به نام خدا

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی سیستم های هوشمند

مینی پروژه سوم

لینک کد:

https://colab.research.google.com/drive/1IAip20dLskVH6GLQptSZsjaXra1FawxE?usp=drive_link

لینک گیت:

https://github.com/MMM1381/Fundamentals-of-intelligence-system-projects

محمدمهدی معاضدی

9930473

استاد : آقای دکتر مهدی علیاری

1403

مقدمه

توضیحات پیاده‌سازی ANFIS

در سوال پنجم، برای پیاده‌سازی ANFIS با استفاده از پایتون از کتاب Deep Neuro-Fuzzy Systems with Python: With Case Studies and Himanshu استفاده شده است. در این کتاب، برای پیاده‌سازی ANFIS از یک کتابخانه خاص استفاده شده که نیازمند نصب است.

اما با توجه به تجربه کار عملی:

کتابخانه ذکرشده برای import توابع مختلف با مشکلاتی (باگ‌ها) همراه بود.

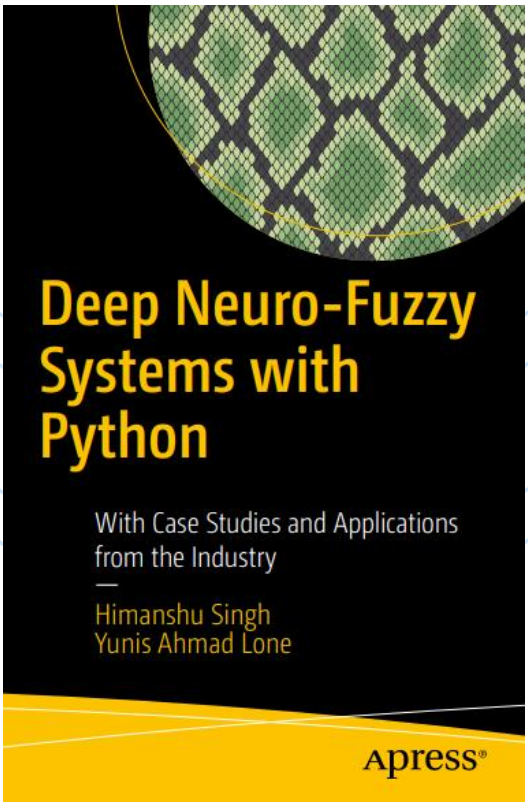
برای رفع این مشکل، بهترین راه دانلود کدها از ریپازیتوری رسمی موجود در GitHub و استفاده مستقیم از فایل‌های anfis.py و فایل‌های داخل پوشه membershipfunction است.

پس فایل anfis.py و پوشه membershipfunction باید حتما در مکانی که کد اجرا می شود باشند

لینک ریپازیتوری: https://github.com/twmeggs/anfis

این روش تضمین می‌کند که وابستگی‌ها و توابع لازم بدون مشکلات اضافی اجرا شوند.

نکته من در بقیه سوالات anfis رو در متلب زدم اونجا از توابع خود متلب استفاده کردم



فهرست مطالب

پرسش دوم 4
پارک کردن ماشین با منطق فازی 4
پرسش سوم 8
Ball and Beam 8
پرسش چهارم 11
شناسایی یک سیستم غیرخطی با فازی و گرادیان کاهشی 11
پرسش پنجم 15
نتایج مدل‌های RBF و ANFIS 15
پایان 18

پرسش دوم

پارک کردن ماشین با منطق فازی

کد فایل : car.m و Car.fis

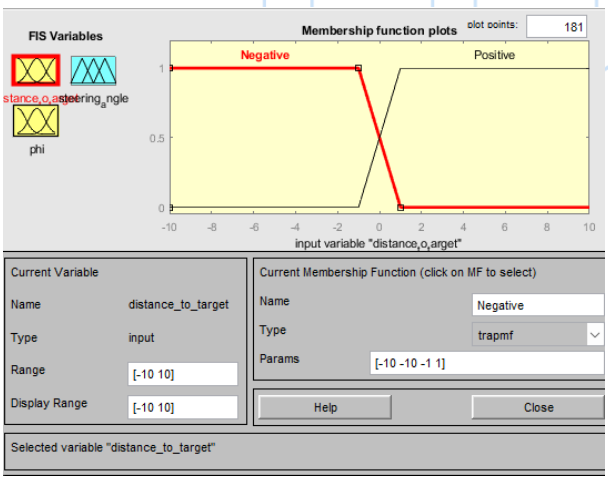
هدف این مسأله پارک کردن یک ماشین در نقطه که $(x = 10$ و $\phi = 90)$ ϕ نشان‌دهنده زاویه دوران ماشین است می‌باشد. ماشین فقط به صورت دنده عقب حرکت می‌کند و کنترل ماشین تنها با تنظیم زاویه فرمان θ امکان‌پذیر است.
برای حل این مسئله از منطق فازی و نرم‌افزار **Fuzzy Logic Designer** در **MATLAB** استفاده شده است.

