt hiệu quả tốt nhất, ta cần cực tiểu hóa hàm tính độ lỗi ECM:

Thay (1) và (2) vào (3) ta được:

$$ECM = c(2|1)p_1 \int_{R_2} f_1(x)dx + c(1|2)p_2 \int_{R_1} f_2(x)dx \qquad (4)$$

Vì $R_1 \cup R_2 = \Omega$, do đó:

$$\int_{R_2} f_1(x)dx + \int_{R_1} f_1(x)dx = 1$$

- Thay $\int_{R_2} f_1(x)dx = 1 - \int_{R_1} f_1(x)dx$ vào (4):

$$ECM = c(2|1)p_1\left(1 - \int_{R_1} f_1(x)dx\right) + c(1|2)p_2 \int_{R_1} f_2(x)dx$$

$$\Rightarrow \quad ECM = \int_{R_1} [c(1|2)p_2 f_2(x) - c(2|1)p_1 f_1(x)]dx + c(2|1)p_1$$

Để ECM đạt giá trị nhỏ nhất thì:

$$\int_{R_1} [c(1|2)p_2 f_2(x) - c(2|1)p_1 f_1(x)]dx \leq 0$$

$$\Rightarrow \quad c(1|2)p_2 f_2(x) - c(2|1)p_1 f_1(x) \leq 0$$

Đưa ra kết luận, giá trị X được phân lớp π_1 khi:

$$\boldsymbol{R_1:} \quad \frac{f_1(x)}{f_2(x)} \geq \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{p_2}{p_1}\right)$$

- Tương tự ta thay $\int_{R_1} f_2(x)dx = 1 - \int_{R_2} f_2(x)dx$ vào (4):

$$ECM = c(2|1)p_1 \int_{R_2} f_1(x)dx + c(1|2)p_2(1 - \int_{R_2} f_2(x)dx)$$

$$\Rightarrow \quad ECM = \int_{R_2} [c(2|1)p_1 f_1(x) - c(1|2)p_2 f_2(x)]dx + c(1|2)p_2$$

Để ECM đạt giá trị nhỏ nhất thì:

$$\int_{R_2} [c(2|1)p_1 f_1(x) - c(1|2)p_2 f_2(x)]dx \leq 0$$

$$\Rightarrow \quad c(2|1)p_1 f_1(x) - c(1|2)p_2 f_2(x) \leq 0$$

Đưa ra kết luận, giá trị X được phân lớp π_2 khi:

$$\boldsymbol{R_2:} \quad \frac{f_1(x)}{f_2(x)} \leq \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{p_2}{p_1}\right)$$

Ví dụ 2.1.1.1

Cho 2 quần thể π_1, π_2 có hàm mật độ xác suất lần lượt là:

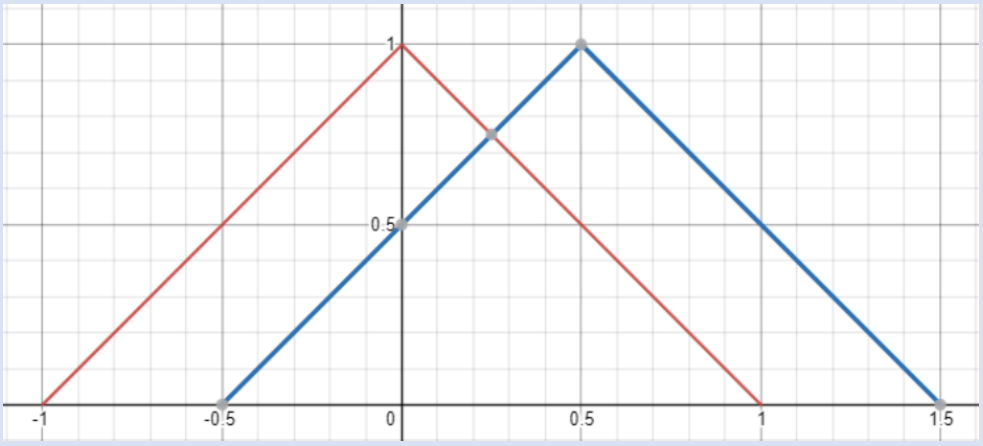
$$f_1(x) = \begin{cases} 1 - |x| \ (-1 \le x \le 1) \\ 0 \ (\text{các trường hợp khác}) \end{cases}$$

$$f_2(x) = \begin{cases} 1 - |x - 0.5| \ (-0.5 \le x \le 1.5) \\ 0 \ (\text{các trường hợp khác}) \end{cases}$$

a) Xác định quy luật phân lớp khi $p_1 = p_2, c(1|2) = c(2|1)$

b) Xác định quy luật phân lớp khi $p1 = 0.2, c(1|2) = c(2|1)$

c) Xác định quy luật phân lớp khi $p1 = p2,\ 2c(1|2) = c(2|1)$



$$R_1: \frac{f_1(x)}{f_2(x)} \ge \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{p_2}{p_1}\right)$$

a) $p_1 = p_2, c(1|2) = c(2|1)$

$$R_1: \frac{f_1(x)}{f_2(x)} \ge 1 \Leftrightarrow 1 - |x| \ge 1 - |x - 0.5| \Leftrightarrow -0.5 \le x \le 0.25$$

$$\Leftrightarrow x \le 0.25$$

b) $p1 = 0.2, c(1|2) = c(2|1)$

$$\boldsymbol{R_1}: \frac{\boldsymbol{f_1(x)}}{\boldsymbol{f_2(x)}} \geq \mathbf{4} \Leftrightarrow \mathbf{1} - |\boldsymbol{x}| \geq \mathbf{4} - \mathbf{4}|\boldsymbol{x} - \mathbf{0.5}| \Leftrightarrow -\mathbf{0.5} \leq \mathbf{x} \leq -\frac{\mathbf{1}}{\mathbf{3}}$$

$$\Leftrightarrow \mathbf{x} \leq -\frac{\mathbf{1}}{\mathbf{3}}$$

c) $\boldsymbol{p1} = \boldsymbol{p2},\ \mathbf{2c(1|2)} = \mathbf{c(2|1)}$

$$\boldsymbol{R_1}: \frac{\boldsymbol{f_1(x)}}{\boldsymbol{f_2(x)}} \geq \frac{\mathbf{1}}{\mathbf{2}} \Leftrightarrow \mathbf{1} - |\boldsymbol{x}| \geq \frac{\mathbf{1}}{\mathbf{2}} - \frac{\mathbf{1}}{\mathbf{2}}|\boldsymbol{x} - \mathbf{0.5}| \Leftrightarrow \mathbf{x} \leq \frac{\mathbf{1}}{\mathbf{2}}$$

2.1.3 Phân lớp với 2 quần thể có phân phối chuẩn

Các mô hình phân lớp với quần thể có phân phối chuẩn thường được sử dụng nhiều trong thống kê vì tính đơn giản và hiệu quả cao trên nhiều mô hình thực tế.

Gọi $f_1(x)$ và $f_2(x)$ là 2 hàm mật độ xác suất có phân phối chuẩn của lớp π_1 và π_2 với mean vector và covariance matrix tương ứng là μ_1, Σ_1 và μ_2,Σ_2

Hàm mật độ xác suất của lớp π_1:

$$f_1(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_1|^{1/2}} exp\left[-\frac{1}{2}(x-\mu_1)'\Sigma_1^{-1}(x-\mu_1)\right]$$

Của lớp π_2:

$$f_2(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_2|^{1/2}} exp\left[-\frac{1}{2}(x-\mu_2)'\Sigma_2^{-1}(x-\mu_2)\right]$$

2.1.3.1 Trường hợp $\Sigma_1 = \Sigma_2 = \Sigma$

Thực hiện cực tiểu hàm ECM trên vùng R_1 ta được bất đẳng thức:

$$\frac{f_1(x)}{f_2(x)} \geq \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{p_2}{p_1}\right)$$

$$\Rightarrow exp\left[-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_2)'\Sigma^{-1}(x-\mu_2)\right] \geq \left(\frac{c(1|2)}{c(2|1)}\right)\frac{p_2}{p_1}$$

Tương tự, thực hiện cực tiểu hàm ECM trên vùng R_2:

$$\frac{f_1(x)}{f_2(x)} < \left(\frac{c(1|2)}{c(2|1)}\right)\left(\frac{p_2}{p_1}\right)$$

$$\Rightarrow ex\,p\left[-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(x-\mu_1)+\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(x-\mu_2)\right] < \left(\frac{c(1|2)}{c(2|1)}\right)\frac{p_2}{p_1}$$

Như vậy ta có quy luật phân lớp:

- Gán nhãn một mẫu là π_1 nếu:

$$\exp\left[-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(\mathrm{x}-\mu_1)+\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(\mathrm{x}-\mu_2)\right] \geq \left(\frac{c(1|2)}{c(2|1)}\right)\frac{p_2}{p_1}$$

- Gán nhãn một mẫu là π_2 nếu:

$$\exp\left[-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(\mathrm{x}-\mu_1)+\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(\mathrm{x}-\mu_2)\right] < \left(\frac{c(1|2)}{c(2|1)}\right)\frac{p_2}{p_1}$$

Thực hiện phép biển đổi, ta được quy luật:

$$\begin{aligned}\pi_1&: (\mu_1-\mu_2)'\Sigma^{-1}\mathrm{x}-\frac{1}{2}(\mu_1-\mu_2)'\Sigma^{-1}(\mu_1+\mu_2) \geq \ln\left(\left(\frac{c(1|2)}{c(2|1}\right)\frac{p_2}{p_1}\right)\\ \pi_2&: (\mu_1-\mu_2)'\Sigma^{-1}\mathrm{x}-\frac{1}{2}(\mu_1-\mu_2)'\Sigma^{-1}(\mu_1+\mu_2) < \ln\left(\left(\frac{c(1|2)}{c(2|1}\right)\frac{p_2}{p_1}\right)\end{aligned} \quad (2.1.3.1)$$

Chứng minh (2.1.3.1)

Ta có quy luật phân lớp ban đầu:

$$\boldsymbol{\pi_1: exp\left[-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(\mathrm{x}-\mu_1)+\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(\mathrm{x}-\mu_2)\right] \geq \left(\frac{c(1|2)}{c(2|1)}\right)\frac{p_2}{p_1}}$$

$$\boldsymbol{\pi_2: exp\left[-\frac{1}{2}(x-\mu_1)'\Sigma^{-1}(\mathrm{x}-\mu_1)+\frac{1}{2}(x-\mu_2)'\Sigma^{-1}(\mathrm{x}-\mu_2)\right] < \left(\frac{c(1|2)}{c(2|1)}\right)\frac{p_2}{p_1}}$$

