**Code:**

clear

% Tạo dữ liệu mẫu cho sự kiện được chú thích (annotated) và phát hiện (extracted)

x1 = [0.1, 0.5, 1.0, 1.5, 2.0]; % sự kiện chú thích (đơn vị có thể là giây)

x2 = [0.12, 0.52, 1.05, 1.58, 2.1]; % sự kiện phát hiện (đơn vị có thể là giây)

MaxDifference = 0.02; % Sự chênh lệch tối đa cho phép giữa các sự kiện

% Khởi tạo các biến để lưu trữ kết quả

TP\_inAnnotate = [];

TP\_inExtracted = [];

FN = [];

FP = [];

% Duyệt qua từng sự kiện trong x1

for i = 1:length(x1)

% Tìm các sự kiện trong x2 có sự chênh lệch nhỏ hơn MaxDifference

t = find(abs(x1(i) - x2) < MaxDifference);

if ~isempty(t)

% Cập nhật các sự kiện TP

TP\_inAnnotate = [TP\_inAnnotate, i];

TP\_inExtracted = [TP\_inExtracted, t];

end

end

% Tính toán số lượng True Positives (TP)

TP = length(TP\_inAnnotate);

% Tính toán các sự kiện False Negatives (FN)

FN\_events = x1;

FN\_events(TP\_inAnnotate) = [];

FN = length(FN\_events);

% Tính toán các sự kiện False Positives (FP)

FP\_events = x2;

FP\_events(TP\_inExtracted) = [];

FP = length(FP\_events);

% Tính toán các chỉ số hiệu suất

SE = TP / (TP + FN); % Sensitivity

PP = TP / (TP + FP); % Positive Predictive Value

% Hiển thị kết quả

disp(['True Positives: ', num2str(TP)]);

disp(['False Negatives: ', num2str(FN)]);

disp(['False Positives: ', num2str(FP)]);

disp(['Sensitivity: ', num2str(SE)]);

disp(['Positive Predictive Value: ', num2str(PP)]);

**Mô tả:**

**TP (True Positives)**: Sự kiện được chú thích mà thuật toán phát hiện đúng.

**FN (False Negatives)**: Sự kiện được chú thích nhưng không được thuật toán phát hiện.

**FP (False Positives)**: Sự kiện mà thuật toán phát hiện nhưng không có trong tập chú thích.

**Sensitivity (SE)**: Tỷ lệ giữa số sự kiện đúng trên tổng số sự kiện thực sự.

**Positive Predictive Value (PP)**: Tỷ lệ giữa số sự kiện đúng trên tổng số sự kiện mà thuật toán phát hiện.

**Kết quả:**

**A black text on a white background

Description automatically generated**

**Code:**

clear

% Hàm giả lập DETECTOR (dễ dàng thay thế bằng hàm thực tế của bạn)

function ExtractedEvents = DETECTOR(F1, F2, W1, W2, beta)

% Giả lập việc trích xuất sự kiện từ các tham số

% Trong thực tế, hàm này sẽ thực hiện các phép tính và trả về kết quả

ExtractedEvents = F1 \* sin(F2) + W1 \* cos(W2) + beta; % Đây chỉ là một ví dụ giả lập

end

% Hàm giả lập calculateAccuracy

function [SE, PP] = calculateAccuracy(AnnotateEvents, ExtractedEvents)

% Tính số lượng TP, FN, FP

MaxDifference = 0.02; % Giới hạn tối đa giữa sự kiện chú thích và sự kiện phát hiện

TP\_inAnnotate = [];

TP\_inExtracted = [];

FN = [];

FP = [];

for i = 1:length(AnnotateEvents)

t = find(abs(AnnotateEvents(i) - ExtractedEvents) < MaxDifference);

if ~isempty(t)

TP\_inAnnotate = [TP\_inAnnotate, i];

TP\_inExtracted = [TP\_inExtracted, t];

end

end

% Tính TP, FN, FP

TP = length(TP\_inAnnotate); % Sự kiện đúng (được phát hiện và đúng)