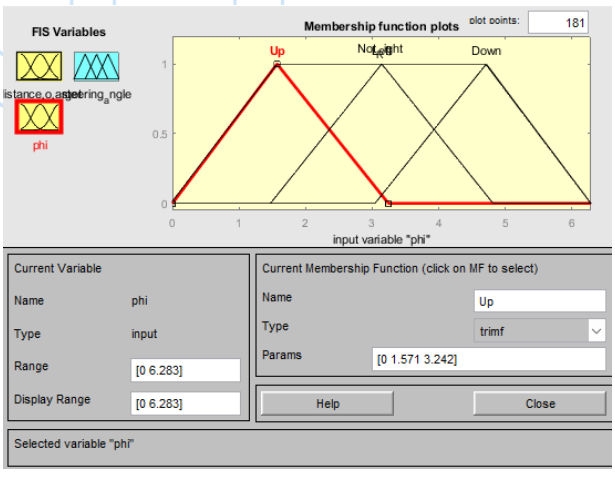
متغیرهای فازی و توابع تعلق

در این مسئله، ورودی‌ها و خروجی‌های سیستم فازی به صورت زیر تعریف شده‌اند:

1. ورودی‌ها

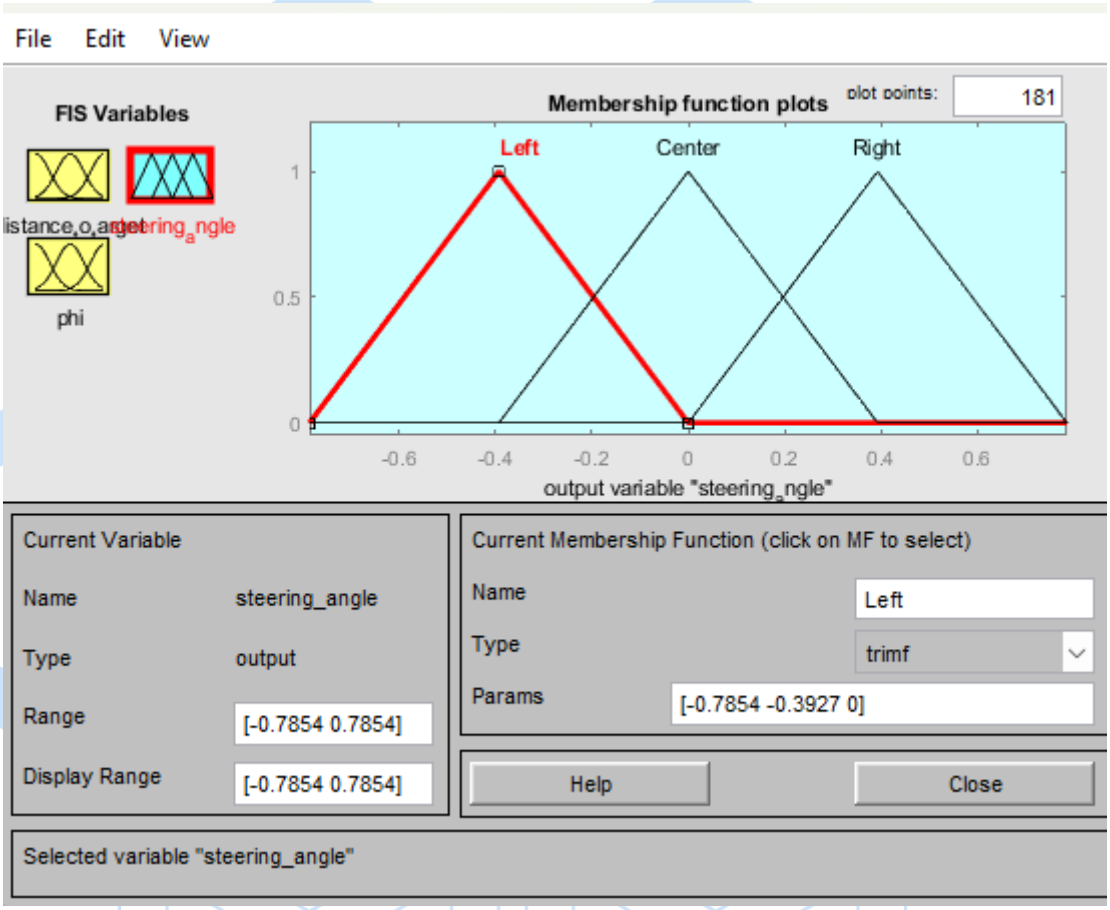
- **Φ** زاویه دوران ماشین با 4 تابع تعلق:
  - **Up:** ماشین در حالت بالا یا نزدیک به زاویه 90 درجه.
  - **Not_Right:** ماشین در زاویه غیر راست.
  - **Down:** ماشین در حالت پایین.
  - **Right:** ماشین در زاویه راست.
- **distance_to_target** فاصله از نقطه هدف با 2 تابع تعلق:
  - **Positive:** فاصله مثبت (ماشین در سمت راست نقطه هدف).
  - **Negative:** فاصله منفی (ماشین در سمت چپ نقطه هدف).





