**Code:**

clear

% Tạo một ma trận Features giả lập với numFeat đặc trưng

numFeat = 3; % số lượng đặc trưng

Features = rand(3, 5); % tạo ma trận 3x5 với các giá trị ngẫu nhiên trong khoảng [0, 1]

% Dữ liệu mẫu:

% Features = [0.1, 0.4, 0.7, 0.8, 0.2;

% 0.2, 0.5, 0.3, 0.9, 0.6;

% 0.8, 0.2, 0.9, 0.5, 0.1];

% Giá trị mong muốn cho Min và Max

MinVal = 0; % giá trị tối thiểu mong muốn

MaxVal = 1; % giá trị tối đa mong muốn

% Chuẩn hóa đặc trưng

for i = 1:numFeat

theMin = min(Features(i, :)); % giá trị nhỏ nhất trong đặc trưng thứ i

theMax = max(Features(i, :)); % giá trị lớn nhất trong đặc trưng thứ i

% Chuẩn hóa giá trị của đặc trưng thứ i về khoảng [MinVal, MaxVal]

NormalizedFeatures(i, :) = MinVal + ((MaxVal - MinVal) \* (Features(i, :) - theMin)) / (theMax - theMin);

end

% Hiển thị kết quả chuẩn hóa

disp('Normalized Features:');

disp(NormalizedFeatures);

**Mô tả:**

**Khởi tạo giá trị mong muốn**:

MinVal = 0; và MaxVal = 1; là các giá trị giới hạn cho phép chuẩn hóa đặc trưng về khoảng từ 0 đến 1.

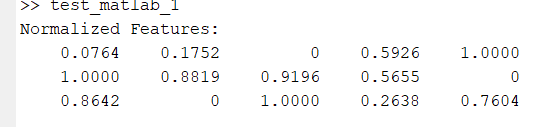
**Chuẩn hóa từng đặc trưng**:

Mỗi đặc trưng (mỗi dòng trong ma trận Features) được chuẩn hóa sao cho các giá trị trong mỗi đặc trưng nằm trong khoảng [MinVal, MaxVal].

**Công thức chuẩn hóa**:

Đoạn mã sử dụng công thức chuẩn hóa tuyến tính cho từng đặc trưng.

**Kết quả:**

****

**Code:**

clear

% Tạo một ma trận Features giả lập với numFeat đặc trưng

numFeat = 3; % số lượng đặc trưng

Features = rand(3, 5) \* 10; % tạo ma trận 3x5 với các giá trị ngẫu nhiên trong khoảng [0, 10]

% Dữ liệu mẫu:

% Features = [1, 4, 7, 8, 2;

% 2, 5, 3, 9, 6;

% 8, 2, 9, 5, 1];

% Bước 1: Nhân với hệ số

r = 0.5; % Hệ số r

Features = (1.0 / r) \* Features; % Nhân tất cả các giá trị trong Features với (1/r)

% Bước 2: Áp dụng hàm sigmoid

NormalizedFeatures = 1.0 ./ (1.0 + exp(-Features)); % Áp dụng hàm sigmoid để chuẩn hóa giá trị

% Hiển thị kết quả

disp('Normalized Features:');

disp(NormalizedFeatures);

**Mô tả:**

r = 0.5;: Hệ số r được đặt bằng 0.5.

Features = (1.0 / r) \* Features;: Tất cả các giá trị trong ma trận Features sẽ được nhân với 1.0/r. Điều này sẽ làm thay đổi phạm vi của các giá trị trong Features.

NormalizedFeatures = 1.0 ./ (1.0 + exp(-Features));: Đây là công thức của hàm sigmoid, có tác dụng chuyển đổi các giá trị của Features về một khoảng [0, 1].

