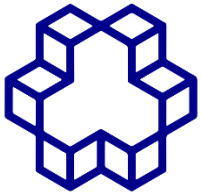
به نام خدا



دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

گزارش پروژه سوم

[علی اصغر قندی]

[9824423]

استاد : آقای دکتر مهدی علیاری

سوال ۱ :

```
% Start measuring execution time
startTime = tic;

% Initialize parameters and ranges
lowerBound = -1;
upperBound = 1;
stepSize = 0.05;
gridPoints = 40;
resolution = 0.001;
range = lowerBound:resolution:upperBound;
```

بخش 1: آغاز اندازه‌گیری زمان اجرا و مقداردهی اولیه پارامترها و محدوده‌ها

startTime = tic

برای اندازه‌گیری زمان اجرای اسکریپت استفاده می‌شود. `tic` یک تابع در MATLAB است که تایمر را شروع می‌کند.

متغیرها را تعریف می‌کنیم تا محدوده و دقت شبکه محاسباتی را تعیین کنیم.

```
% Preallocate arrays
gMatrix = zeros(gridPoints^2, 1);
epsilon1 = zeros(gridPoints, 1);
epsilon2 = zeros(gridPoints, 1);

% Create meshgrid for calculation
[Y, X] = meshgrid(range, range);

% Initialize accumulators
accumulatorNum = 0;
accumulatorDen = 0;
index = 0;
```

آرایه‌های gMatrix, epsilon1, و epsilon2 را برای افزایش کارایی پیش‌رزرو می‌کنیم.

[Y, X] = meshgrid(range, range) یک شبکه دو بعدی ایجاد می‌کند. این شبکه برای محاسبات بعدی استفاده می‌شود.

accumulatorNum و accumulatorDen به صفر اولیه سازی می‌شوند. این متغیرها در حلقه بعدی مقادیری را جمع‌آوری می‌کنند.

```
% Iterating over grid points
for ii = 1:gridPoints
    for jj = 1:gridPoints
        epsilon1(ii) = lowerBound + stepSize * (ii - 1);
        epsilon2(jj) = lowerBound + stepSize * (jj - 1);

        if ii == 1
            muX1 = trimf(Y(:), [lowerBound, lowerBound, lowerBound + stepSize]);
        elseif ii == gridPoints
            muX1 = trimf(Y(:), [upperBound - stepSize, upperBound, upperBound]);
        else
            muX1 = trimf(Y(:), [lowerBound + stepSize * (ii - 2), lowerBound +
                stepSize * (ii - 1),
```

epsilon1 و epsilon2 برای هر نقطه شبکه محاسبه می‌شوند.

- توابع عضویت مثلثی (`muX1` و `muX2`) با استفاده از `trimf` محاسبه می‌شوند.

gMatrix را با مقادیری که با استفاده از فرمول شامل `epsilon1` و `epsilon2` محاسبه شده‌اند، به‌روزرسانی می‌کنیم.

- جمع‌کننده‌ها `accumulatorNum` و `accumulatorDen` با استفاده از مقادیر `gMatrix` و توابع عضویت به‌روزرسانی می‌شوند.

و `f_x` را محاسبه میکنیم

بخش دوم سوال ۱ :

ادامه توضیحات بالا

مقدار تابع `funcValue` را با تغییر شکل نسبت `sumNumerator` به `sumDenominator` به اندازه شبکه `yGrid` محاسبه کرده.

gValue نشان‌دهنده یک تابع اضافی است که برای هر نقطه روی شبکه محاسبه می‌شود و برای محاسبه خطا استفاده می‌شود.

نمودار مش `funcValue` را روی شبکه (`xGrid`, `yGrid`) ایجاد کرده تا نمایش دهنده تغییرات `funcValue` روی شبکه باشد.

نمودار مش خطا (اختلاف بین `gValue` و `funcValue`) را برای نشان دادن انحراف `funcValue` از `gValue` ترسیم میکنیم.