2.خروجی

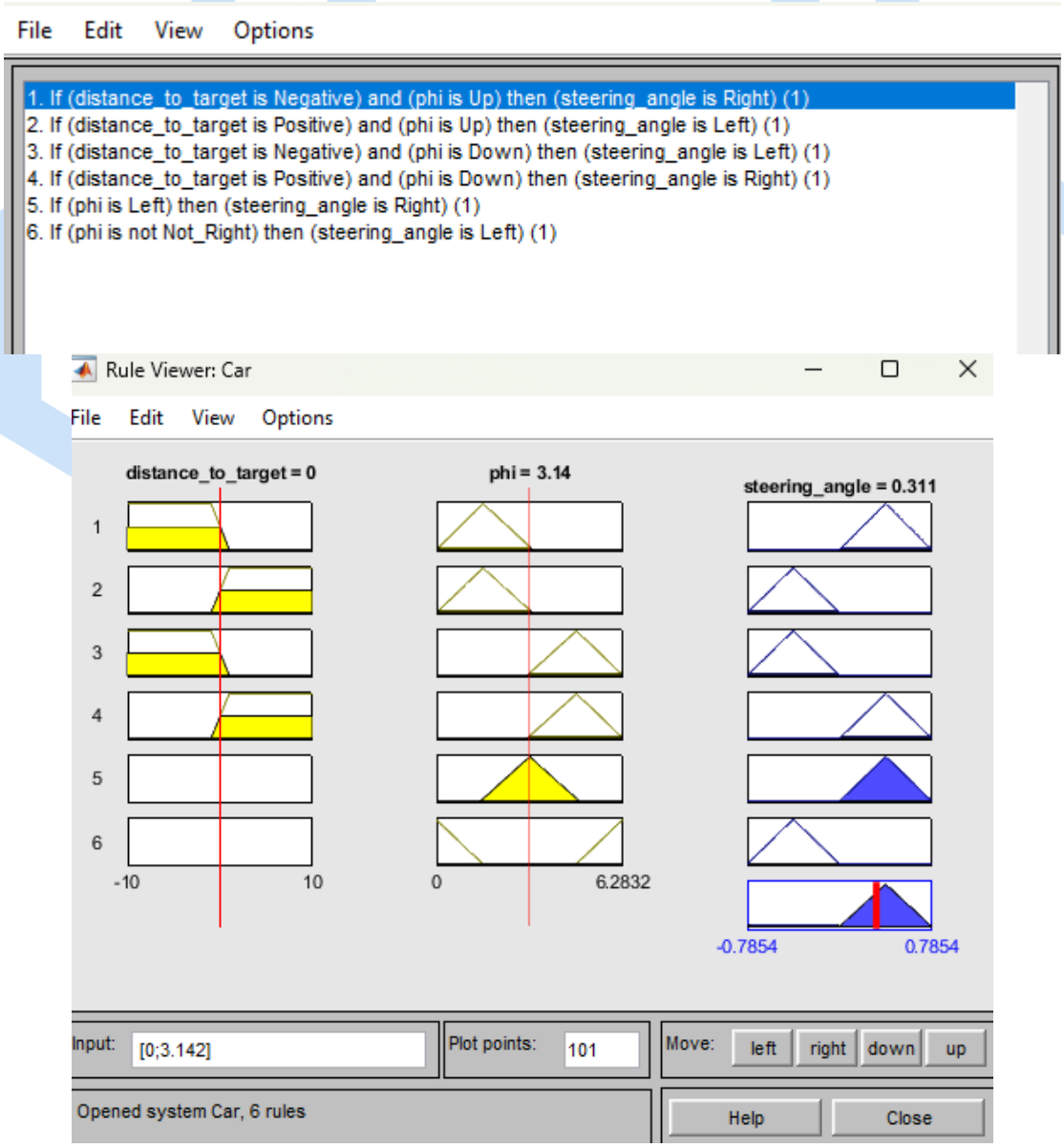
- **θ**زاویه فرمان با 3 تابع تعلق :
  - **Left:** فرمان به سمت چپ.
  - **Right:** فرمان به سمت راست.
  - **Center:** فرمان در حالت مستقیم.



قوانین فازی

6 قانون فازی برای کنترل ماشین طراحی شده‌اند. این قوانین به صورت منطقی تنظیم شده‌اند تا جهت حرکت ماشین با توجه به زاویه ϕو فاصله از نقطه هدف تعیین شود. قوانین به شرح زیر هستند:

1. اگر distance_to_targetمنفی باشد و ϕدر حالت Up باشد، زاویه فرمان θباید به سمت راست تنظیم شود.
2. اگر distance_to_targetمثبت باشد و ϕدر حالت Up باشد، زاویه فرمان θباید به سمت چپ تنظیم شود.
3. اگر distance_to_targetمنفی باشد و ϕدر حالت Down باشد، زاویه فرمان θباید به سمت چپ تنظیم شود.
4. اگر distance_to_targetمثبت باشد و ϕدر حالت Down باشد، زاویه فرمان θباید به سمت راست تنظیم شود.
5. اگر ϕدر حالت Left باشد، زاویه فرمان θباید به سمت راست تنظیم شود.
6. اگر ϕدر حالت Not_Right نباشد، زاویه فرمان θباید به سمت چپ تنظیم شود.