**Kết quả:**

**A number and numbers on a white background

Description automatically generated**

**Code:**

clear

% Tạo dữ liệu mẫu x và y

x = randn(1, 100); % tạo 100 giá trị ngẫu nhiên từ phân phối chuẩn chuẩn hóa

y = randn(1, 100) + 1; % tạo 100 giá trị ngẫu nhiên từ phân phối chuẩn chuẩn hóa với trung bình = 1

% Kiểm định t hai mẫu

h\_ttest2 = ttest2(x, y);

disp(['ttest2 result: ', num2str(h\_ttest2)]);

% Kiểm định Kolmogorov-Smirnov một mẫu

h\_kstest = kstest(x);

disp(['kstest result: ', num2str(h\_kstest)]);

% Kiểm định Lilliefors

h\_lillietest = lillietest(x);

disp(['lillietest result: ', num2str(h\_lillietest)]);

**Mô tả:**

* **ttest2(x, y)**: Kiểm tra xem liệu hai mẫu x và y có khác biệt về trung bình hay không.
* Nếu h\_ttest2 == 1, có sự khác biệt có ý nghĩa giữa x và y.
* Nếu h\_ttest2 == 0, không có sự khác biệt có ý nghĩa giữa x và y.
* **kstest(x)**: Kiểm tra xem liệu dữ liệu trong x có phải là phân phối chuẩn chuẩn hóa hay không.
* Nếu h\_kstest == 1, dữ liệu trong x không đến từ phân phối chuẩn chuẩn hóa.
* Nếu h\_kstest == 0, dữ liệu có thể đến từ phân phối chuẩn chuẩn hóa.
* **lillietest(x)**: Kiểm tra xem liệu dữ liệu trong x có đến từ phân phối chuẩn hay không.
* Nếu h\_lillietest == 1, dữ liệu không đến từ phân phối chuẩn.
* Nếu h\_lillietest == 0, dữ liệu có thể đến từ phân phối chuẩn.

**Kết quả:**

**A black text on a white background

Description automatically generated**

**Code và mô tả:**

x = randn(50, 1); % tạo 50 giá trị ngẫu nhiên với phân phối chuẩn

y = randn(50, 1) + 0.5; % tạo 50 giá trị ngẫu nhiên từ phân phối chuẩn, nhưng với giá trị trung bình khác

h = ttest2(x, y);

disp(h); % In ra kết quả

x = randn(100, 1); % tạo 100 giá trị ngẫu nhiên với phân phối chuẩn

h = kstest(x);

disp(h); % In ra kết quả

x = randn(50, 1); % tạo 50 giá trị ngẫu nhiên với phân phối chuẩn

h = lillietest(x);

disp(h); % In ra kết quả

x = randn(50, 1); % mẫu x

y = randn(50, 1) + 1; % mẫu y với giá trị trung bình khác

p = ranksum(x, y);

disp(p); % In ra giá trị p

testtype = 't2'; % loại kiểm định (t-test hai mẫu độc lập)

p0 = 0.5; % giả thuyết không

p1 = 0.7; % giả thuyết thay thế

nout = sampsizepwr(testtype, p0, p1);

disp(nout); % In ra cỡ mẫu cần thiết

x = randn(30, 1); % mẫu x

y = x + 0.3; % mẫu y có sự thay đổi so với x

h = ttest(x, y);

disp(h); % In ra kết quả

**Mô tả:**

Kiểm định **t-test hai mẫu độc lập** để so sánh hai mẫu (x và y). Kiểm định này kiểm tra giả thuyết không (null hypothesis) rằng các dữ liệu trong x và y đến từ các phân phối chuẩn với trung bình bằng nhau và phương sai không xác định nhưng bằng nhau.

**Giải thích**:

* Nếu kết quả h = 1, thì ta bác bỏ giả thuyết không với mức ý nghĩa 5% (nghĩa là có sự khác biệt đáng kể giữa hai mẫu).
* Nếu h = 0, thì không có đủ chứng cứ để bác bỏ giả thuyết không.

Kiểm định **Kolmogorov-Smirnov một mẫu** kiểm tra giả thuyết không rằng dữ liệu trong x đến từ một phân phối chuẩn chuẩn hóa (mean = 0, variance = 1).

**Giải thích**:

* Nếu kết quả h = 1, thì ta bác bỏ giả thuyết không với mức ý nghĩa 5%, nghĩa là dữ liệu không đến từ phân phối chuẩn chuẩn hóa.
* Nếu h = 0, thì không có đủ chứng cứ để bác bỏ giả thuyết không.