FN\_events = AnnotateEvents;

FN\_events(TP\_inAnnotate) = [];

FN = length(FN\_events); % Sự kiện bị bỏ lỡ

FP\_events = ExtractedEvents;

FP\_events(TP\_inExtracted) = [];

FP = length(FP\_events); % Sự kiện phát hiện sai

% Tính SE và PP

SE = TP / (TP + FN); % Độ nhạy

PP = TP / (TP + FP); % Giá trị dự đoán tích cực

end

% Dữ liệu mẫu cho các tham số vector

F1\_vector = [0.1, 0.2, 0.3]; % Các giá trị có thể chọn cho F1

F2\_vector = [0.5, 1.0]; % Các giá trị có thể chọn cho F2

W1\_vector = [10, 20]; % Các giá trị có thể chọn cho W1

W2\_vector = [5, 15]; % Các giá trị có thể chọn cho W2

beta\_vector = [0.5, 1.0]; % Các giá trị có thể chọn cho beta

% Dữ liệu mẫu cho AnnotateEvents (các sự kiện được chú thích)

AnnotateEvents = [0.1, 0.5, 1.0, 1.5, 2.0]; % Các sự kiện được chú thích (đơn vị có thể là giây)

% Khởi tạo mảng để lưu kết quả

SE = [];

PP = [];

parameters = [];

n = 1;

% Vòng lặp qua tất cả các giá trị tham số

for F1 = F1\_vector

for F2 = F2\_vector

for W1 = W1\_vector

for W2 = W2\_vector

for beta = beta\_vector

% Trích xuất sự kiện bằng thuật toán TERMA

ExtractedEvents = DETECTOR(F1, F2, W1, W2, beta);

% Tính toán độ chính xác SE và PP

[SE(n), PP(n)] = calculateAccuracy(AnnotateEvents, ExtractedEvents);

% Lưu kết quả

parameters(n, :) = [F1, F2, W1, W2, beta];

n = n + 1;

end

end

end

end

end

% Lưu trữ kết quả

Results = [SE(:), PP(:), parameters];

% Sắp xếp kết quả theo thứ tự giảm dần

SortedResults = sortrows(Results, 'descend');

% Chọn tham số tốt nhất

BestParameters = SortedResults(1, 3:end);

% Hiển thị kết quả

disp(['Best Parameters: ', num2str(BestParameters)]);

**Mô tả:**

Đoạn mã thực hiện việc tối ưu hóa các tham số của thuật toán **TERMA** bằng cách thử tất cả các kết hợp của tham số từ các vector F1\_vector, F2\_vector, W1\_vector, W2\_vector, và beta\_vector.

Sau đó, tính toán độ chính xác (Sensitivity và Positive Predictive Value) cho mỗi bộ tham số và lưu trữ kết quả.

Cuối cùng, sắp xếp các kết quả và chọn bộ tham số cho kết quả tốt nhất (SE và PP cao nhất).

**Kết quả:**

****

**Code:**

clear

% Hàm giả lập DETECTOR (dễ dàng thay thế bằng hàm thực tế của bạn)

function ExtractedEvents = DETECTOR(F1, F2, W1, W2, beta)

% Giả lập việc trích xuất sự kiện từ các tham số

% Trong thực tế, hàm này sẽ thực hiện các phép tính và trả về kết quả

ExtractedEvents = F1 \* sin(F2) + W1 \* cos(W2) + beta; % Đây chỉ là một ví dụ giả lập

end

% Hàm giả lập calculateAccuracy

function [SE, PP] = calculateAccuracy(AnnotateEvents, ExtractedEvents)

% Tính số lượng TP, FN, FP

MaxDifference = 0.02; % Giới hạn tối đa giữa sự kiện chú thích và sự kiện phát hiện

TP\_inAnnotate = [];

TP\_inExtracted = [];

FN = [];

FP = [];

for i = 1:length(AnnotateEvents)

t = find(abs(AnnotateEvents(i) - ExtractedEvents) < MaxDifference);

if ~isempty(t)