عملکرد سیستم

در اخر با نوشتن سیمولاتوری بر اساس فرمول های داده شده می توانیم عملکرد سیستم را ارزیابی کنیم و مشاهده میکنیم که سیستم با شرایط اولیه دشوار phi=60 و x=5 موفق به پارک می شود

نمودار بالا موقعت ماشین را با رنگ آبی و جهت ان را با فلش قرمز رنگ نشان می دهد که در احر موفق به پارک در نقطه x=10 و زاویه phi90 شده است فلش قرمز رنگ به خاطر zoo out دیده نمی شود

تصویر بزرگ تر:

پرسش سوم

کد در فایل soal3.m

Ball and Beam

1. بارگذاری داده‌ها

```
data = readtable('ballbeam.dat');
x = data{:, 1}; % First column as input
y = data{:, 2}; % Second column as output
```

فایل داده‌ی `ballbeam.dat` بارگذاری مس کنیم . ستون اول به‌عنوان ورودی (`x`) و ستون دوم به‌عنوان خروجی (`y`) در نظر گرفته شده است.

---

2. نرمال‌سازی داده‌ها

```
x = (x - min(x)) / (max(x) - min(x)); % Normalize input
y = (y - min(y)) / (max(y) - min(y)); % Normalize output
```

- **هدف :**مقادیر ورودی و خروجی به بازه [0, 1] نرمال‌سازی می کنیم تا مدل سریع‌تر و دقیق‌تر عمل کند.

---

3. تقسیم داده‌ها به آموزش و تست

```
rng(73); % Set seed for reproducibility
indices = randperm(length(x)); % Random permutation of indices
train_indices = indices(1:num_train); % Training indices
test_indices = indices(num_train+1:end); % Testing indices

% Create training and testing sets
x_train = x(train_indices);
y_train = y(train_indices);
x_test = x(test_indices);
y_test = y(test_indices);
```

هدف :داده‌ها به دو مجموعه تقسیم می‌شوند :

- **آموزشی** 80% **:**از داده‌ها برای آموزش مدل.
- **تستی** 20% **:**برای ارزیابی مدل.

4. تنظیمات و تولید مدل ANFIS

```
opt = genfisOptions('GridPartition');
opt.NumMembershipFunctions = 3; % Set number of membership functions
opt.InputMembershipFunctionType = 'gbellmf'; % Generalized bell-shaped MF

% Generate initial FIS
infis = genfis(x_train, y_train, opt);
```

تنظیمات اولیه سیستم استنتاج فازی تطبیقی (ANFIS) ایجاد می کنیم.

- **ویژگی‌ها :**
  - تعداد توابع عضویت3 :(NumMembershipFunctions)
  - نوع توابع عضویت: توابع زنگوله‌ای تعمیم‌یافته.(gbellmf)
  - دستور **genfis** مدل اولیه فازی ایجاد می‌کند.

---

5. آموزش مدل ANFIS

```
train_data = [x_train, y_train]; % Combine input and output for training
opt_train = anfisOptions('InitialFis', infis, 'EpochNumber', 100);
mynetwork = anfis(train_data, opt_train);
```