Kiểm định **Lilliefors** là một dạng kiểm định của kiểm định Kolmogorov-Smirnov, nhưng nó kiểm tra dữ liệu xem có phải đến từ phân phối chuẩn không.

**Giải thích**:

* Nếu kết quả h = 1, thì ta bác bỏ giả thuyết không với mức ý nghĩa 5%.
* Nếu h = 0, thì không có đủ chứng cứ để bác bỏ giả thuyết không.

Kiểm định **Wilcoxon rank sum** là một kiểm định phi tham số dùng để so sánh hai mẫu độc lập với nhau, kiểm tra giả thuyết rằng hai mẫu có cùng trung vị.

**Giải thích**:

* Hàm trả về giá trị p, cho phép kiểm tra xem có đủ chứng cứ để bác bỏ giả thuyết không.

Kiểm tra **cỡ mẫu cần thiết** cho phép kiểm định với một mức độ mạnh (power) là 90% và mức ý nghĩa 5%. testtype là loại kiểm định (như 't2' cho kiểm định t hai mẫu độc lập), p0 là tham số dưới giả thuyết không, và p1 là tham số dưới giả thuyết thay thế.

Kiểm định **t-test mẫu ghép** (paired t-test) để so sánh hai mẫu có liên hệ với nhau, kiểm tra giả thuyết không rằng các giá trị trong x và y có trung bình bằng nhau.

**Giải thích**:

* h = 1 nếu ta bác bỏ giả thuyết không, tức là có sự khác biệt đáng kể giữa các giá trị trong x và y.
* h = 0 nếu không có đủ chứng cứ để bác bỏ giả thuyết không.

**Kết quả:**

**A number and numbers on a white background

Description automatically generated**

**Code:**

clear

% Generating synthetic data for normotensive and hypertensive groups

N1 = 100; % Number of normotensive subjects

N2 = 40; % Number of hypertensive subjects

mu1 = 3; % Mean for normotensive group

mu2 = 0; % Mean for hypertensive group

var1 = 1; % Variance for normotensive group

var2 = 1; % Variance for hypertensive group

Normtensive = mu1 + var1 \* randn(1, N1); % Normotensive data (normally distributed)

Hypertensive = mu2 + var2 \* randn(1, N2); % Hypertensive data (normally distributed)

% Plotting histograms for both classes

figure;

histogram(Normtensive, 'FaceColor', 'g', 'EdgeColor', 'black', 'FaceAlpha', 0.5);

hold on;

histogram(Hypertensive, 'FaceColor', 'r', 'EdgeColor', 'black', 'FaceAlpha', 0.5);

% Adding labels

xlabel('Feature value');

ylabel('Frequency');

title('Histogram of Normotensive vs Hypertensive Subjects');

legend('Normotensive', 'Hypertensive');

% Assuming synthetic class labels for normotensive (1) and hypertensive (2)

irisTargets = [ones(1, N1), 2\*ones(1, N2)]; % True labels (1 for normotensive, 2 for hypertensive)

% Simulating predicted outputs as the feature values themselves

irisOutputs = [Normtensive, Hypertensive]; % Classifier outputs (just the feature values for now)

% ROC curve analysis (for the actual classifier output)

[tpr, fpr, thresholds] = roc(irisTargets, irisOutputs);

% Plotting the ROC curve

figure;

plot(fpr, tpr);

xlabel('False Positive Rate');

ylabel('True Positive Rate');

title('ROC Curve');

grid on;

**Mô tả:**

**Dữ liệu tổng quát**:

* **Normotensive**: Là nhóm không cao huyết áp, được mô phỏng bằng phân phối chuẩn với trung bình (mu1 = 3) và phương sai (var1 = 1).
* **Hypertensive**: Là nhóm cao huyết áp, được mô phỏng bằng phân phối chuẩn với trung bình (mu2 = 0) và phương sai (var2 = 1).
* **Mục tiêu** là so sánh hai nhóm này về các đặc tính (đặc trưng là giá trị của phân phối chuẩn).