Thực hiện lấy logarit nepe 2 vế 2 bất đẳng thức:

$$\boldsymbol{\pi_1}: -\frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x}-\boldsymbol{\mu_1})'\boldsymbol{\Sigma^{-1}}(\mathbf{x}-\boldsymbol{\mu_1}) + \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x}-\boldsymbol{\mu_2})'\boldsymbol{\Sigma^{-1}}(\mathbf{x}-\boldsymbol{\mu_2}) \geq \ln\left(\left(\frac{\boldsymbol{c(1|2)}}{\boldsymbol{c(2|1}}\right)\frac{\boldsymbol{p_2}}{\boldsymbol{p_1}}\right)$$

$$\boldsymbol{\pi_2}: -\frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x}-\boldsymbol{\mu_1})'\boldsymbol{\Sigma^{-1}}(\boldsymbol{x}-\boldsymbol{\mu_1}) + \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x}-\boldsymbol{\mu_2})'\boldsymbol{\Sigma^{-1}}(\boldsymbol{x}-\boldsymbol{\mu_2}) < \boldsymbol{ln}\left(\left(\frac{\boldsymbol{c(1|2)}}{\boldsymbol{c(2|1}}\right)\frac{\boldsymbol{p_2}}{\boldsymbol{p_1}}\right)$$

Với x,y là 2 vector và A là 1 ma trận đối xứng, Ta có

$$\boldsymbol{x'Ay} = (\boldsymbol{x'Ay})' = \boldsymbol{y'A'x} = \boldsymbol{y'Ax}$$

Từ tính chất trên, ta phân tích vế trái bất đẳng thức:

$$-\frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x}-\boldsymbol{\mu_1})'\boldsymbol{\Sigma^{-1}}(\mathbf{x}-\boldsymbol{\mu_1}) + \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x}-\boldsymbol{\mu_2})'\boldsymbol{\Sigma^{-1}}(\mathbf{x}-\boldsymbol{\mu_2})$$

$$= -\frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x'\Sigma^{-1}x} - \boldsymbol{x'\Sigma^{-1}u_1} - \boldsymbol{u_1'\Sigma^{-1}x} + \boldsymbol{\mu_1'\Sigma^{-1}\mu_1})$$

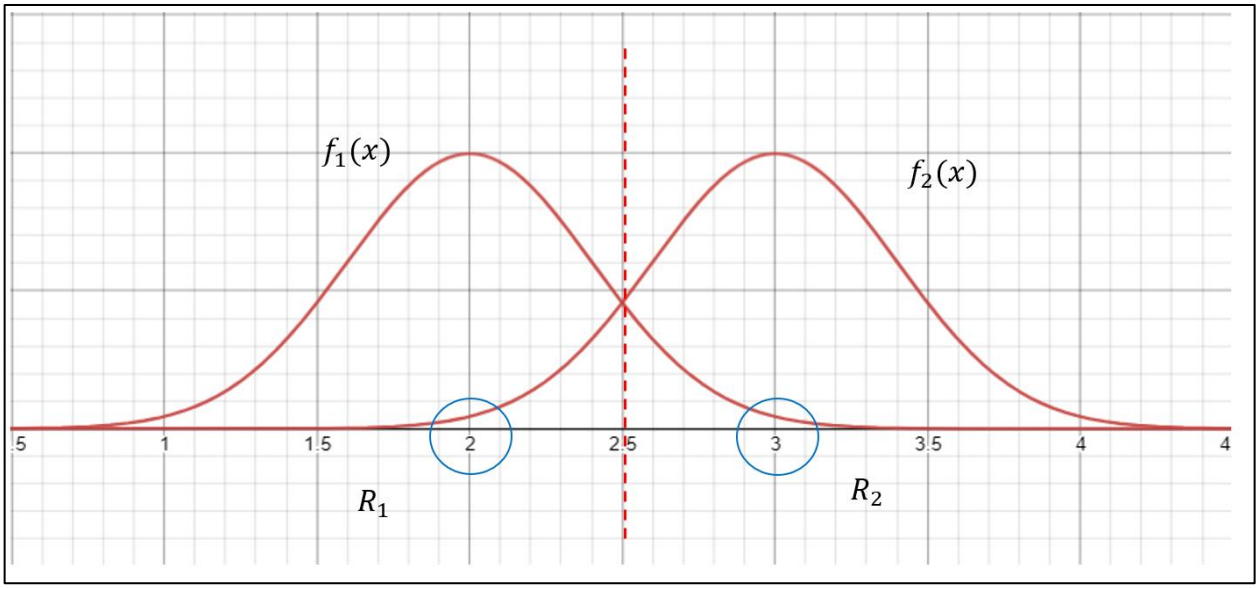
$$+\frac{\mathbf{1}}{\mathbf{2}}\left(\boldsymbol{x'^{\Sigma^{-1}}x} - \boldsymbol{x'\Sigma^{-1}u_2} - \boldsymbol{u_2'^{\Sigma^{-1}}x} + \boldsymbol{\mu_1'^{\Sigma^{-1}\mu_2}}\right)$$

$$= -\frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x'\Sigma^{-1}x} - \mathbf{2}\boldsymbol{u_1'\Sigma^{-1}x} + \boldsymbol{\mu_1'\Sigma^{-1}\mu_1}) + \frac{\mathbf{1}}{\mathbf{2}}\left(\boldsymbol{x'^{\Sigma^{-1}}x} - \mathbf{2}\boldsymbol{u_2'\Sigma^{-1}x} + \boldsymbol{\mu_1'^{\Sigma^{-1}\mu_2}}\right)$$

$$= (\boldsymbol{\mu_1}-\boldsymbol{\mu_2})'^{\boldsymbol{\Sigma^{-1}}}\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}}\left(\boldsymbol{\mu_1'^{\Sigma^{-1}\mu_1}} - \boldsymbol{\mu_2'^{\Sigma^{-1}\mu_2}}\right)$$

$$= (\boldsymbol{\mu_1}-\boldsymbol{\mu_2})'\boldsymbol{\Sigma^{-1}}\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{\mu_1}-\boldsymbol{\mu_2})'\boldsymbol{\Sigma^{-1}}(\boldsymbol{\mu_1}+\boldsymbol{\mu_2})$$

Từ đó ta có được 2 bất đẳng thức ở (2.1.3.1)



Ảnh minh họa: Phân lớp 2 quần thể có phân phối chuẩn với $\mu_1 \neq \mu_2,\ \Sigma_1 = \Sigma_2$

Quy luật phân lớp (2.1.3.1) phụ thuộc vào các giá trị của quần thể như μ_1, μ_2, Σ, Thường sẽ không dễ tìm ra trong thực tế. Vì vậy để áp dụng quy luật phân lớp vào mẫu, ta cần thực hiện xấp xỉ những đại lượng của quần thể bằng đại lượng của mẫu.

Từ dữ liệu tập mẫu với n_1 biến thuộc π_1 và n_2 biến thuộc π_2, ta tính được các đại lượng:

$$\overline{x_1}_{(p\times 1)} = \frac{1}{n_1}\sum_{j=1}^{n_1} x_{1j}\,,\quad S_{1_{(p\times p)}} = \frac{1}{n_1-1}\sum_{j=1}^{n_1}(x_{1j}-\overline{x_1})\,(x_{1j}-\overline{x_1})'$$

$$\overline{x_2}_{(p\times 1)} = \frac{1}{n_2}\sum_{j=1}^{n_1} x_{2j}\,, S_{2_{(p\times p)}} = \frac{1}{n_2-1}\sum_{j=1}^{n_2}(x_{2j}-\overline{x_2})\,(x_{2j}-\overline{x_2})'$$

Ta có thể xấp xỉ các giá trị của quần thể như sau:

- Xấp xỉ trung bình quần thể bằng trung bình mẫu
- Do 2 quần thể có cùng covariance matrix, ta có thể xấp xỉ $\Sigma = S_{pooled}$ theo phương pháp trung bình có trọng số:

$$S_{pooled} = \left[\frac{n_1-1}{(n_1-1)+(n_2-1)}\right]S_1 + \left[\frac{n_2-1}{(n_1-1)+(n_2-1)}\right]S_2$$

Quy luật phân lớp:

$$\pi_1: (\overline{x_1} - \overline{x_2})' S_{pooled}^{-1} \mathrm{x} - \frac{1}{2}(\overline{x_1} - \overline{x_2})' S_{pooled}^{-1}(\overline{x_1} + \overline{x_2}) \geq \ln\left(\left(\frac{c(1|2)}{c(2|1}\right)\frac{p_2}{p_1}\right)$$

$$\pi_2: (\overline{x_1} - \overline{x_2})' S_{pooled}^{-1} \mathrm{x} - \frac{1}{2}(\overline{x_1} - \overline{x_2})' S_{pooled}^{-1}(\overline{x_1} + \overline{x_2}) < \ln\left(\left(\frac{c(1|2)}{c(2|1}\right)\frac{p_2}{p_1}\right)$$

Ví dụ 2.1.3.1

cho $\boldsymbol{n_1 = 11}$ và $\boldsymbol{n_2 = 12}$ là số quan sát thực hiện trên 2 quần thể đa biến $\boldsymbol{\pi_1}$, $\boldsymbol{\pi_2}$ có phân phối chuẩn $\boldsymbol{X_1}$, $\boldsymbol{X_2}$ có cùng covariance matrix $\boldsymbol{\Sigma}$.

Các giá trị trung bình mẫu và ma trận hiệp phương sai mẫu như sau:

$$\widehat{X_1} = \begin{pmatrix} -1 \\ -1 \end{pmatrix},\ \widehat{X_2} = \begin{pmatrix} 2 \\ 1 \end{pmatrix},$$

$$S_1 = \begin{pmatrix} 8 & -1 \\ -1 & 4.5 \end{pmatrix}, S_2 = \begin{pmatrix} 7 & -1.2 \\ -1.2 & 4.8 \end{pmatrix}$$

a) Xây dựng quy tắc phân lớp 2 quần thể $\boldsymbol{X_1, X_2}$

b) Mẫu quan sát mới $\boldsymbol{X_0} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ thuộc về quần thể nào?