---

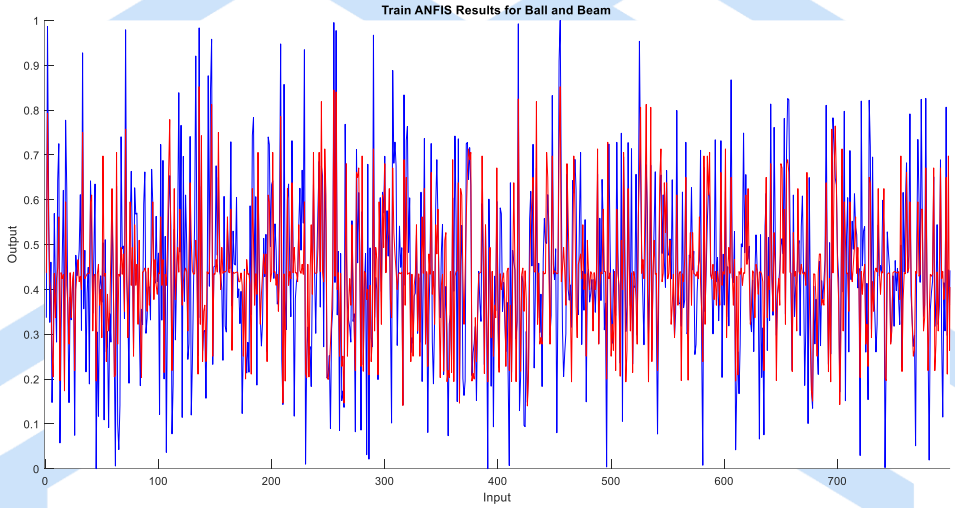
6. پیش‌بینی و ارزیابی مدل

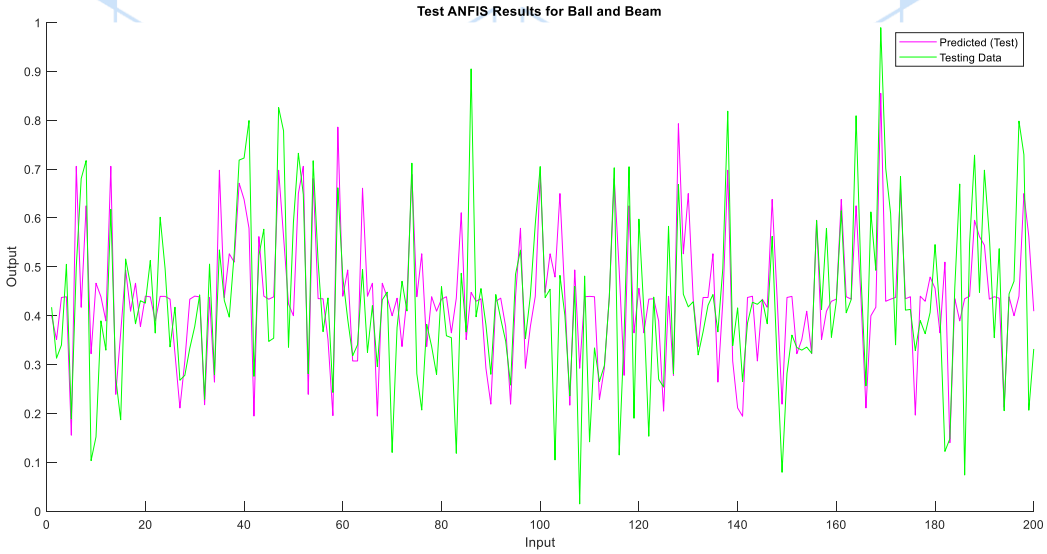
```
% Prediction
y_pred_train = evalfis(mynetwork, x_train); % Predict on training data
y_pred_test = evalfis(mynetwork, x_test);   % Predict on testing data

% Calculate performance metrics
mse_train = mean((y_train - y_pred_train).^2); % MSE for training
mse_test = mean((y_test - y_pred_test).^2);   % MSE for testing

% Display performance metrics
fprintf('MSE (Train): %.4f\n', mse_train);
fprintf('MSE (Test): %.4f\n', mse_test);

% Plot results
figure;
hold on;
plot( y_train, 'b', 'DisplayName', 'Training Data'); % Training data
plot(y_pred_train, 'r', 'DisplayName', 'Predicted (Train)'); % Predicted
train
xlabel('Input');
ylabel('Output');
title('Train ANFIS Results for Ball and Beam');

figure;
hold on;
plot(y_pred_test, 'm', 'DisplayName', 'Predicted (Test)');    % Predicted
test
plot(y_test, 'g', 'DisplayName', 'Testing Data');    % Testing data
legend;
```

```
xlabel('Input');
ylabel('Output');
title('Test ANFIS Results for Ball and Beam');
hold off;
```

MSE (Train): 0.0184

MSE (Test): 0.0170





پرسش چهارم

شناسایی یک سیستم غیرخطی با فازی و گرادیان کاهشی

کد در فایل soal4.ipynb

معادله دیفرانسیل زیر تعریف شده است

$$y_{k+1} = 0.3y_k + 0.6y_{k-1} + g[u_k]$$

تابع غیرخطی به صورت زیر تعریف می‌شود:

$$g[u] = 0.6\sin(\pi u) + 0.3\sin(3\pi u) + 0.1\sin(5\pi u)$$

هدف ما استفاده از یک مدل فازی برای تقریب تابع غیرخطی بر اساس ورودی و خروجی است. این مدل فازی با استفاده از روش کمترین مربعات انرژی (LSE) آموزش داده می‌شود .

اول داده‌های ورودی و خروجی را با محاسبه دینامیک سیستم و جداسازی اجزای خطی و غیرخطی تولید می‌کنیم

کد شبیه‌سازی:

```
def simulation(N):
        u = np.sin(np.linspace(0, 2 * np.pi, N))  # Input signal u(k)
        y_true = np.zeros(N)
        output = np.zeros(N)
        y_k = 0  # Initial value of y(k)
        y_k_1 = 0  # Initial value of y(k-1)
        # Compute the true output y(k+1) based on the system dynamics
        for k in range(2, N):
            y_true[k] = (
                0.3 * y_k
                + 0.6 * y_k_1
                + 0.6 * np.sin(np.pi * u[k])
                + 0.3 * np.sin(3 * np.pi * u[k])
                + 0.1 * np.sin(5 * np.pi * u[k])
            )
            # Update the previous values
            output[k]=y_true[k] - 0.3 * y_k - 0.6 * y_k_1
            y_k_1 = y_k
            y_k = y_true[k]
        return u,output
```

این کد دینامیک سیستم را محاسبه کرده و بخش غیرخطی را با تفریق سهم خطی استخراج می‌کند.

مدل سیستم فازی

سپس با درست کردن لیستی از centerها و انحراف از معیار ها دروقع سیستم فازیی را تشکیل دادیم که با استفاده از توابع عضویت گوسی تعریف شده. و حال با استفاده از کد زیر سیستم ما نسبت به ورودی داده شده خروجی را حساب می کند.