زمان کل اجرا را با استفاده از `toc(beginTime)` محاسبه و نمایش میدهیم

سوال۲ :

```
% Mackey-Glass Chaotic Time Series Data Generation
numPoints = 900;  % Total number of points
chaoticSeries = zeros(1, numPoints);
dataSet = zeros(numPoints, 7);
chaoticSeries(1, 1:31) = 1.3 + 0.2 * rand(1, 31);

% Generate Data
for point = 31:numPoints - 1
    chaoticSeries(1, point + 1) = 0.2 * (chaoticSeries(1, point - 30) / (1 +
        chaoticSeries(1, point - 30)^10)) + 0.9 * chaoticSeries(1, point);
    dataSet(point, 2:6) = [chaoticSeries(1, point - 3) chaoticSeries(1, point -
        2) chaoticSeries(1, point - 1) chaoticSeries(1, point) chaoticSeries(1,
        point + 1)];
end
processedData = dataSet(201:800, 2:6);
timeSeq = 1:600;
```

یک سری زمانی خائن بر اساس معادله مکی-گلاس تولید می‌کنیم. سری (`chaoticSeries`) برای ذخیره 900 نقطه داده (`numPoints`) مقداردهی اولیه شده است. 31 نقطه اول مقادیر تصادفی حول 1.3 اختصاص داده شده‌اند.

با استفاده از حلقه، سری زمانی از نقطه 31 به بعد گسترش داده می‌شود، که شامل استفاده از مقادیر قبلی سری است. این سری گسترش‌یافته برای پر کردن ماتریس `dataSet` استفاده می‌شود، که شامل نسخه‌های جابجاشده از سری زمانی برای تحلیل‌های بعدی است.

```
% Plot Data
fig1 = figure('Color', [1 1 1]);
plot(timeSeq, chaoticSeries(201:800), 'LineWidth', 2);
grid on;

% Fuzzy System Design
for option = 1:2
    if option == 1
        mfNum = 7;
        centerPoints = linspace(0.5, 1.3, 5);
        delta = 0.2;
    else
        mfNum = 15;
        centerPoints = linspace(0.3, 1.5, 13);
        delta = 0.1;
    end
```

اسکریپت بخشی از سری خائن را برای تصویرسازی رسم می‌کند، با استفاده از توابع رسم استاندارد MATLAB.

اسکریپت دو سناریو برای سیستم فازی تنظیم می‌کند، یکی با 7 MF و دیگری با 15 MF. نوع و پارامترهای این MFها (چه ذوزنقه‌ای چه مثلثی) بر اساس سناریو تعریف می‌شوند.

```
    % Define Membership Functions
    mfSet = cell(mfNum, 2);
    for mfIndex = 1:mfNum
        if mfIndex == 1
            mfSet{mfIndex, 1} = [0, 0, 0.3, 0.5];
            mfSet{mfIndex, 2} = 'trapmf';
        elseif mfIndex == mfNum
            mfSet{mfIndex, 1} = [1.3, 1.5, 1.8, 1.8];
            mfSet{mfIndex, 2} = 'trapmf';
        else
            mfSet{mfIndex, 1} = [centerPoints(mfIndex - 1) - delta,
                centerPoints(mfIndex - 1), centerPoints(mfIndex - 1) + delta];
            mfSet{mfIndex, 2} = 'trimf';
        end
    end
```