Giải

a) $\widehat{X_1} - \widehat{X_2} = \begin{pmatrix} -3 \\ -2 \end{pmatrix}$

$$S_{pooled} = \left(\frac{n_1 - 1}{n_1 + n_2 - 2}\right) S_1 + \left(\frac{n_2 - 1}{n_1 + n_2 - 2}\right) S_2$$

$$= \frac{10}{21}\begin{pmatrix} 8 & -1 \\ -1 & 4.5 \end{pmatrix} + \frac{11}{21}\begin{pmatrix} 7 & -1.2 \\ -1.2 & 4.8 \end{pmatrix} = \begin{pmatrix} 7.5 & -1.1 \\ -1.1 & 4.7 \end{pmatrix}$$

$$S_{pooled}^{-1} = \frac{1}{7.5 \times 4.7 - 1.1 \times 1.1}\begin{pmatrix} 4.7 & 1.1 \\ 1.1 & 7.5 \end{pmatrix} = \begin{pmatrix} 0.13 & 0.03 \\ 0.03 & 0.22 \end{pmatrix}$$

Theo quy tắc ECM, ta cần so sánh biến vô hướng:

$$\hat{y} = (\overline{x_1} - \overline{x_2})' S_{pooled}^{-1} x$$

$$= (-3 \quad -2)\begin{pmatrix} 0.13 & 0.03 \\ 0.03 & 0.22 \end{pmatrix} x = (-0.45 \quad -0.53)x$$

Với giá trị:

$$\hat{m} = \frac{1}{2}(\overline{x_1} - \overline{x_2})' S_{pooled}^{-1}(\overline{x_1} + \overline{x_2}) = \frac{1}{2}(-0.45 \quad -0.53)\begin{pmatrix} 1 \\ 0 \end{pmatrix} = -0.23$$

Biến X_0 thuộc π_1 nếu $(-0.45 \quad -0.53)x \geq -0.23$ và thuộc π_2 trong trường hợp còn lại.

b) $\widehat{y_0} = (-0.45 \quad -0.53)\begin{pmatrix} 0 \\ 1 \end{pmatrix} = -0.53 < -0.23$

Vậy, mẫu quan sát này thuộc về lớp π_2

2.1.3.2 Trường hợp $\Sigma_1 \neq \Sigma_2$

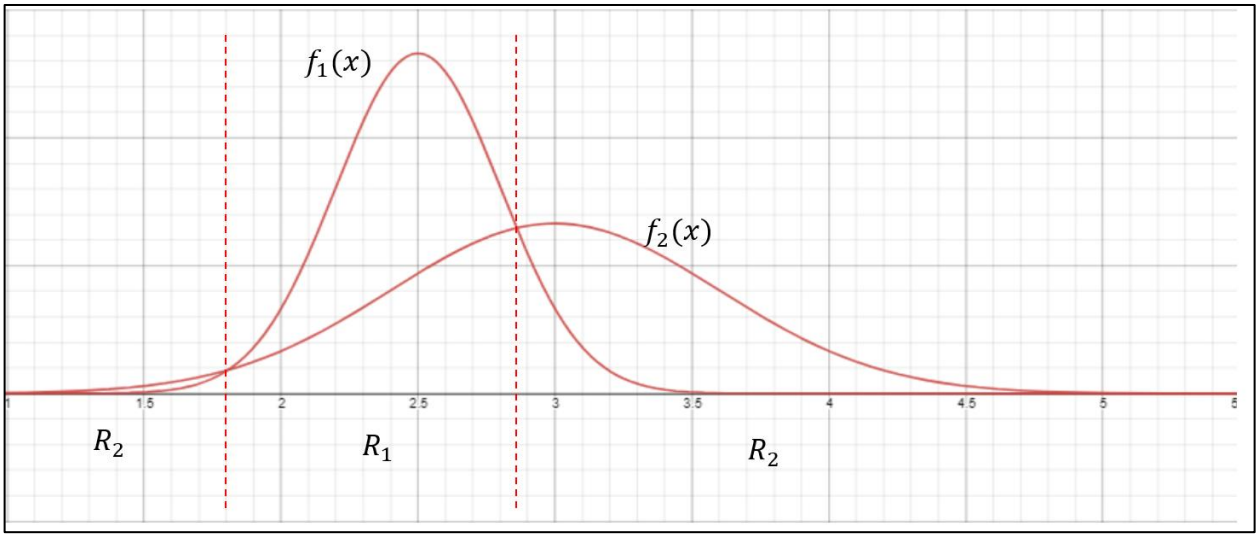
Biến đổi tương tự trường hợp cùng covariance matrix, ta được:

$$R_1: \frac{1}{2}x'(\Sigma_1^{-1} - \Sigma_2^{-1})x + (\mu_1'\Sigma_1^{-1} - \mu_2'\Sigma_2^{-1})x - k \geq \ln\left(\left(\frac{c(1|2)}{c(2|1}\right)\frac{p_2}{p_1}\right)$$

$$R_2: \frac{1}{2}x'(\Sigma_1^{-1} - \Sigma_2^{-1})x + (\mu_1'\Sigma_1^{-1} - \mu_2'\Sigma_2^{-1})x - k < \ln\left(\left(\frac{c(1|2)}{c(2|1}\right)\frac{p_2}{p_1}\right)$$

Trong đó:

$$k = \frac{1}{2}\ln\left(\frac{|\Sigma_1|}{|\Sigma_2|}\right) + \frac{1}{2}(\mu_1\Sigma_1^{-1}\mu_1 - \mu_2\Sigma_2^{-1}\mu_2)$$



Ảnh minh họa: Phân lớp 2 quần thể có phân phối chuẩn với $\mu_1 \neq \mu_2,\ \Sigma_1 \neq \Sigma_2$

2.2 Phân đa lớp bằng cực tiểu ECM

2.2.1 Phát biểu bài toán

- Đặt số lớp (hay số quần thể) là g, $g \geq 2$

  Cho $X^T = [X_1, X_2, \ldots, X_n]$ là một vector ngẫu nhiên.

  $f_1(x),\ f_2(x), \ldots, f_g(x)$ lần lượt là hàm phân phối xác suất của g quần thể $\pi_1, \pi_2, \ldots, \pi_g$

Vấn đề đặt ra: Với bộ giá trị dữ liệu cụ thể $X = x$, ta phải tìm quy tắc phân lớp của từng quần thể để xác định $X = x$ sẽ được gán nhãn 1 trong các quần thể $\pi_1, \pi_2, \ldots, \pi_g$.

2.2.2 Phương pháp

- Đặt $f_i(x)$ là hàm mật độ xác suất tương ứng với quần thể π_i, với $i = 1,\ 2,\ \ldots,\ g$

  Đặt p_i là xác suất phân lớp đúng đối tượng vào lớp π_i

  Đặt $c(k|i)$ là chi phí phân lớp sai đối tượng vào k khi đối tượng đó thuộc i, với $i, k = 1,\ 2 \ldots,\ g$. Khi $k = i$ thì $c(k|i) = c(i|i) = 0$

  Đặt R_k là tập các giá trị x được phân loại vào quần thể π_k

$$P(k|i) = P(\text{phân lớp đối tượng vào lớp } \pi_k | \text{đối tượng thuộc lớp } \pi_i) = \int_{R_k} f_i(x)dx$$

với $i, k = 1,\ 2 \ldots,\ g$. Và $P(i|i) = 1 - \sum_{k=1, k\neq i}^{g} P(k|i)$

- Cũng giống như bài toán phân 2 lớp, ta cần tìm 1 hàm độ lỗi để tính chi phí phân lớp sai cho đối tượng x, và sau đó cực tiểu hóa hàm độ lỗi này.
- Tương tự công thức (3), độ lỗi của việc phân loại sai x từ π_1 thành π_2, hoặc π_3, hoặc π_4,..., hoặc π_g là:

$$ECM(1) = P(2|1)c(2|1) + P(3|1)c(3|1) + \ldots + P(g|1)c(g|1)$$
$$= \sum_{k=2}^{g} P(k|1)c(k|1)$$

Tương tự ta tính được $ECM(2),\ ECM(3), \ldots,\ ECM(g)$.

- Với mỗi $ECM(i)$ có điều kiện, ta nhân với xác suất phân lớp đúng p_i. Ta được hàm tổng độ lỗi ECM:

$$ECM = p_1 ECM(1) + p_2 ECM(2) + \ \ldots\ + p_g ECM(g)$$

$$= p_1 \sum_{k=2}^{g} P(k|1)c(k|1) + p_2 \sum_{k=1,\ k\neq 2}^{g} P(k|2)c(k|2) + \ \ldots\ + p_g \sum_{k=1}^{g-1} P(k|g)c(k|g)$$

$$= \sum_{i=1}^{g} p_i \left[\sum_{k=1,\ k\neq i}^{g} P(k|i)c(k|i) \right]$$

- Để phân lớp đạt hiểu quả tốt nhất, ta cần cực tiểu hóa hàm tổng độ lỗi ECM. Thì khi đó, $\sum_{i=1,\ i\neq k}^{g} p_i f_i(x) c(k|i)$ đạt nhỏ nhất khi ta phân lớp x vào π_k với $k = 1,\ 2, \ldots, g$

Giả sử, tất cả các chi phí phân loại sai bằng nhau ($c(k|i) = const$, với $k \neq i$). Khi đó, bài toán đưa về cực tiểu hóa hàm $\sum_{i=1,\ i\neq k}^{g} p_i f_i(x)$ khi ta phân lớp x vào π_k với $k = 1,\ 2, \ldots,\ g$.

Tổng trên nhỏ nhất, khi số hạng bị bỏ qua $p_k f_k(x)$ đạt giá trị lớn nhất.

Do đó, trong trường hợp các chi phí phân loại sai bằng nhau, để phân lớp x_0 vào quần thể π_k đạt hiểu qua nhất khi:

$$\boldsymbol{p_k f_k(x_0) > p_i f_i(x_0),\ \text{với}\ i = 1,\ 2,\ \ldots,\ g\ \text{và}\ i \neq k}$$

2.2.3 Phân đa lớp với các quần thể có phân phối chuẩn

Gọi $f_i(x)$ hàm mật độ xác suất theo phân phối chuẩn của lớp π_i, với $i = 1,2,\dots,g$, có mean vector và covariance matrix là μ_i, Σ_i:

$$f_i(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_i|^{1/2}} exp\left[-\frac{1}{2}(x-\mu_i)'\Sigma_i^{-1}(x-\mu_i)\right] với\ i = 1,2,\dots,g$$

Giả sử chi phí phân lớp lỗi là như nhau, tức: $c(i|i) = 0$, $c(i,k) = 1\ với\ k \neq i$, ta thực hiện phân lớp x_0 vào π_k nếu:

$$p_k f_k(x_0) = \max_i p_i f_i(x_0)$$

Hay:

$$\ln\left(p_k f_k(x_0)\right) = \max_i \ln\left(p_i f_i(x_0)\right.$$

Mà:

$$\ln\left(p_k f_k(x_0)\right) = \ln(p_k) - \frac{p}{2}\ln(2\pi) - \frac{1}{2}\ln\left(|\Sigma_k|\right) - \frac{1}{2}(x-\mu_k)'\Sigma_k^{-1}(x-\mu_k)$$

Nhận thấy rằng $-\frac{p}{2}\ln(2\pi)$ là một hằng số. Vậy, ta gán x_0 vào lớp π_k nếu đại lượng discrimination score $\widehat{d_k}(\mathrm{x})$ đạt max với:

$$\widehat{d_k}(\mathrm{x}) = \ln(p_k) - \frac{1}{2}\ln(|\Sigma_k|) - \frac{1}{2}(x-\mu_k)'\Sigma_k^{-1}(x-\mu_k)$$

2.2.3.1 Trường hợp các lớp đều có Σ như nhau

discrimination score có dạng:

$$\widehat{d_k}(\mathrm{x}) = \ln(p_k) + \mathrm{x}\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^{-1}\Sigma^{-1}\mu_k$$

Để áp dụng vào bài toán phân lớp thực tế, ta cần xấp xỉ các giả trị quần thể trên bằng các giá trị của mẫu như sau:

- Xấp xỉ trung bình quần thể bằng trung bình mẫu

- Xấp xỉ $\Sigma = S_{pooled}$, với S_{pooled} là trung bình có trọng số các ma trận hiệp phương sai mẫu của các lớp:

$$S_{pooled} = \left[\frac{n_1 - 1}{(n_1 - 1) + (n_2 - 1) + \cdots + (n_g - 1)}\right] S_1 + \left[\frac{n_1 - 1}{(n_1 - 1) + (n_2 - 1) + \cdots + (n_g - 1)}\right] S_2 + \cdots + \left[\frac{n_1 - 1}{(n_1 - 1) + (n_2 - 1) + \cdots + (n_g - 1)}\right] S_g$$