کد سیستم فازی:

```
def fuzzy_system(x, centers, sigmas, y):
    # Compute the z_i values for each input using Gaussian membership
functions
    z = np.array([gaussian(x, c, sigma) for c, sigma in zip(centers,
sigmas)])

    # Compute b as the sum of the z_i's
    b = np.sum(z)

    # Compute a as the weighted sum of centers (using z_i as weights)
    a = np.sum(z * y)

    # Compute the output of the fuzzy system as f(x) = a / b
    if b != 0:
        f_x = a / b
    else:
        f_x = 0  # In case the sum of z_i's is zero (just for safety)

    return f_x, z, b, a
```

آموزش مدل

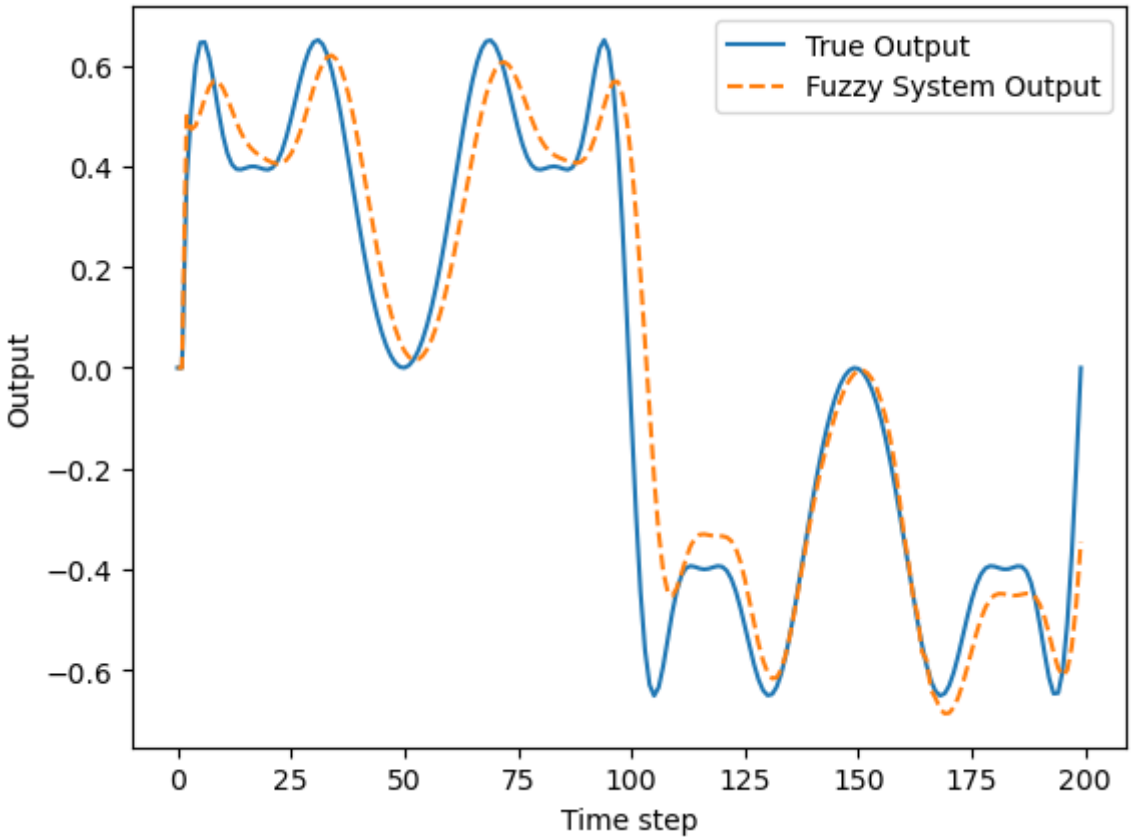
پارامترهای سیستم فازی (مراکز، پهناها و وزن‌ها) با استفاده از گرادیان نزولی بهینه‌سازی می‌شوند. تابع خطا میانگین مربعات خطا بین خروجی پیش‌بینی‌شده و واقعی است.

کد گرادیان نزولی:

```
# Gradient Descent for the centers and spreads
def gradient_descent_update(x, y_true, centers, sigmas, y, alpha, M, z, b,
a):
    # Predicted output from fuzzy system
    y_pred = a / b

    # Compute the error (y_true is the actual output)
    error = y_true - y_pred

    # Update the centers (c_i) and spreads (sigma_i) using gradient descent
    for i in range(M):
        # Gradient with respect to the center c_i
        grad_y = (-error / b) * z[i]
        grad_c = (
            ((-error * z[i]) / b)
            * (y[i] - y_pred)
            * z[i]
            * (2 * (x - centers[i]) / sigmas[i] ** 2)
        )
        # Gradient with respect to the spread sigma_i
        grad_sigma = (
            ((-error * z[i]) / b)
            * (y[i] - y_pred)
            * z[i]
            * (2 * (x - centers[i]) ** 2 / sigmas[i] ** 3)
        )

        # Update the parameters using the gradients
        y[i] -= alpha * grad_y
        centers[i] -= alpha * grad_c
        sigmas[i] -= alpha * grad_sigma

    return centers, sigmas,
```

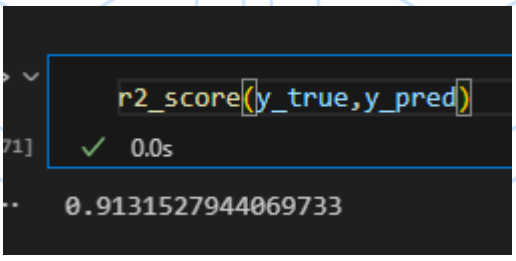
نتایج

سیستم فازی با موفقیت بخش غیرخطی را تقریب زد. عملکرد آموزش و آزمایش مدل با استفاده از معیارهای زیر ارزیابی شد:



نتیجه‌گیری

مدل فازی که با استفاده از گرادیان نزولی آموزش دیده است، به طور مؤثری بخش غیرخطی سیستم را شناسایی کرد. کارهای آینده می‌تواند شامل بررسی تکنیک‌های بهینه‌سازی پیشرفته‌تر یا استفاده از توابع عضویت پیچیده‌تر برای افزایش دقت باشد.