**irisTargets**:

* Đây là nhãn thực sự (true labels) của các điểm dữ liệu, cho biết mỗi điểm dữ liệu thuộc nhóm nào:
  + Nhóm normotensive được gán nhãn 1.
  + Nhóm hypertensive được gán nhãn 2.

**irisOutputs**:

* Đây là các giá trị được giả định là kết quả dự đoán của một mô hình phân loại. Trong ví dụ này, để đơn giản, chúng ta sử dụng chính các giá trị đặc trưng (Normtensive và Hypertensive) làm dự đoán. Trong thực tế, irisOutputs sẽ là kết quả từ một mô hình phân loại, chẳng hạn như một mô hình học máy.

**Đoạn mã roc(irisTargets, irisOutputs)**:

* Hàm roc tính toán các chỉ số **True Positive Rate (TPR)** và **False Positive Rate (FPR)** cho các ngưỡng (thresholds) khác nhau. Các chỉ số này dùng để đánh giá chất lượng phân loại của mô hình:
  + **TPR (True Positive Rate)**: Tỷ lệ chính xác trong việc phân loại đúng các điểm dữ liệu thuộc lớp thực sự.
  + **FPR (False Positive Rate)**: Tỷ lệ lỗi trong việc phân loại sai các điểm dữ liệu thuộc lớp thực sự.
* Đường cong ROC được vẽ bằng cách vẽ **FPR** trên trục x và **TPR** trên trục y.

**Kết quả:**

**A screenshot of a computer

Description automatically generated**

**Code:**

u1=rand(100); % simulates feature values extracted from Normtensive subjects

u2=rand(100); % simulates feature values extracted from Prehypertensive subjects

u3=rand(100); % simulates feature values extracted from Hypertensive subjects

U = [u1 u2 u3];

anova1(U)

**Mô tả:**

**u1 = rand(100);**: Tạo một ma trận 100x100 với các giá trị ngẫu nhiên từ phân phối đồng đều trên khoảng [0,1], đại diện cho các giá trị đặc trưng của nhóm Normotensive (huyết áp bình thường).

**u2 = rand(100);**: Tạo một ma trận tương tự cho nhóm Prehypertensive (huyết áp tiền tăng).

**u3 = rand(100);**: Tạo một ma trận tương tự cho nhóm Hypertensive (huyết áp cao).

**U = [u1 u2 u3];**: Kết hợp ba ma trận trên thành một ma trận duy nhất, U, có kích thước 100x300.

**anova1(U);**: Thực hiện kiểm tra ANOVA một chiều để xem liệu có sự khác biệt đáng kể về các giá trị đặc trưng giữa ba nhóm không.

**Kết quả:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Code:**

clear

M = meshgrid(1:100); % Tạo một lưới 100x100 với các giá trị từ 1 đến 100

U = rand(100, 3); % Tạo ma trận ngẫu nhiên 100x3 để thay thế U

V = M(:,1:3); % Lấy 3 cột đầu tiên của ma trận M

W = U .\* V; % Nhân từng phần tử của ma trận U và V

anova1(W); % Thực hiện kiểm tra ANOVA trên ma trận W

**Mô tả:**

**M = meshgrid(1:100);**: Tạo một lưới 100x100 từ các giá trị trong khoảng từ 1 đến 100. meshgrid sẽ tạo ra hai ma trận cho các trục X và Y, nhưng trong trường hợp này, chỉ có một ma trận đầu ra M vì bạn chỉ đưa vào một vector.

**U = rand(100, 3);**: Tạo một ma trận ngẫu nhiên U có kích thước 100x3, các giá trị được lấy từ phân phối đồng đều trên [0, 1].

**V = M(:, 1:3);**: Lấy 3 cột đầu tiên của ma trận M. Tuy nhiên, M chỉ có một cột nên đoạn mã này có thể gây lỗi nếu không chỉnh sửa. Trong trường hợp này, hãy xem lại M để chắc chắn rằng bạn đang lấy các giá trị phù hợp.