Ví dụ 2.2.3.1

khảo sát 3 lớp ngẫu nhiên có phân phối chuẩn và có cùng covariance matrix, mỗi lớp có 3 mẫu như sau:

$$\boldsymbol{\pi_1}: \boldsymbol{X_1} = \begin{pmatrix} \mathbf{-2} & \mathbf{5} \\ \mathbf{0} & \mathbf{3} \\ \mathbf{-1} & \mathbf{1} \end{pmatrix}$$

$$\boldsymbol{\pi_2}: \boldsymbol{X_2} = \begin{pmatrix} \mathbf{0} & \mathbf{6} \\ \mathbf{2} & \mathbf{4} \\ \mathbf{1} & \mathbf{2} \end{pmatrix}$$

$$\boldsymbol{\pi_3}: \boldsymbol{X_3} = \begin{pmatrix} \mathbf{1} & \mathbf{-2} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-4} \end{pmatrix}$$

Biết rằng, xác suất của từng lớp là $\boldsymbol{p_1 = p_2 = 0.25, p_3 = 0.5}$, hãy phân loại $\boldsymbol{x_0} = [\mathbf{-2} \quad \mathbf{-1}]$ vào 1 trong các lớp trên

Giải

Trung bình mẫu và ma trận hiệp phương sai mẫu:

$$\overline{\boldsymbol{x_1}} = \begin{pmatrix} \mathbf{-1} \\ \mathbf{3} \end{pmatrix},$$

$$S_1 = \frac{1}{3-1}\left[\begin{pmatrix}-1\\2\end{pmatrix}(-1 \quad 2) + \begin{pmatrix}1\\0\end{pmatrix}(1 \quad 0) + \begin{pmatrix}0\\-2\end{pmatrix}(0 \quad -2)\right] = \begin{pmatrix}1 & -1\\-1 & 4\end{pmatrix}$$

$$\overline{x_2} = \begin{pmatrix}1\\4\end{pmatrix}, S_2 = \begin{pmatrix}1 & -1\\-1 & 4\end{pmatrix}$$

$$\overline{x_3} = \begin{pmatrix}0\\-2\end{pmatrix}, S_2 = \begin{pmatrix}1 & 1\\1 & 4\end{pmatrix}$$

Trung bình trọng số các ma trận hiệp phương sai:

$$S_{pooled} = \frac{3-1}{9-3}\left[\begin{pmatrix}1 & -1\\-1 & 4\end{pmatrix} + \begin{pmatrix}1 & -1\\-1 & 4\end{pmatrix} + \begin{pmatrix}1 & 1\\1 & 4\end{pmatrix}\right] = \begin{pmatrix}1 & -\frac{1}{3}\\-\frac{1}{3} & 4\end{pmatrix}$$

$$S_{pooled}^{-1} = \frac{1}{4-\frac{1}{9}}\begin{pmatrix}4 & \frac{1}{3}\\\frac{1}{3} & 1\end{pmatrix} = \frac{1}{35}\begin{pmatrix}36 & 3\\3 & 9\end{pmatrix}$$

Discrimination score:

$$\widehat{d_1}(x) = \ln(p_1) + \mathrm{x}' S_{pooled}^{-1}\overline{x_1} - \frac{1}{2}\overline{x_1}^{-1} S_{pooled}^{-1}\overline{x_1}$$

$$= \ln(0.25) + \frac{x'}{35}\begin{pmatrix}-27\\24\end{pmatrix} - \frac{1}{2}\begin{pmatrix}99\\35\end{pmatrix}$$

$$\widehat{d_2}(x) = \ln(0.25) + \frac{x'}{35}\begin{pmatrix}48\\39\end{pmatrix} - \frac{1}{2}\begin{pmatrix}204\\35\end{pmatrix}$$

$$\widehat{d_3}(x) = \ln(0.5) + \frac{x'}{35}\begin{pmatrix}-6\\-18\end{pmatrix} - \frac{1}{2}\begin{pmatrix}36\\35\end{pmatrix}$$

Thay x_0 vào các biểu thức:

$$\widehat{d_1}(x_0) = \ln(0.25) + \frac{(-2 \quad -1)}{35}\begin{pmatrix}-27\\24\end{pmatrix} - \frac{1}{2}\begin{pmatrix}99\\35\end{pmatrix} = -1.943$$

$$\widehat{d_2}(x_0) = \ln(0.25) + \frac{(-2 \quad -1)}{35}\begin{pmatrix}48\\39\end{pmatrix} - \frac{1}{2}\begin{pmatrix}204\\35\end{pmatrix} = -8.158$$

$$\widehat{d_3}(x_0) = \ln(0.5) + \frac{(-2 \quad -1)}{35}\begin{pmatrix} -6 \\ -18 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 36 \\ 35 \end{pmatrix} = -0.35$$

Thấy rằng discrimination score $\widehat{d_3}(x_0)$ có giá trị lớn nhất, vì vậy ta phân lớp $x_0 = [-2 \quad -1]$ vào lớp π_3

3 Linear Discriminant Analysis

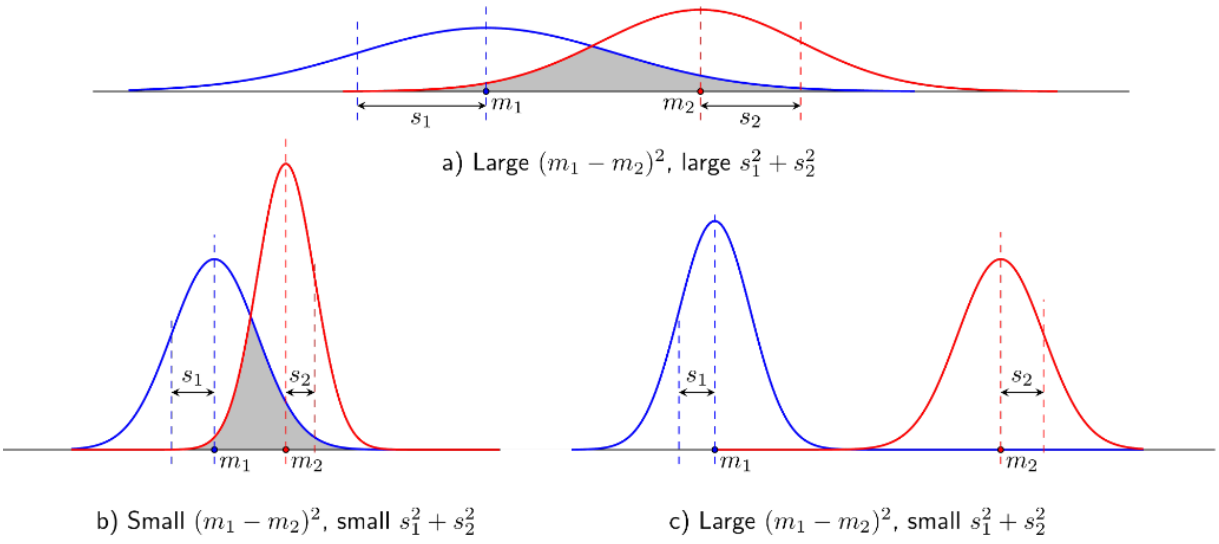
3.1 Giới thiệu

Linear Discriminant Analysis (LDA) là một phương pháp giảm chiều dữ liệu cho bài toán classification. LDA có thể được coi là một phương pháp giảm chiều dữ liệu (dimensionality reduction), và cũng có thể được coi là một phương pháp phân lớp (classification), và cũng có thể được áp dụng đồng thời cho cả hai, tức giảm chiều dữ liệu sao cho việc phân lớp hiệu quả nhất. Số chiều của dữ liệu mới là nhỏ hơn hoặc bằng $C - 1$ trong đó C là số lượng classes. Từ 'Discriminant' được hiểu là những thông tin đặc trưng cho mỗi class, khiến nó không bị lẫn với các classes khác. Từ 'Linear' được dùng vì cách giảm chiều dữ liệu được thực hiện bởi một ma trận chiếu (projection matrix), là một phép biến đổi tuyến tính (linear transform).

Giải bài toán LDA là đi tìm một phép chiếu sao cho tỉ lệ độ dị biệt giữa các lớp và độ dị biệt trong một lớp lớn nhất có thể.

3.2 Bài toán LDA cho 2 lớp

- Ý tưởng cơ bản



Hình 3.1

- Hình 3.1a: khoảng cách giữa hai kì vọng lớn nhưng phương sai trong mỗi class cũng lớn khiến 2 phân phối chồng lên nhau (phần màu xám), tức là dữ liệu chưa discriminant
- Hình 3.1b: độ lệch chuẩn của hai class đều nhỏ, tức dữ liệu tập trung hơn. Tuy nhiên, vấn đề với trường hợp này là khoảng cách giữa hai class, được đo bằng khoảng cách giữa hai kỳ vọng , là quá nhỏ, khiến cho phần chồng lấn cũng chiếm môt tỉ lệ lớn, và cũng không tốt cho classification.
- Hình 3.1c: phương sai đủ nhỏ và khoảng cách giữa hai kỳ vọng đủ lớn, ta thấy rằng dữ liệu discriminant hơn.

Độ lệch chuẩn nhỏ thể hiện việc dữ liệu ít phân tán. Điều này có nghĩa là dữ liệu trong mỗi class có xu hướng giống nhau. Hai phương sai s_1^2, s_2^2 còn được gọi là các $within - class\ variances$.

Khoảng cách giữa các kỳ vọng là lớn chứng tỏ rằng hai classes nằm xa nhau, tức dữ liệu giữa các classes là khác nhau nhiều. Bình phương khoảng cách giữa hai kỳ vọng $(m_1 - m_2)^2$ còn được gọi là $between - class\ variance$.

Hai classes được gọi là discriminative nếu hai class đó cách xa nhau (between-class variance lớn) và dữ liệu trong mỗi class có xu hướng giống nhau (within-class variance nhỏ). Linear Discriminant Analysis là thuật toán đi tìm một phép chiếu sao cho tỉ lệ giữa between-class variance và within-class variance lớn nhất có thể.

- Xây dựng hàm mục tiêu

Giả sử có N điểm dữ liệu $x_1, x_2, \dots, x_n$ trong đó $N_1 < N$ điểm đầu tiên thuộc class thứ nhất và $N - N_1$ điểm sau thuộc class thứ hai. Ký hiệu $C_1 = \{n | 1 \leq n \leq N_1\}$ là tập chỉ số các điểm thuộc class 1 và $C_2 = \{m | N_1 + 1 \leq m < N\}$ là tập các chỉ số thuộc class thứ 2. Phép chiếu dữ liệu xuống 1 đường thẳng có thể được mô tả bằng một vector hệ số w, giá trị tương ứng của mỗi điểm dữ liệu mới là:

$$y_n = w^T x_n, 1 \leq n \leq N$$

Vector kì vọng của mỗi class:

$$m_k = \frac{1}{N_k} \sum_{n \in C_k} x_n, \qquad k = 1, 2 \quad (1)$$

Khi đó khoảng cách giữa các kì vọng trong không gian mới là:

$$M_1 - M_2 = \frac{1}{N_1} \sum_{i \in C_1} y_i - \frac{1}{N_2} \sum_{i \in C_2} y_i = w^T (m_1 - m_2) \quad (2)$$

Các $within\ class\ variances$ được định nghĩa là:

$$s_k^2 = \sum_{n \in C_k} (y_n - M_k)^2, \ \ k = 1, 2 \quad (3)$$

LDA là thuật toán đi tìm giá trị lớn nhất của hàm mục tiêu:

$$J(w) = \frac{(M_1 - M_2)^2}{s_1^2 + s_2^2} \quad (4)$$

Tìm sự phụ thuộc giữa tử số và mẫu số của (4) vào w:

- Tử số

$$(M_1 - M_2)^2 = w^T(m_1 - m_2)(m_1 - m_2)^T w = w^T S_B w \quad (5)$$

$$với\ S_B = (m_1 - m_2)(m_1 - m_2)^T$$

S_B còn được gọi là $between - class\ covariance\ matrix$. Đây là một ma trận đối xứng nửa xác định dương.