پرسش پنجم

نتایج مدل‌های RBF و ANFIS

کد در فایل soal5.ipynb

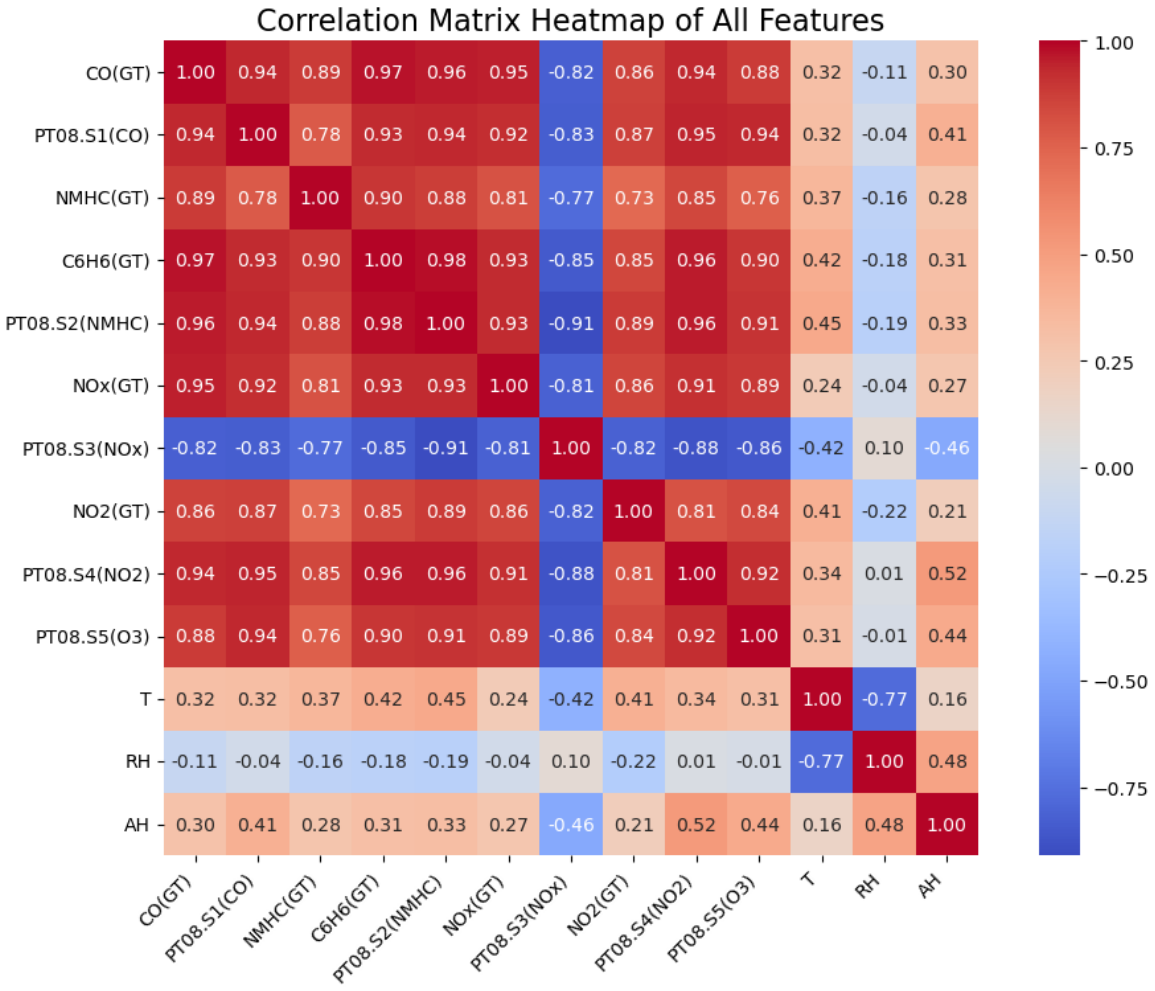
در این سوال، دو مدل مختلف شامل شبکه عصبی با پایه‌های شعاعی (RBF) و سیستم استنتاج فازی تطبیقی (ANFIS) برای مدل‌سازی داده‌های پیچیده و غیرخطی استفاده شده است. هدف از این تحلیل مقایسه عملکرد این دو مدل و تعیین مدل بهتر با توجه به معیارهای میانگین مربعات خطا (MSE) و ضریب تعیین (R^2) بوده است.