**W = U .\* V;**: Nhân từng phần tử của ma trận U và ma trận V theo phần tử. Kết quả là ma trận W.

**anova1(W);**: Thực hiện kiểm tra ANOVA một chiều trên ma trận W. Mục đích là kiểm tra xem có sự khác biệt có ý nghĩa thống kê giữa các nhóm trong ma trận W hay không.

**Kết quả:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Code:**

% Giả sử F\_N và F\_H là các ma trận hoặc vector chứa đặc trưng PPG của các nhóm đối tượng

F\_N = rand(100, 1); % Ví dụ, F\_N là đặc trưng PPG của nhóm đối tượng bình thường

F\_H = rand(100, 1); % Ví dụ, F\_H là đặc trưng PPG của nhóm đối tượng tăng huyết áp

m\_N = mean(F\_N); % Tính giá trị trung bình của đặc trưng PPG nhóm bình thường

m\_H = mean(F\_H); % Tính giá trị trung bình của đặc trưng PPG nhóm tăng huyết áp

v\_N = var(F\_N); % Tính phương sai của đặc trưng PPG nhóm bình thường

v\_H = var(F\_H); % Tính phương sai của đặc trưng PPG nhóm tăng huyết áp

J\_F = (m\_N - m\_H)^2 / (v\_N + v\_H); % Tính chỉ số phân biệt giữa hai nhóm

% Hiển thị kết quả

fprintf('Giá trị trung bình của nhóm bình thường (m\_N): %.4f\n', m\_N);

fprintf('Giá trị trung bình của nhóm tăng huyết áp (m\_H): %.4f\n', m\_H);

fprintf('Phương sai của nhóm bình thường (v\_N): %.4f\n', v\_N);

fprintf('Phương sai của nhóm tăng huyết áp (v\_H): %.4f\n', v\_H);

fprintf('Chỉ số phân biệt J\_F: %.4f\n', J\_F);

**Mô tả:**

**F\_N và F\_H**: Đây là các ma trận hoặc vector chứa đặc trưng PPG của hai nhóm đối tượng: nhóm **bình thường (normotensive)** và nhóm **tăng huyết áp (hypertensive)**. Trong ví dụ này, tôi sử dụng rand(100, 1) để tạo ra các giá trị ngẫu nhiên đại diện cho các đặc trưng PPG của từng nhóm.

**m\_N = mean(F\_N); và m\_H = mean(F\_H);**: Tính giá trị trung bình của đặc trưng PPG trong hai nhóm, với m\_N là giá trị trung bình của nhóm bình thường và m\_H là giá trị trung bình của nhóm tăng huyết áp.

**v\_N = var(F\_N); và v\_H = var(F\_H);**: Tính phương sai của đặc trưng PPG trong mỗi nhóm, với v\_N là phương sai của nhóm bình thường và v\_H là phương sai của nhóm tăng huyết áp.

**J\_F = (m\_N - m\_H)^2 / (v\_N + v\_H);**: Tính chỉ số phân biệt giữa hai nhóm, thường được sử dụng trong phân tích phân biệt để đo lường sự khác biệt giữa các nhóm. Công thức này tính độ chênh lệch giữa giá trị trung bình của hai nhóm chia cho tổng phương sai của hai nhóm.

**Kết quả:**

**A close up of text

Description automatically generated**

**Code:**

clear

% Giả sử F\_N và F\_H là các ma trận hoặc vector chứa đặc trưng PPG của các nhóm đối tượng

F\_N = rand(100, 10); % Ví dụ, F\_N là ma trận đặc trưng PPG của nhóm bình thường (100 mẫu, 10 đặc trưng)

F\_H = rand(100, 10); % F\_H là ma trận đặc trưng PPG của nhóm tăng huyết áp (100 mẫu, 10 đặc trưng)