- Mẫu số

$$s_1^2 + s_2^2 = \sum_{k=1}^{2} \sum_{n \in C_k} (w^T(x_n - m_k)^2 = w^T \sum_{k=1}^{2} \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T w$$

$$s_1^2 + s_2^2 = w^T S_w w \quad (6)$$

Với $S_w = \sum_{k=1}^{2} \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T$ được gọi là $within - class\ covariance\ matrix$. Đây cũng là một ma trận đối xứng nửa xác định dương vì nó là tổng của hai ma trận đối xứng nửa xác định dương.

Như vậy, bài toán tối ưu cho LDA trở thành:

$$w^* = argmax_w \frac{w^T S_B w}{w^T S_w w} \quad (7)$$

- Nghiệm của bài toán tối ưu

  Nghiệm của w trong (7) sẽ là nghiệm của phương trình đạo hàm của hàm mục tiêu J(w) bằng 0

$$\nabla_w J = \frac{1}{\left((w^T S_w w)\right)^2} (2S_B w(w^T S_w w) - 2w^T S_B w^T S_w w) = 0 \leftrightarrow S_B w = \frac{w^T S_B w}{w^T S_w w} S_w w$$

$$\rightarrow S_w^{-1} S_B w = J(w) w \quad (8)$$

Vì $J(w)$ là một số vô hướng, ta suy ra w là một vector riêng của $S_w^{-1} S_B$ ứng với một trị riêng nào đó và trị riêng này bằng J_w. Vậy để hàm mục tiêu đạt giá trị lớn nhất thì J_w là trị riêng lớn nhất của $S_w^{-1} S_B$

Nếu w là nghiệm thì kw cũng là nghiệm với k là một số thực khác 0 bất kỳ. Vậy ta có thể chọn w sao cho $(m_1 - m_2)^T w = J(w) = L = trị\ riêng\ lớn\ nhất\ của\ S_w^{-1} S_B$. Khi đó thay định nghĩa S_B ở (5) vào (8) ta có:

$$Lw = S_w^{-1}(m_1 - m_2)(m_1 - m_2)^T w = LS_w^{-1}(m_1 - m_2)$$

Điều này có nghĩa ta có thể chọn $w = \alpha S_w^{-1}(m_1 - m_2)$

3.3 Bài toán LDA cho nhiều lớp

- Bài toán đặt ra
  - Độ dị biệt giữa các thành phần trong class (within class):

    Within class variance của class k được tính $\sigma_k^2 = \sum_{n \in C_k} ||y_n - e_k||^2 = ||Y_k - E_k||^2$

    $$= ||W^T(X_k - M_k)||^2 = trace(W^T(X_k - M_k)(X_k - M_k)^T W) \quad (16)$$

    Với E_k là ma trận có các cột giống hệt nhau và giống vector kì vọng e_k

    Có thể thấy $E_k = W^T M_k$ với M_k là ma trận có các cột giống hệt nhau và bằng vector kì vọng m_k trong không gian ban đầu

    Tổng độ dị biệt trong các class được tính theo công thức:

$$s_w = \sum_{k=1}^{C} \sigma_k^2 = \sum_{k=1}^{C} trace(W^T(X_k - M_k)(X_k - M_k)^T W) = trace(W^T S_w W) \quad (18)$$

    Với $S_w = \sum_{k=1}^{C} \|X_k - M_k\|^2 = \sum_{k=1}^{C} \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T$ (19)

    Được gọi là within class covariance matrix

  - Độ dị biệt giữa các class (between class)

    Between-class lớn, có thể đạt được nếu tất cả các điểm trong không gian mới đều xa vector kỳ vọng chung e. Việc này cũng có thể đạt được nếu các vector kỳ vọng của mỗi class xa các vector kỳ vọng chung (trong không gian mới).

$$s_b = \sum_{k=1}^{C} N_k \|e_k - e\|^2 = \sum_{k=1}^{C} \|E_k - E\|^2$$

$$Với\ N_k\ là\ trọng\ số\ vì\ mỗi\ class\ có\ số\ phần\ tử\ khác\ nhau$$

$$s_b = \sum_{k=1}^{c} trace(W^T(M_k - M)(M_k - M)^T W) = trace(W^T S_B W)$$

$$Với\ S_B = \sum_{k=1}^{C}(M_k - M)(M_k - M)^T = \sum_{k=1}^{C} N_k(m_k - m)(m_k - m)^T$$

- Với cách định nghĩa và ý tưởng về within-class nhỏ và between-class lớn như trên, ta có thể xây dựng bài toán tối ưu như sau:

$$f(W) = \frac{trace(W^T S_B W)}{trace(W^T S_W W)}$$

Tìm W sao cho f(W) đạt cực đại

- Nghiệm của bài toán

Để hàm $f(W)$ đạt cực đại thì đạo hàm của nó phải bằng 0

$$\nabla f(W) = 0 \Leftrightarrow \frac{2(S_W W * trace(W^T S_B W) - S_B W * trace(W_T S_W W))}{trace(W^T S_W W)^2}$$

$$\Leftrightarrow S_W W * trace(W^T S_B W) - S_B W * trace(W_T S_W W) = 0 \Leftrightarrow S_w^{-1} S_B W = J(W) W$$

Suy ra các cột của ma trận W là các vector riêng độc lập tuyến tính ứng với trị riêng lớn nhất của $S_w^{-1} S_B$

3.4 So sánh LDA với PCA

PCA	LDA
Giảm số chiều của dữ liệu, chuyển dữ liệu từ không gian nhiều chiều sang không gian ít chiều hơn	
Xây dựng không gian mới với số chiều nhỏ hơn ban đầu sao cho giữ lại độ dị biệt lớn nhất có thể	Xây dựng không gian mới với số chiều nhỏ hơn ban đầu sao cho tỉ lệ độ dị biệt giữa các class và độ dị biệt của các thành phần trong từng class là lớn nhất

4 Ứng dụng

4.1 Cài đặt môi trường

Các thuật toán phân lớp thống kê được nhóm cài đặt bằng ngôn ngữ Python 3 trên môi trường Jupiter Notebook. Các thư viện Python mà nhóm sử dụng cùng với cách cài đặt:

- Thư viện numpy để làm việc với ma trận và sử dụng các hàm toán học. Sử dụng pip để cài đặt

$$pip\ install\ numpy$$

- Thư viện sklearn để lấy dữ liệu và chia dữ liệu ra thành tập train và tập test

$$pip\ install\ -U\ scikit-learn$$

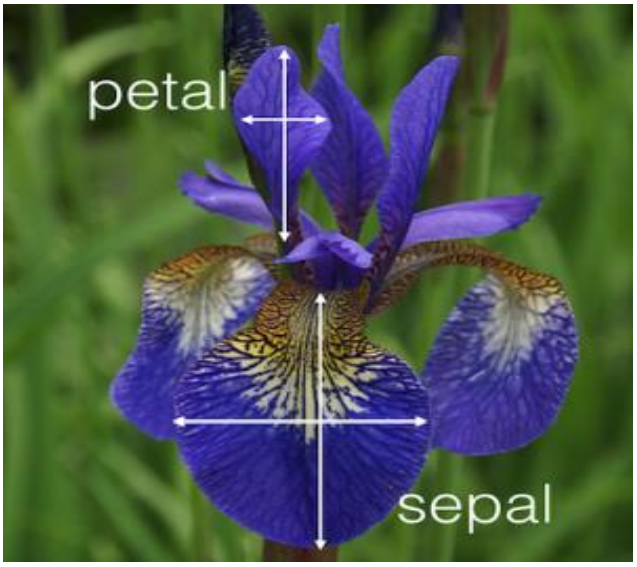
- Thư viện matplotlib để trực quan hóa dữ liệu

$$python\ -m\ pip\ install\ -U\ matplotlib$$

4.2 Phân lớp sử dụng LDA

4.2.1 LDA cho tập dữ liệu hoa Iris

- Tập dữ liệu Iris.csv:
  - 50 mẫu của 3 loài Iris (setosa, virginica, versicolor)
  - 4 biến: chiều dài, chiều rộng đài hoa, cánh hoa



- Chương trình
  - Import các thư viện và các hàm cần thiết

```
# dữ liệu sử dụng cho ví dụ
from sklearn import datasets                          # lấy dữ liệu
from sklearn.model_selection import train_test_split  # chia dữ liệu ra thành tập train và tập test
import numpy as np                                    # thư viện để làm việc với mảng và các hàm toán học
from scipy.spatial import distance                    # tính khoảng cách eclipse
import matplotlib.pyplot as plt                       # trực quan hóa dữ liệu
```

  - Chia data thành tập train và tập test

```
# chia dữ liệu thành tập train và tập test
# load dữ liệu
data = datasets.load_iris()
X, y = data.data, data.target
# với tập train là 75% dữ liệu
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state =0 )

# biến toàn cục
X_train_projected = None
X_test_projected = None
X_train
```

Một số dòng dữ liệu trong tập dữ liệu

```
array([[5.9, 3. , 4.2, 1.5],
       [5.8, 2.6, 4. , 1.2],
       [6.8, 3. , 5.5, 2.1],
       [4.7, 3.2, 1.3, 0.2],
       [6.9, 3.1, 5.1, 2.3]])
```