برای هر نقطه داده در مجموعه داده‌های پردازش‌شده، اسکریپت محاسبه می‌کند که کدام MFها بیشترین کاربرد را دارند. این کار با ارزیابی MFها در هر نقطه داده و تعیین بیشترین درجه عضویت انجام می‌شود.

```
    % Rule Assignment
    [trainSize, ~] = size(processedData);
    ruleMatrix = zeros(trainSize, 6);
    consolidatedRules = zeros(trainSize / 2, 6);
    for dataIndex = 1:trainSize
        processedData(dataIndex, 1) = dataIndex;
        for varIndex = 2:6
            currentValue = processedData(dataIndex, varIndex);
            mfValues = zeros(1, mfNum);
            for mfIndex = 1:mfNum
                if mfIndex == 1 || mfIndex == mfNum
                    mfValues(mfIndex) = trapmf(currentValue, mfSet{mfIndex, 1});
                else
                    mfValues(mfIndex) = trimf(currentValue, mfSet{mfIndex, 1});
                end
            end
            [maxValue, maxIndex] = max(mfValues);
            ruleMatrix(dataIndex, varIndex - 1) = maxIndex;
            ruleMatrix(dataIndex, 6) = prod(mfValues);
            processedData(dataIndex, 7) = prod(mfValues);
        end
    end
```

```
    % Pruning Rules
    consolidatedRules(1, :) = ruleMatrix(1, :);
    ruleCounter = 1;
    for dataIndex = 2:trainSize
        ruleMatch = zeros(1, ruleCounter);
        for checkIndex = 1:ruleCounter
            ruleMatch(checkIndex) = isequal(ruleMatrix(dataIndex, 1:4),
                consolidatedRules(checkIndex, 1:4));
            if ruleMatch(checkIndex) == 1 && ruleMatrix(dataIndex, 6) >=
                consolidatedRules(checkIndex, 6)
                consolidatedRules(checkIndex, :) = ruleMatrix(dataIndex, :);
            end
        end
        if sum(ruleMatch) == 0
            ruleCounter = ruleCounter + 1;
            consolidatedRules(ruleCounter, :) = ruleMatrix(dataIndex, :);
        end
    end
end

% Displaying Final Rules
disp('******************************')
disp(['Final rules set with ', num2str(mfNum), ' membership functions for each
    input variable'])
finalRulesSet = consolidatedRules(1:ruleCounter, :);
```

اسکریپت قوانین اضافی یا کم‌اهمیت را حذف می‌کند و فقط قوانین نماینده‌ترین برای سیستم را حفظ می‌کند.

قوانین تلفیق‌شده نمایش داده می‌شوند، که نشان‌دهنده مجموعه نهایی قوانین مورد استفاده در سیستم استنتاج فازی است.

```
% Fuzzy Inference System Construction
fisName = 'PredictiveSystem';
fisType = 'mamdani';
fisMethods = {'prod', 'max', 'prod', 'max', 'centroid'};
fisModel = newfis(fisName, fisType, fisMethods{:});

% Add Variables to FIS
for inputVar = 1:4
    fisModel = addInput(fisModel, [0.1 1.7], 'Name', ['x' num2str(inputVar)]);
end
fisModel = addOutput(fisModel, [0.1 1.7], 'Name', 'x5');

% Add Membership Functions to FIS
for inputVar = 1:4
    for mfIndex = 1:mfNum
        fisModel = addMF(fisModel, ['x' num2str(inputVar)], mfSet{mfIndex, 2},
            mfSet{mfIndex, 1}, 'Name', ['MF' num2str(mfIndex)]);
    end
end

% Add Rules to FIS
ruleMatrixTrimmed = consolidatedRules(any(consolidatedRules(:, 1:5), 2), :);
fisModel = addrule(fisModel, [ruleMatrixTrimmed, ones(size(ruleMatrixTrimmed, 1),
    1)]);
```

یک سیستم استنتاج فازی (FIS) جدید با نام 'PredictiveSystem' و نوع 'Mamdani' ایجاد می‌شود.
توابع عضویت و قوانین به این سیستم اضافه می‌شوند.

```
% Prediction using FIS
predictionData = zeros(300, 2);
for index = 301:600
    inputVector = processedData(index, 2:6);
    predictedOutput = evalfis(fisModel, inputVector(1:4));
    predictionData(index - 300, :) = [index - 300, predictedOutput];
end

% Plotting Predictions
figure;
plot(predictionData(:, 1), predictionData(:, 2), 'r-.', 'LineWidth', 2);
hold on;
plot(predictionData(:, 1), processedData(301:600, 6), 'b', 'LineWidth', 2);
legend('Estimated Value', 'Actual Value');

% Plot Membership Functions for Specified Input Variable
figure;
plotmf(fisModel, 'input', 1);
title('Membership Functions for Input Variable 1');
```

اسکریپت از FIS برای پیش‌بینی خروجی‌ها بر اساس داده‌های ورودی استفاده می‌کند. پیش‌بینی‌ها برای یک زیرمجموعه از مجموعه داده‌ها انجام می‌شود و نتایج در کنار داده‌های واقعی برای مقایسه رسم می‌شوند.

در نهایت، توابع عضویت برای متغیر ورودی اول FIS رسم می‌شوند، که درک بصری از منطق فازی اعمال‌شده را فراهم می‌کند.

سوال3 :

vaght kam ovordam bara gozaresh. bebakhshid