% Tính hiệp phương sai của các đặc trưng

Cov\_N = cov(F\_N); % Hiệp phương sai của nhóm bình thường

Cov\_H = cov(F\_H); % Hiệp phương sai của nhóm tăng huyết áp

% Tính nghịch đảo của các ma trận hiệp phương sai

iCov\_N = inv(Cov\_N); % Nghịch đảo của hiệp phương sai nhóm bình thường

iCov\_H = inv(Cov\_H); % Nghịch đảo của hiệp phương sai nhóm tăng huyết áp

% Tính giá trị trung bình của các đặc trưng

m\_N = mean(F\_N, 1); % Trung bình của nhóm bình thường

m\_H = mean(F\_H, 1); % Trung bình của nhóm tăng huyết áp

% Tính P1 theo công thức

P1 = trace((iCov\_N \* Cov\_H) + (iCov\_H \* Cov\_N) - 2) / 2;

% Tính P2 theo công thức

x1 = (m\_N - m\_H); % Hiệu trung bình

x2 = (iCov\_N + iCov\_H); % Tổng nghịch đảo hiệp phương sai

P2 = (x1 \* x2 \* x1') / 2;

% Tính chỉ số phân biệt J\_d

J\_d = (P1 + P2) / size(F\_N, 2); % Phân biệt chuẩn hóa theo số đặc trưng

% Hiển thị kết quả

fprintf('Chỉ số phân biệt J\_d: %.4f\n', J\_d);

**Mô tả:**

**Cov\_N** và **Cov\_H** là ma trận hiệp phương sai của các đặc trưng PPG cho nhóm đối tượng bình thường (normotensive) và nhóm đối tượng tăng huyết áp (hypertensive).

**iCov\_N** và **iCov\_H** là các ma trận nghịch đảo của các ma trận hiệp phương sai tương ứng.

**m\_N** và **m\_H** là các giá trị trung bình của các đặc trưng PPG trong hai nhóm đối tượng.

**P1** và **P2** là các chỉ số liên quan đến sự khác biệt giữa hai nhóm, được tính bằng các công thức toán học dựa trên các ma trận hiệp phương sai và trung bình.

Cuối cùng, **J\_d** là giá trị đo sự phân biệt giữa hai nhóm, được tính từ **P1** và **P2**.

**Kết quả:**

****

**Code:**

% Giả sử F\_N và F\_H là các ma trận chứa đặc trưng PPG của hai nhóm đối tượng

F\_N = rand(100, 10); % Ví dụ: F\_N là ma trận đặc trưng PPG của nhóm bình thường (100 mẫu, 10 đặc trưng)

F\_H = rand(100, 10); % F\_H là ma trận đặc trưng PPG của nhóm tăng huyết áp (100 mẫu, 10 đặc trưng)

% Tính hiệp phương sai của các đặc trưng

Cov\_N = cov(F\_N); % Hiệp phương sai của nhóm bình thường

Cov\_H = cov(F\_H); % Hiệp phương sai của nhóm tăng huyết áp

% Tính giá trị trung bình của các đặc trưng

m\_N = mean(F\_N, 1); % Trung bình của nhóm bình thường

m\_H = mean(F\_H, 1); % Trung bình của nhóm tăng huyết áp

% Tính P1 theo công thức (hiệu trung bình giữa hai nhóm)

P1 = (m\_N - m\_H);

% Tính P2 (nghịch đảo của ma trận hiệp phương sai trung bình)

P2 = inv((Cov\_N + Cov\_H) / 2);

% Tính P3 (log-determinant để đo sự phân biệt)

P3 = 0.5 \* log(det(Cov\_N + Cov\_H) / sqrt(det(Cov\_N) + det(Cov\_H)));

% Tính P4 (chỉ số phân biệt kết hợp giữa P1, P2 và P3)

% Cần sử dụng phép nhân ma trận hợp lệ

P4 = (P1 \* P2 \* P1') / 8 + P3;

% Tính chỉ số phân biệt J\_d

J\_d = P4 / size(F\_N, 2); % Phân biệt chuẩn hóa theo số đặc trưng

% Hiển thị kết quả

fprintf('Chỉ số phân biệt J\_d: %.4f\n', J\_d);