- Tạo class LDA

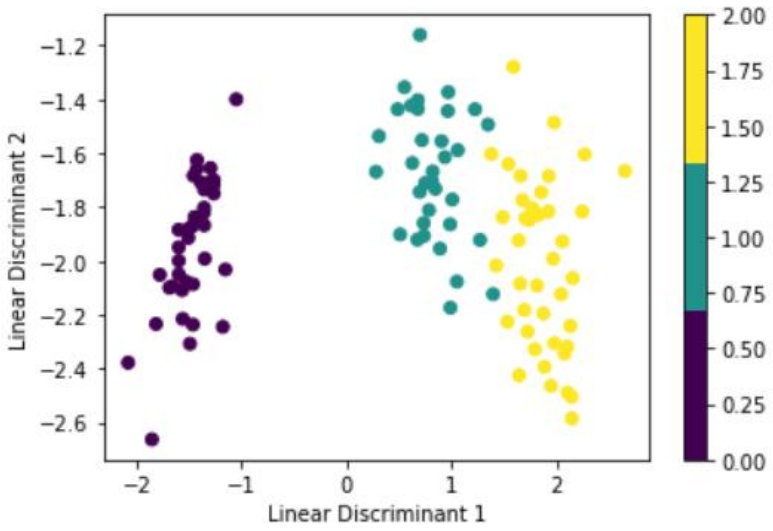
```python
class LDA:
    def __init__(self, n_components):
        # số chiều dữ liệu còn lại trong không gian mới
        self.n_components = n_components
        self.linear_discriminants = None

    def fit(self, X, y):
        n_features = X.shape[1]
        class_labels = np.unique(y)   # liệt kê các lớp trong tập dữ liệu
        # Within class scatter matrix:
        # SW = sum((X_c - mean_X_c)^2 )
        # Between class scatter:
        # SB = sum( n_c * (mean_X_c - mean_overall)^2 )
        m = np.mean(X, axis=0)
        SW = np.zeros((n_features, n_features))       #  within-class covariance matrix
        SB = np.zeros((n_features, n_features))       #  between-class covariance matrix
        for k in class_labels:
            X_k = X[y == k]                           # Lấy các mẫu thuộc cùng class
            m_k = np.mean(X_k, axis=0)                # Trung bình các mẫu thuộc class
            # (4, N_k) * (N_k, 4) = (4,4)
            SW += (X_k - m_k).T.dot((X_k - m_k))
            # (4, 1) * (1, 4) = (4,4) -> reshape
            N_k = X_k.shape[0]
            mean_diff = (m_k - m).reshape(n_features, 1)
            SB += N_k * (mean_diff).dot(mean_diff.T)
        # Tính SW^-1 * SB
        A = np.linalg.inv(SW).dot(SB)
        # Tính trị riêng và vector riêng của SW^-1 * SB
        eigenvalues, eigenvectors = np.linalg.eig(A)
        # sắp xếp trị riêng từ lớn đến bé
        eigenvectors = eigenvectors.T
        idxs = np.argsort(abs(eigenvalues))[::-1]
        eigenvalues = eigenvalues[idxs]
        eigenvectors = eigenvectors[idxs]
        # giữ lại n_components vector ứng với các giá trị riêng lớn nhất
        self.linear_discriminants = eigenvectors[0 : self.n_components]
    def transform(self, X):
        # chiếu dữ liệu sang không gian mới
        return np.dot(X, self.linear_discriminants.T)
```

- Giảm chiều dữ liệu từ 4 chiều xuống 2 chiều

```python
# Testing
if __name__ == "__main__":

    # Chiếu dữ liệu trong không gian 4 chiều ban đầu sang không gian 2 chiều
    lda = LDA(2)
    lda.fit(X_train, y_train)
    X_train_projected = lda.transform(X_train)

    print("Shape of X:", X_train.shape)
    print("Shape of y:", y_train.shape)
    print("Shape of transformed X:", X_train_projected.shape)

    x1, x2 = X_train_projected[:, 0], X_train_projected[:, 1]
    plt.scatter(x1, x2, c=y_train, alpha=1, cmap=plt.cm.get_cmap("viridis", 3))

    plt.xlabel("Linear Discriminant 1")
    plt.ylabel("Linear Discriminant 2")
    plt.colorbar()
    plt.show()
```

Dữ liệu trong không gian mới

```
Shape of X: (112, 4)
Shape of y: (112,)
Shape of transformed X: (112, 2)
```



Chiếu tập dữ liệu test và gán nhãn dự đoán cho tập dữ liệu test bằng cách tính độ đo Euclid

```python
# Chiếu tập dữ liệu test vào không gian mới
X_test_projected = lda.transform(X_test)
```

```python
# Tính trung bình của mỗi class trong không gian mới
means = np.array([None,None,None])
class_labels = np.unique(y_train)
for c in class_labels:
    X_c = X_train_projected[y_train == c]
    means[c] = np.mean(X_c, axis=0)
```

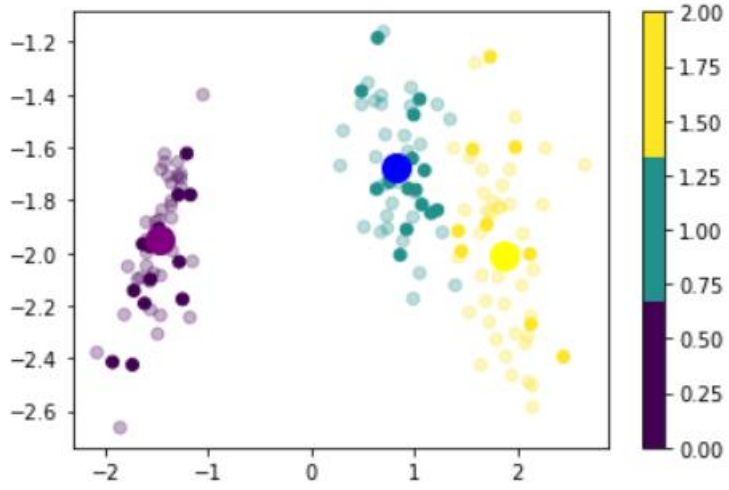
```python
# tính khoảng cách eclipse và và dự đoán class dựa trên khoảng cách nhỏ nhất đến các điểm trung tâm mỗi class
y_test_predict = np.zeros(y_test.shape, dtype = 'i1')
for i in range(X_test_projected.shape[0]):
    dis = [distance.euclidean(means[j], X_test_projected[i]) for j in range(3)]
    y_test_predict[i] = np.argmin(dis)
```

```python
# vẽ dữ liệu trong không gian mới (2 chiều) với các điểm trung tâm
x_test1, x_test2 = X_test_projected[:, 0], X_test_projected[:, 1]
x1, x2 = X_train_projected[:, 0], X_train_projected[:, 1]

plt.scatter(x1, x2, c=y_train, alpha=0.3, cmap=plt.cm.get_cmap("viridis", 3))

plt.scatter(x_test1, x_test2, c=y_test_predict, alpha=1, cmap=plt.cm.get_cmap("viridis", 3))
plt.colorbar()

plt.scatter(means[0][0],means[0][1],200, c = 'purple')
plt.scatter(means[1][0],means[1][1],200, c = 'blue')
plt.scatter(means[2][0],means[2][1],200, c = 'yellow')

```

```
<matplotlib.collections.PathCollection at 0x1fee9c2d430>
```

Dữ liệu trong không gian mới và các điểm trung tâm của các class


```
<matplotlib.collections.PathCollection at 0x1fee9c2d430>
```

Độ chính xác:

```python
# tính độ chính xác
s = np.sum(y_test_predict==y_test)
print("accuracy = ", s * 100/y_test.size, "%")
```

```
accuracy =  97.36842105263158 %
```

- Giảm chiều dữ liệu từ 4 chiều xuống 3 chiều

Giảm chiều dữ liệu còn 3

```python
# Giảm chiều dữ liệu từ 4 chiều xuống 3 chiều
lda = LDA(3)
lda.fit(X_train, y_train)
X_train_projected = lda.transform(X_train)
x_test_predict = lda.transform(X_test)
```

```python
# Tìm điểm trung tâm của mỗi class
means1 = np.array([None,None,None])
class_labels = np.unique(y_train)
for c in class_labels:
    X_c = X_train_projected[y_train == c]
    means1[c] = np.mean(X_c, axis=0)
```

```python
# Dự đoán nhãn cho dữ liệu test
for i in range(X_test.shape[0]):
    dis = [distance.euclidean(means1[j], x_test_predict[i]) for j in range(3)]
    y_test_predict[i] = np.argmin(dis)
```

Độ chính xác

```python
# tính độ chính xác
s = np.sum(y_test_predict==y_test)
print("accuracy = ", s * 100/y_test.size, "%")
```

```
accuracy =  97.36842105263158 %
```

- Để nguyên số chiều để phân lớp

Để nguyên số chiều dữ liệu là 4

```
# Tìm điểm trung tâm của mỗi class
means1 = np.array([None,None,None])
class_labels = np.unique(y_train)
for c in class_labels:
    X_c = X_train[y_train == c]
    means1[c] = np.mean(X_c, axis=0)
```

```
# Dự đoán nhãn cho dữ liệu test
for i in range(X_test.shape[0]):
    dis = [distance.euclidean(means1[j], X_test[i]) for j in range(3)]
    y_test_predict[i] = np.argmin(dis)
```

Độ chính xác

```
# tính độ chính xác
s = np.sum(y_test_predict==y_test)
print("accuracy = ", s * 100/y_test.size, "%")
```

```
accuracy =  89.47368421052632 %
```

- Giảm số chiều xuống 1 chiều

Giảm xuống 1 chiều

```
# Giảm chiều dữ liệu từ 4 chiều xuống 1 chiều
lda = LDA(3)
lda.fit(X_train, y_train)
X_train_projected = lda.transform(X_train)
x_test_predict = lda.transform(X_test)
```

```
# Tìm điểm trung tâm của mỗi class
means1 = np.array([None,None,None])
class_labels = np.unique(y_train)
for c in class_labels:
    X_c = X_train_projected[y_train == c]
    means1[c] = np.mean(X_c, axis=0)
```

```
# Dự đoán nhãn cho dữ liệu test
for i in range(X_test.shape[0]):
    dis = [distance.euclidean(means1[j], x_test_predict[i]) for j in range(3)]
    y_test_predict[i] = np.argmin(dis)
```

Độ chính xác

```
# tính độ chính xác
s = np.sum(y_test_predict==y_test)
print("accuracy = ", s * 100/y_test.size, "%")
```

```
accuracy =  97.36842105263158 %
```

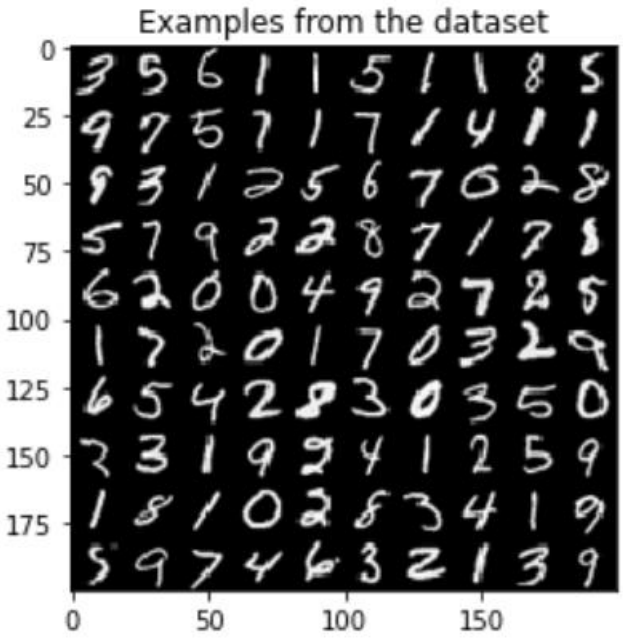
- So sánh độ chính xác:

Giữ nguyên số chiều	Giảm còn 3 chiều	Giảm còn 2 chiều	Giảm còn 1 chiều
89.4%	97.3%	97.3%	97.3%

➔ Từ bảng so sánh trên, ta thấy đối với tập dữ liệu Iris, ta có thể giảm dữ liệu xuống còn 1 chiều để phân lớp và vẫn đảm bảo phân lớp với độ chính xác ở mức cao là 97.3%. Khi giữ nguyên số chiều của dữ liệu, các điểm dữ liệu không "Discriminant" nên việc phân lớp bằng Euclid không mang lại hiệu quả cao khi không giảm chiều.