TP\_inAnnotate = [TP\_inAnnotate, i];

TP\_inExtracted = [TP\_inExtracted, t];

end

end

% Tính TP, FN, FP

TP = length(TP\_inAnnotate); % Sự kiện đúng (được phát hiện và đúng)

FN\_events = AnnotateEvents;

FN\_events(TP\_inAnnotate) = [];

FN = length(FN\_events); % Sự kiện bị bỏ lỡ

FP\_events = ExtractedEvents;

FP\_events(TP\_inExtracted) = [];

FP = length(FP\_events); % Sự kiện phát hiện sai

% Tính SE và PP

SE = TP / (TP + FN); % Độ nhạy

PP = TP / (TP + FP); % Giá trị dự đoán tích cực

end

clear

% Dữ liệu mẫu cho các tham số vector

F1\_vector = [0.1, 0.2, 0.3]; % Các giá trị có thể chọn cho F1

F2\_vector = [0.5, 1.0]; % Các giá trị có thể chọn cho F2

W1\_vector = [10, 20]; % Các giá trị có thể chọn cho W1

W2\_vector = [5, 15]; % Các giá trị có thể chọn cho W2

beta\_vector = [0.5, 1.0]; % Các giá trị có thể chọn cho beta

% Dữ liệu mẫu cho AnnotateEvents (các sự kiện được chú thích)

AnnotateEvents = [0.1, 0.5, 1.0, 1.5, 2.0]; % Các sự kiện được chú thích (đơn vị có thể là giây)

% Khởi tạo mảng để lưu kết quả

SE = [];

PP = [];

parameters = [];

n = 1;

% Vòng lặp qua tất cả các giá trị tham số

for F1 = F1\_vector

for F2 = F2\_vector

for W1 = W1\_vector

for W2 = W2\_vector

for beta = beta\_vector

% Trích xuất sự kiện bằng thuật toán TERMA (sử dụng hàm giả lập)

ExtractedEvents = DETECTOR(F1, F2, W1, W2, beta);

% Tính toán độ chính xác SE và PP

[SE(n), PP(n)] = calculateAccuracy(AnnotateEvents, ExtractedEvents);

% Lưu kết quả

parameters(n, :) = [F1, F2, W1, W2, beta];

n = n + 1;

end

end

end

end

end

% Lưu trữ kết quả

Results = [SE(:), PP(:), parameters];

% Sắp xếp kết quả theo thứ tự giảm dần

SortedResults = sortrows(Results, 'descend');

% Chọn tham số tốt nhất

BestParameters = SortedResults(1, 3:end);

% Hiển thị kết quả

disp(['Best Parameters: ', num2str(BestParameters)]);

**Mô tả:**

**Hàm ConstrainedFun**:

**Ràng buộc bất đẳng thức**:

c(1) = para(1) - para(2); % F1 < F2: Đảm bảo rằng F1 nhỏ hơn F2.

c(2) = para(3) - para(4); % W1 < W2: Đảm bảo rằng W1 nhỏ hơn W2.

**Ràng buộc đẳng thức**: Không có ràng buộc đẳng thức nào trong ví dụ này, vì vậy ceq = [].

**Hàm mục tiêu (objectiveFun)**:

Hàm mục tiêu này sử dụng hàm calculateAccuracy để tính toán độ chính xác của các sự kiện được trích xuất (tính toán bằng DETECTOR với các tham số [F1, F2, W1, W2, beta]). Mục tiêu là tối đa hóa độ chính xác, tức là tối thiểu hóa 1 - calculateAccuracy.

**Tối ưu hóa với fmincon**:

fmincon được sử dụng để tối ưu hóa hàm mục tiêu. Nó nhận tham số ban đầu para0, các giới hạn thấp và cao của tham số, và hàm ràng buộc (được định nghĩa trong ConstrainedFun).

Sau khi tối ưu hóa, optimalPara sẽ chứa các tham số tối ưu.

**Kết quả:**

****