**Mô tả:**

* **Cov\_N** và **Cov\_H**: Tính ma trận hiệp phương sai cho hai nhóm, giúp đo lường sự biến động giữa các đặc trưng PPG trong mỗi nhóm.
* **m\_N** và **m\_H**: Trung bình của các đặc trưng PPG trong mỗi nhóm, giúp xác định "tâm" của phân phối đặc trưng.
* **P1**: Hiệu giữa trung bình của hai nhóm, biểu thị sự khác biệt giữa chúng.
* **P2**: Nghịch đảo của ma trận hiệp phương sai trung bình, dùng để tính toán mức độ "đồng nhất" giữa hai nhóm.
* **P3**: Được tính từ log-determinant, giúp đo lường sự phân biệt giữa hai ma trận hiệp phương sai.
* **P4**: Kết hợp các yếu tố từ **P1**, **P2** và **P3** để tính toán chỉ số phân biệt.
* **J\_d**: Chỉ số phân biệt cuối cùng, được chuẩn hóa theo số đặc trưng (cột trong ma trận **F\_N**).

**Kết quả:**

**A black and white text

Description automatically generated**

**Code:**

clear

% Giả sử F\_N và F\_H là các ma trận đặc trưng PPG của hai nhóm đối tượng

F\_N = rand(100, 10); % Ví dụ: F\_N là ma trận đặc trưng PPG của nhóm bình thường (100 mẫu, 10 đặc trưng)

F\_H = rand(100, 10); % F\_H là ma trận đặc trưng PPG của nhóm tăng huyết áp (100 mẫu, 10 đặc trưng)

% 1. Tính hiệp phương sai của mỗi nhóm

Cov\_N = cov(F\_N, 1); % Hiệp phương sai của nhóm bình thường

Cov\_H = cov(F\_H, 1); % Hiệp phương sai của nhóm tăng huyết áp

% 2. Tính số lượng mẫu trong mỗi nhóm

n1 = size(F\_N, 1); % Số mẫu của nhóm bình thường

n2 = size(F\_H, 1); % Số mẫu của nhóm tăng huyết áp

% 3. Tính tổng số mẫu

N = n1 + n2;

% 4. Tính ma trận hiệp phương sai giữa hai nhóm

Sw = ((n1/N) \* Cov\_N + (n2/N) \* Cov\_H);

% 5. Kết hợp cả hai nhóm lại thành ma trận c

c = [F\_N; F\_H];

% 6. Tính hiệp phương sai tổng thể (covariance matrix của cả hai nhóm)

Sm = cov(c, 1);

% 7. Tính chỉ số phân biệt J\_s

J\_s = trace(inv(Sw) \* Sm) / size(F\_N, 2);

% Hiển thị kết quả

fprintf('Chỉ số phân biệt J\_s: %.4f\n', J\_s);

**Mô tả:**

* **Tính hiệp phương sai**:
* Cov\_N = cov(F\_N, 1) và Cov\_H = cov(F\_H, 1) tính hiệp phương sai của các đặc trưng trong hai nhóm.
* **Tính kích thước**:
* n1 = size(F\_N, 1) và n2 = size(F\_H, 1) tính số lượng mẫu (dòng) trong mỗi nhóm.
* **Tính hiệp phương sai giữa các nhóm (Sw)**:
* Sw là ma trận hiệp phương sai giữa các nhóm. Điều này được tính bằng cách lấy trọng số giữa hiệp phương sai của nhóm F\_N và F\_H, với trọng số dựa trên số lượng mẫu của mỗi nhóm.
* **Tính ma trận hiệp phương sai tổng thể (Sm)**:
* Sm = cov(c, 1) tính ma trận hiệp phương sai của tất cả các mẫu từ hai nhóm kết hợp lại.
* **Tính chỉ số phân biệt (J\_s)**:
* J\_s = trace(inv(Sw) \* Sm) / size(F\_N, 2) tính chỉ số phân biệt giữa các nhóm bằng cách sử dụng ma trận hiệp phương sai riêng và tổng thể.