4.2.2 LDA cho tập dữ liệu chữ số viết tay

- Tập dữ liệu gồm có 5000 mẫu chia đều cho 10 lớp có nhãn là các số từ 0 đến 9. Với mỗi mẫu là một ảnh chữ viết tay có độ phân giải là 20x20, được lưu trữ dưới dạng ma trận (5000x400)
- Ví dụ 100 mẫu ngẫu nhiên được trực quan hóa



- Chương trình
  - Import các thư viện và hàm cần thiết

Sử dụng LDA phân lớp và so sánh với sử dụng softmax để phân lớp

]:
```
import numpy as np
import pandas as pd
import matplotlib.pylab as plt
from scipy.optimize import minimize
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
import time
%matplotlib inline
```

- Tải dữ liệu từ file matlab

:
```
# Tải dữ liệu lên từ file matlab
import scipy.io as sio

data = sio.loadmat("data.mat")

# giá trị điểm ảnh của các chữ số viết tay

# Gồm 5000 ảnh, mỗi ảnh có độ phân giải 20x20
X = data["X"]

# nhãn
y = data["y"]

y = y.reshape(len(y))
for j in range(len(y)):
    if y[j] == 10:
        y[j] = 0
m, n = X.shape

m,n
```

- Chia dữ liệu: với mỗi lớp, lấy 400 mẫu để train và 100 mẫu để test

```
# y(i,j) nhãn của ảnh thứ j lớp thứ i
_y = y.reshape(10,500)
# X(i,j,k) data của ảnh thứ j lớp thứ i
_X = X.reshape(10,500,400)
```

```
# mỗi lớp có 500 ảnh, lấy 400 ảnh train, 100 ảnh test
X_train = _X[:,:400,:]
X_test = _X[:,400:,:]
y_train = _y[:,:400]
y_test = _y[:,400:]
X_train = X_train.reshape(4000,400)
X_test = X_test.reshape(1000,400)
y_train = y_train.reshape(4000)
y_test = y_test.reshape(1000)


```

- Thêm bias – sử dụng cho hàm softmax

:
```python
# Thêm một cột bias cho dữ liệu đầu vào X_train
bias = np.ones((4000,1))
X_train_bias = np.append(bias, X_train, axis=1)
# Thêm một cột bias cho dữ liệu đầu vào X_test
bias = np.ones((1000,1))
X_test_bias = np.append(bias, X_test, axis=1)
```

- Sử dụng softmax để phân lớp

```python
# hàm tính softmax
def softmax(Z):
    e_Z = np.exp(Z)
    return e_Z / np.sum(e_Z)

# hàm tính cost
def compute_cost(theta, X, y):
    m = len(y)
    z = X@theta.T
    y_pre = np.zeros(y.shape)
    for i in range(m):
        y_pre[i] = softmax(z[i])
    cost = -np.sum(y*np.log(y_pre))
    return cost

# hàm tính gradient
def compute_gradient(theta, X, y):
    m = len(y)
    n = len(theta)
    z = X@theta.T
    y_pre = np.zeros(y.shape)
    for i in range(m):
        y_pre[i] = softmax(z[i])
    gradient = X.T @ (y_pre - y)
    return gradient
```

```python
# chuyển các nhãn thành các vector one-hot
def mapper(y,K):
    Y = np.zeros((len(y),K))
    for i in range(len(y)):
        for j in range(0,K):
            if (y[i]==j):
                Y[i][j] = 1
            else:
                Y[i][j] = 0
    return Y
```

Hàm train và train mô hình

```python
# Train mô hình softmax
def train(X, y, K,epoch, lr):
    n = X.shape[1]
    tmp = np.zeros(len(y))
    theta = np.zeros((K, n))
    Y = mapper(y,K)
    for i in range(epoch):
        cost = compute_cost(theta,X,Y)
        theta_grad = compute_gradient(theta,X,Y)
        theta = theta - lr*theta_grad.T
        if ((i+1)%50==0):
            print("loss in epoch " + str(i+1) + " = " + str(cost))
    return theta
```

```python
# Train với epoch = 950 và learningrate = 0.01
K = 10
start_time = time.time()
theta = train(X_train_bias, y_train, K,950,0.01)
print("--- %s seconds ---" % (time.time() - start_time))
```

Quá trình train

```
loss in epoch 50 = 16427.80749765432
loss in epoch 100 = 4028.802031710714
loss in epoch 150 = 7338.889454022058
loss in epoch 200 = 3180.829008443657
loss in epoch 250 = 2006.8159389732052
loss in epoch 300 = 3177.580131412338
loss in epoch 350 = 16820.569508463614
loss in epoch 400 = 1636.8071040939421
loss in epoch 450 = 1824.0600391708108
loss in epoch 500 = 1263.1461314460162
loss in epoch 550 = 1160.1496725661702
loss in epoch 600 = 1137.0418172211455
loss in epoch 650 = 1393.835758301134
loss in epoch 700 = 1604.9621683887224
loss in epoch 750 = 626.9833418355233
loss in epoch 800 = 1147.5927295361034
loss in epoch 850 = 827.5313331228652
loss in epoch 900 = 1396.807887670932
loss in epoch 950 = 534.1831686865638
--- 238.79628252983093 seconds ---
```

Độ chính xác

```
z = X_test_bias@theta.T
y_pre = np.zeros((len(y_test),10))
for i in range(1000):
    y_pre[i] = softmax(z[i])
accuracy = np.mean(np.argmax(y_pre, axis = 1) == y_test) * 100
'Training set accuracy using multi-class classification: {:2}%'.format(accuracy)
```

```
'Training set accuracy using multi-class classification: 87.6%'
```

- Sử dụng LDA để phân lớp

Giảm chiều dữ liệu còn 10

2]:
```
#Giảm số chiều từ 400 xuống min(n_classes - 1, n_features) = 10
start_time = time.time()
lda_10 = LDA()
lda_10.fit(X,y)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
--- 1.2075605392456055 seconds ---
```

Độ chính xác

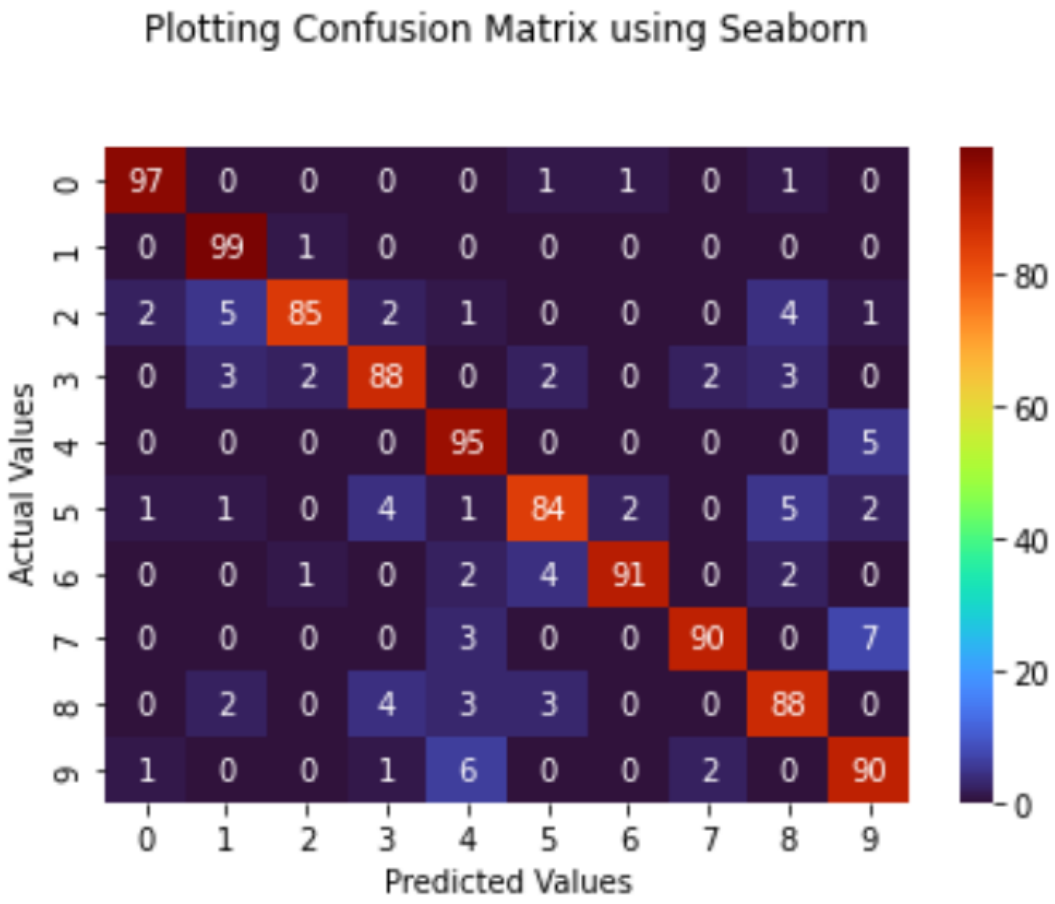
:
```
# dự đoán nhãn cho tập test
y_pre = lda_10.predict(X_test)
```

:
```
# kiểm tra độ chính xác của mô hình
accuracy = np.mean(y_pre == y_test) * 100
'Training set accuracy use LDA for classification: {:2}%'.format(accuracy)
```

:
```
'Training set accuracy use LDA for classification: 90.7%'
```

Vẽ confusion matrix

```
from sklearn import metrics
import seaborn as sebrn
confusion_matrix = metrics.confusion_matrix(y_test, y_pre)
# Using Seaborn heatmap to create the plot
fx = sebrn.heatmap(confusion_matrix, annot=True, cmap='turbo')

# labels the title and x, y axis of plot
fx.set_title('Plotting Confusion Matrix using Seaborn\n\n');
fx.set_xlabel('Predicted Values')
fx.set_ylabel('Actual Values ');

plt.show()
```



Từ confusion matrix trên, ta thấy sử dụng LDA để phân lớp tập dữ liệu chữ số viết tay đưa ra kết quả rất tốt đối với tất cả các lớp, thấm chí nhãn số 1 có độ chính xác lên đến 99%

- So sánh softmax và LDA đối với bộ dữ liệu chữ viết tay

	Thời gian chạy (giây)	Độ chính xác (%)
LDA	1.2	87.6
Softmax	237.8	90.7

Từ bảng trên, ta có thể thấy đối với bộ dữ liệu chữ viết tay, LDA đem lại hiệu quả vượt trội cả về thời gian và độ chính xác so với softmax.