هدف رگرسیون در این مسئله، پیش‌بینی مقدار `PT08.S1(CO)` است که نمایانگر پاسخ حسگر اکسید قلع (tin oxide) **می‌باشد.**

برای مقایسه این دو مدل همانطور که در صورت سوال ذکر شده بود اول دیتا را تقسیم می کنیم و باید به این نکته توجه داشته باشیم که طبق اطلاعات موجود در لینک دیتاست مقدار زیادی missing data وجود دارد گفته شده اعداد 200- نشانه عدم وجود دیتا می باشد پس ای دیتا ها نیز باید حذف شوند

```
# Load the dataset
file_path = "AirQualityUCI.xlsx"  # Replace with the actual file path

df = pd.read_excel(file_path)

# Replace -200 with NaN for better handling of missing data

df.replace(-200, pd.NA, inplace=True)

# Drop rows with missing data (originally indicated by -200)

df.dropna(inplace=True)

# Reset the index after dropping rows

df.reset_index(drop=True, inplace=True)

# Display the cleaned DataFrame

df.head()
```

به علت تعداد زیاد feature ها ما چند تا از مرتبط ترین ویژگی ها را برای trainانتخاب کردیم



Highly Correlated Features (Positive Correlation):

CO(GT) with a correlation of 0.94

C6H6(GT) with a correlation of 0.93

PT08.S2(NMHC) with a correlation of 0.94

PT08.S4(NO2) with a correlation of 0.95

PT08.S5(O3) with a correlation of 0.94

پس ما از این 3 ویژگی استفاده می کنیم

```
CO(GT), C6H6(GT), PT08.S2(NMHC)
```

نتایج مدل RBF

RBF Model - Mean Squared Error: 7853.848859535396

RBF Model - R-Squared: 0.861146314363854

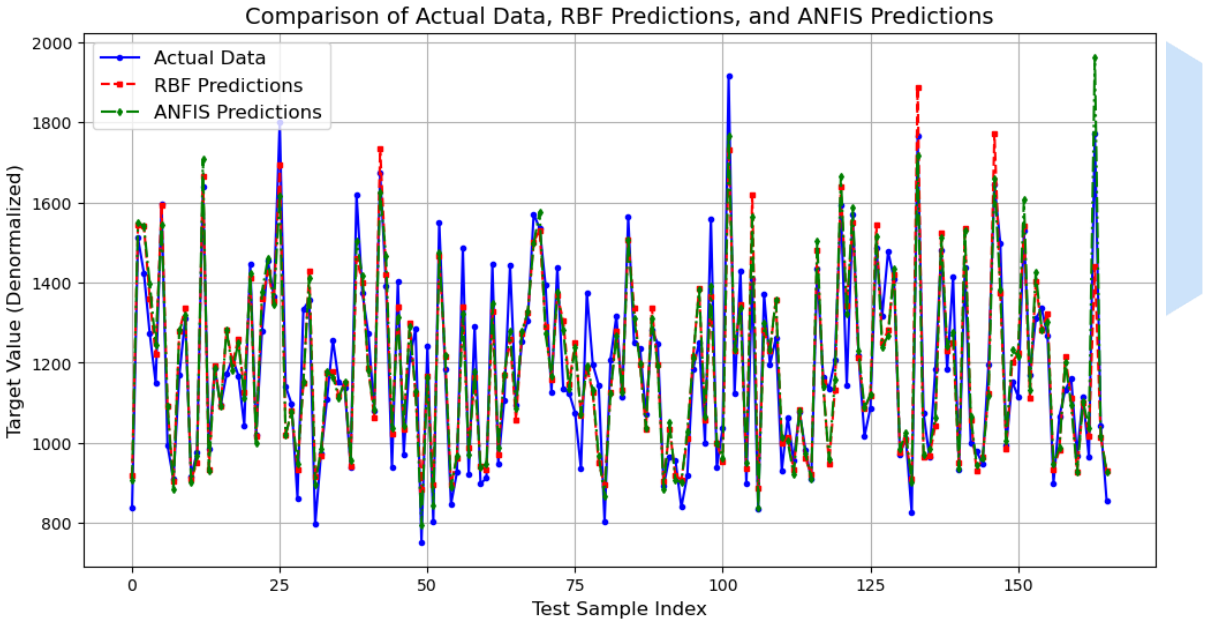
این مقادیر نشان‌دهنده عملکرد متوسط مدل RBF در پیش‌بینی داده‌های تست می‌باشد. با توجه به MSE بالا و R^2 نسبتاً پایین، این مدل توانایی محدودی در مدل‌سازی دقیق داده‌ها نشان داده است.

نتایج مدل ANFIS

ANFIS Model - Mean Squared Error: 6819.833311638766

ANFIS Model - R-Squared: 0.8794273982500316

مدل ANFIS با MSE پایین‌تر و R^2 بالاتر عملکرد بسیار بهتری در پیش‌بینی داده‌ها داشته است. این مقادیر نشان می‌دهد که مدل ANFIS توانسته است روابط پیچیده بین متغیرها را بهتر یاد بگیرد و داده‌های تست را با دقت بیشتری پیش‌بینی کند.



نتیجه‌گیری

مدل ANFIS در این پروژه عملکرد بهتری نسبت به مدل RBF نشان داده است. دلیل اصلی این برتری در توانایی ANFIS در مدل‌سازی روابط غیرخطی پیچیده و استفاده از سیستم استنتاج فازی است که امکان درک بهتری از عدم قطعیت و انعطاف‌پذیری در داده‌ها را فراهم می‌کند. در مقابل، مدل RBF به دلیل محدودیت‌های موجود در نحوه تعریف خوشه‌ها و انتشار شعاعی، نتوانسته است همان دقت را ارائه

دهد. پیشنهاد می‌شود برای مسائل مشابه که شامل داده‌های پیچیده و غیرخطی هستند، از مدل ANFIS استفاده شود.

پایان