**Kết quả:**

**A black and white text

Description automatically generated**

**Code:**

% Hàm tính chỉ số phân biệt J

function J = Compute\_J(f, F\_N, F\_H)

% Tính trung bình và hiệp phương sai của hai nhóm

mean\_N = mean(F\_N(:, f), 1);

mean\_H = mean(F\_H(:, f), 1);

cov\_N = cov(F\_N(:, f));

cov\_H = cov(F\_H(:, f));

% Tính P1 và P2 theo công thức phân biệt

P1 = (mean\_N - mean\_H).^2; % Sửa phép toán bình phương theo phần tử

P2 = inv((cov\_N + cov\_H) / 2);

% Chỉ số phân biệt J

J = (P1 \* P2 \* P1') / 8 + 0.5 \* log(det((cov\_N + cov\_H) / 2) / sqrt(det(cov\_N) \* det(cov\_H)));

end

% Giả sử F\_N và F\_H là các ma trận đặc trưng PPG của hai nhóm đối tượng

F\_N = rand(100, 10); % Ví dụ: F\_N là ma trận đặc trưng PPG của nhóm bình thường (100 mẫu, 10 đặc trưng)

F\_H = rand(100, 10); % F\_H là ma trận đặc trưng PPG của nhóm tăng huyết áp (100 mẫu, 10 đặc trưng)

NumOfRequiredFeatures = 2; % Số lượng đặc trưng cần chọn

NofFeatures = size(F\_N, 2); % Số lượng đặc trưng có sẵn (lấy từ số cột của F\_N)

J\_max = 0; % Giá trị của chỉ số phân biệt lớn nhất

Best\_Features = []; % Mảng lưu các đặc trưng tốt nhất

counter = 1; % Bộ đếm số lượng đặc trưng đã chọn

% Vòng lặp chọn đặc trưng tốt nhất

while counter <= NumOfRequiredFeatures

maxJ = 0; % Giá trị chỉ số phân biệt lớn nhất hiện tại

% Duyệt qua tất cả các đặc trưng

for i = 1:NofFeatures

if isempty(find(Best\_Features == i, 1)) % Kiểm tra nếu đặc trưng chưa được chọn

f = [Best\_Features i]; % Thêm đặc trưng vào mảng Best\_Features

% Tính chỉ số phân biệt J của các đặc trưng đã chọn

J = Compute\_J(f, F\_N, F\_H); % Tính J cho các đặc trưng đã chọn

if J > maxJ % Nếu chỉ số phân biệt của đặc trưng này tốt hơn hiện tại

maxJ = J; % Cập nhật giá trị J\_max

Best\_Features = f; % Cập nhật mảng Best\_Features

end

end

end

% Thêm đặc trưng có chỉ số phân biệt tốt nhất vào danh sách

counter = counter + 1;

end

% Hiển thị các đặc trưng tốt nhất đã chọn

fprintf('Các đặc trưng tốt nhất đã chọn: ');

disp(Best\_Features);

**Mô tả:**

**F\_N và F\_H**: Các ma trận chứa các đặc trưng PPG của nhóm bình thường (normotensive) và nhóm tăng huyết áp (hypertensive).

**NumOfRequiredFeatures**: Biến này xác định số lượng đặc trưng bạn muốn chọn.

**Vòng lặp chính (while)**:

* Chạy đến khi chọn đủ số lượng đặc trưng yêu cầu (NumOfRequiredFeatures).
* Trong mỗi vòng lặp, nó tìm kiếm đặc trưng có chỉ số phân biệt (J) cao nhất và thêm vào danh sách các đặc trưng tốt nhất (Best\_Features).

**Vòng lặp con (for)**:

* Duyệt qua tất cả các đặc trưng có sẵn.
* Kiểm tra xem đặc trưng có trong danh sách Best\_Features hay chưa (isempty(find(...))).
* Nếu chưa có, tính toán chỉ số phân biệt (J) của đặc trưng đó và cập nhật danh sách Best\_Features nếu chỉ số phân biệt lớn hơn giá trị hiện tại.

**Hàm Compute\_J**: Đây là hàm tính chỉ số phân biệt giữa hai nhóm dựa trên các đặc trưng đã chọn. Hàm này có thể giống như các phương pháp phân biệt mà bạn đã sử dụng trước đó, ví dụ: tính khoảng cách Mahalanobis, phân tích Fisher, hoặc các phương pháp phân tích khác.

**Kết quả:**

****