4.3 Phân lớp sử dụng ECM

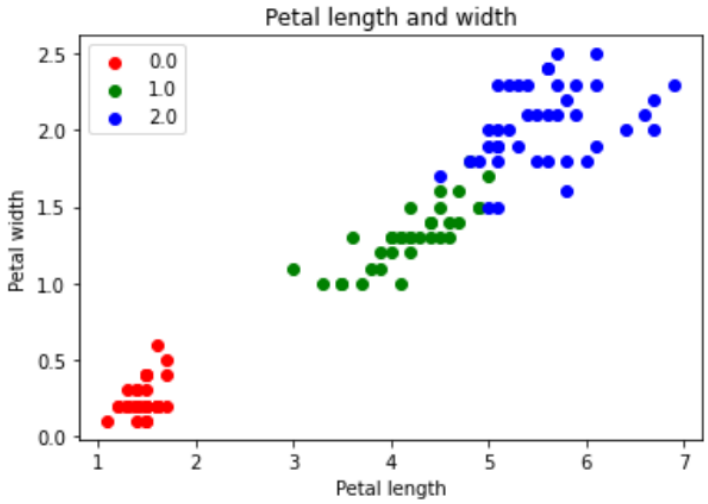
4.3.1 Phân lớp tập dữ liệu hoa Iris bằng cực tiểu ECM

Sử dụng tập dữ liệu hoa Iris, tập dữ liệu gồm có

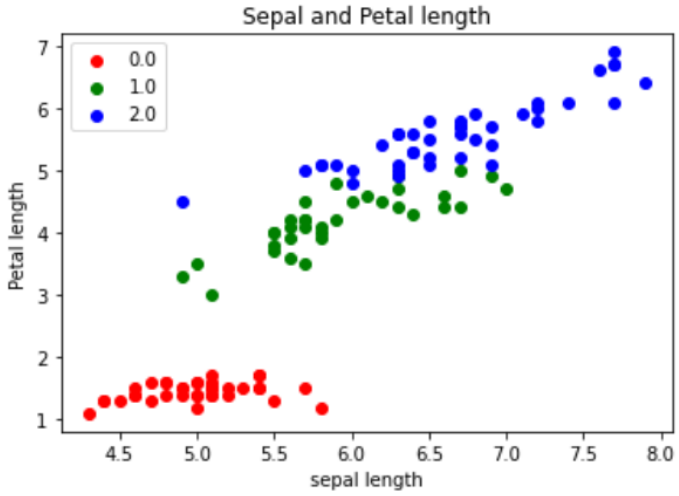
- 50 mẫu của 3 loài Iris (setosa, virginica, versicolor) .
- Mỗi mẫu gồm có 4 biến: chiều dài, chiều rộng đài hoa, cánh hoa.

Một số scatter plot của dữ liệu:

- Tương quan giữa petal length và petal width:



- Tương quan giữa sepal length và petal length:



Để thực hiện phân loại bằng cực tiểu ECM, ta giả định:

– Dữ liệu quần thể các hoa Iris có phân phối chuẩn.
– Chi phí phân loại sai bằng nhau với mọi trường hợp phân loại sai.

Việc phân lớp mẫu x_0 vào quần thể π_k đạt hiểu qua nhất khi:

$$p_k f_k(x_0) > p_i f_i(x_0), \text{ với } i = 1, 2, \ldots, g \text{ và } i \neq k$$

Hàm phân lớp:

$$class(x_0) = \underset{k}{\operatorname{argmax}}\, p_k f_k(x_0) \text{với k} \in \{1, 2, \ldots, g\}$$

$$\Rightarrow class(x_0) = \underset{k}{argmax}\, \widehat{d_k}(x_0)$$

Với đại lượng discrimination score theo lý thuyết:

$$\widehat{d_k}(\mathrm{x}_0) = \ln(p_k) - \frac{1}{2}\ln(|\mathrm{S}_k|) - \frac{1}{2}(x_0 - \bar{x}_k)' S_k^{-1}(x_0 - \bar{x}_k)$$

Cài đặt:

`import numpy as np`
`import matplotlib.pyplot as plt`
`from sklearn import datasets`
`from sklearn.model_selection import train_test_split`
`data = datasets.load_iris()`
`X, y = data.data, data.target`
`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state =1 )`
	Thư viện numpy phục vụ tính toán số và ma trận
	Thư Viện matplotlib để vẽ đồ thị
	Thư viện sklearn để lấy dataset và hàm chia dữ liệu
	Dữ liệu dược chia ngẫu nhiên thành 112 mẫu train để xây dựng hàm phân lớp và 38 mẫu test để kiểm tra kết quả hàm phân lớp

train_X = np.concatenate((X_train, y_train.reshape((y_train.shape[0],1))),axis=1) X_0 = train_X[train_X[:,4] == 0] X_1 = train_X[train_X[:,4] == 1] X_2 = train_X[train_X[:,4] == 2] train_X = [X_0, X_1, X_2] p = [] for i in range(len(train_X)): p.append(len(train_X[i])/ len(train))	Sắp xếp lại theo thứ tự: X_0: hoa setosa X_1: hoa versicolor X_2: hoa virginica Tính xác suất tiên nghiệm bằng cách lấy tổng số mẫu của từng loại hoa chia cho tổng số mẫu hoa. $p_1 + p_2 + p_3 = 1$

Mean = [] for i in range(len(train_X)): Mean.append(np.mean(train_X[i][:,:-1], axis=0)) Cov = [] for i in range(len(train_X)): Cov.append(np.cov(train_X[i][:,:-1].T))	Tính sample mean của từng loại họa Tính covariance của từng loại hoa

Tính discrimination score:

```
def discriminant_score(x, prior, Mean, Cov):
  score = np.log(prior) - (1.0/2) * np.log(np.linalg.det(Cov))
```

```
score -= (1.0/2) * np.dot(np.dot((x - Mean).T, np.linalg.inv(Cov)),(x-Mean))
return score
```

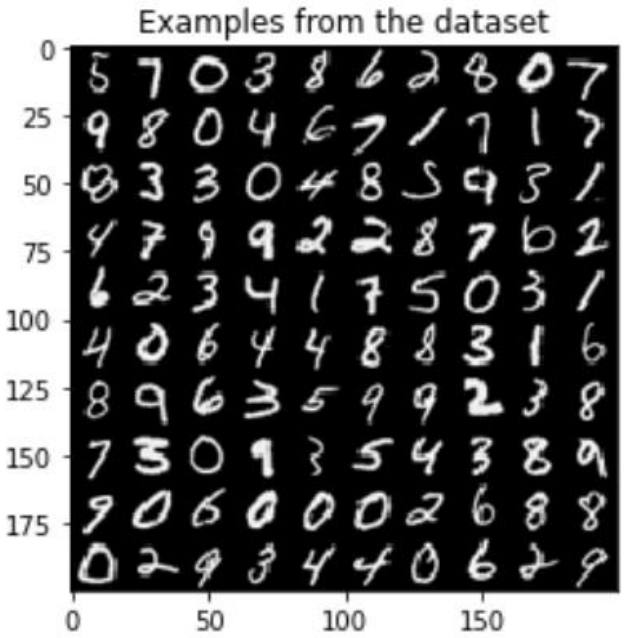
Sau đó, khi có dữ liệu mới chưa có nhãn, ta thực hiện gán nhãn dữ liệu đó là nhãn có đại lượng discrimination score lớn nhất.

Thực hiện tính toán các đại lượng thống kê bằng 75% mẫu (112 mẫu) và đánh giá phương pháp bằng 25% mẫu còn lại (38 mẫu). Do dữ liệu khá đơn giản và tách bạch giữa các lớp, ta thu được độ chính xác là $\frac{37}{38} \approx \mathbf{0.9737}$.

4.3.2 Phân lớp tập dữ liệu chữ số viết tay bằng cực tiểu ECM

Bộ dữ liệu sử dụng gồm 5000 ảnh chữ số viết tay từ 0 đến 9, mỗi chữ số có 500 ảnh.

Một số hình ảnh trong bộ dữ liệu:



Dữ liệu sẽ được sử dụng như sau:

- 4000 ảnh sẽ được sử dụng để tính toán các đại lượng thống kê (mỗi chữ số sử dụng 400 ảnh)
- 1000 ảnh sẽ được sử dụng để đánh giá phương pháp (mỗi chữ số sử dụng 100 ảnh)

Cách cài đặt tương tự như phần phân lớp hoa Iris, tuy nhiên, ở bước tính discriminant score, ta sẽ tính toán trước đại lượng liên quan đến ma trận covariance của 10 lớp để giảm thiểu thời gian thực hiện tính toán (mỗi ảnh gồm 400 pixel, chuyển thành dữ liệu 400 chiều nên việc tính covariance mất nhiều thời gian).

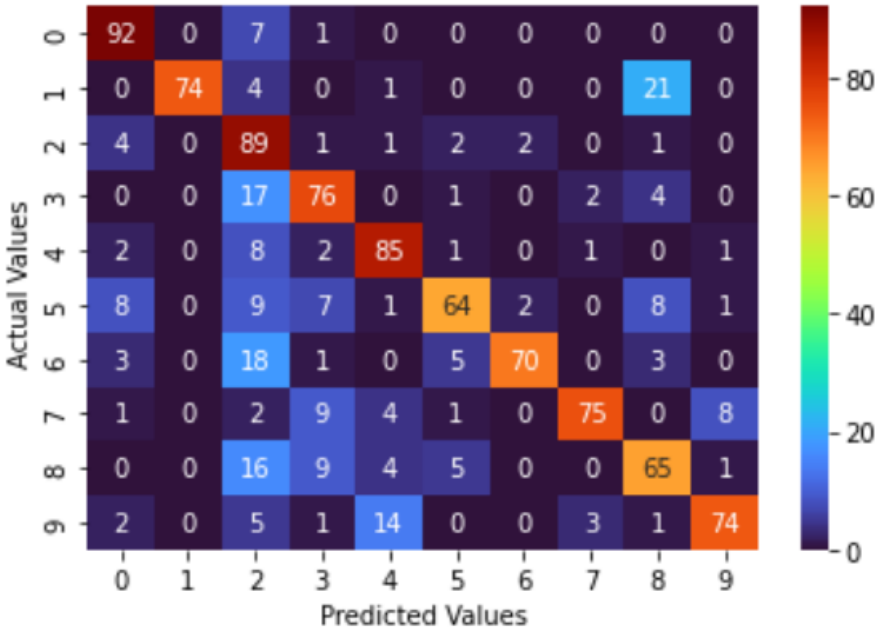
Discriminant score:

$$\widehat{d_k}(\mathrm{x}_0) = \ln(p_k) - \frac{1}{2}\ln(|\mathrm{S}_k|) - \frac{1}{2}(x_0 - \bar{x}_k)' S_k^{-1}(x_0 - \bar{x}_k)$$

```
def discriminant_score(x, Mean, log_prior_log_det_Cov, pinv_Cov):
    score = log_prior_log_det_Cov
    score -= (1.0/2) * np.dot(np.dot((x - Mean).T, pinv_Cov),(x-Mean))
    return score
```

Kết quả đánh giá trên tập test: 76.4%

Confusion matrix:



Các dữ liệu số 0, 2, 4 được dự đoán chính xác cao. Ngược lại, số 5 được dự đoán chính xác thấp nhất.

Số 9 bị dự đoán thành số 4 khá nhiều do sự tương đồng cao giữa 2 chữ số này.

Rất nhiều mẫu bị phân loại sai vào lớp 2.