به نام خدا

نویسنده:

محمدعلی مجتهدسلیمانی

♦ توضیحات پیاده سازی سوال ۵:

*ابتدا توضیحات پیاده سازی کد ها را ارائه میکنم و بعد از آن به موارد خواسته شده در سوال پاسخ میدهم.

• بخش اضافه کردن کتابخانه:

```
import numpy as np
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
```

با توجه به اینکه بنده از IDE، Pycharm استفاده میکنم نمیتوانم از خود کتابخانه matplotlib بصورت مستقیم استفاده بکنم و مجبور شدم backend را برای نمایش plotها بر روی TKAgg قرار بدهم.

• بخش تنظیم ورودی:

```
X = np.array([1, 3, 5, 7, 9])
y = np.array([1.2, 2.4, 2.9, 4.5, 5.1])
centers = np.array([2, 6, 8])
sigma_values = np.array([-0.6, 0.7, 1.2])
```

با کمک کتابخانه numpy دو بردار ورودی x و target که y هست را تعریف کردیم. مراکز (center) های RBF را مشخص کردیم. سیگما را که همان width تابع ما خواهد بود نیز تعریف کردیم.

• بخش تابع Gaussian RBF:

```
def gaussian_rbf(x, center, sigma):
    if sigma <= 0:
        sigma = abs(sigma)
    return np.exp(-(x - center) ** 2 / (2 * sigma ** 2))
```

در این بخش تابع gaussian را پیاده سازی کردیم که بردار ورودی و مراکز و سیگما را به عنوان ورودی تابع میگیرد، بعد از این مرحله مطمئن میشویم سیگما حتما عددی مثبت باشد. در نهایت مقدار تابع gaussian را برمیگردانیم.

• بخش تابع Polynomial RBF:

```
def polynomial_rbf(x, center, sigma):
    return (1 + (x - center) ** 2 / (2 * sigma ** 2))
```

در این بخش تابع polynomial را پیاده سازی کردیم که شبیه تابع gaussian است اما از فرمول متفاوتی استفاده میکند.

• بخش محاسبه خروجی RBF:

```
def calculate_rbf_outputs(x, centers, sigma, rbf_type='gaussian'):
    outputs = []
    for center in centers:
        if rbf_type == 'gaussian':
            output = gaussian_rbf(x, center, sigma)
        else:
            output = polynomial_rbf(x, center, sigma)
        outputs.append(output)
    return np.array(outputs)
```

در این بخش خروجی RBF را برای تمام مراکز محاسبه میکنیم. در این بخش میتوانیم بین تابع Gaussian و polynomial جابجا شویم. در نهایت یک لیست برای هر مرکز برمیگرداند.

• بخش محاسبه خطا MSE:

```
def calculate_mse(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)
```

این بخش خطای MSE را محاسبه میکند که بین مقدار Y واقعی و Y خروجی تابع ما هست که همان پیشبینی ما هست.

• بخش نمایش خروجی برای بخش الف

```
print("Part A: RBF Outputs Comparison")
print("\nGaussian RBF outputs:")
for sigma in sigma_values:
    print(f"\nσ = {sigma}")
    print("X\tGaussian RBF outputs for each center")
    print("-" * 50)
    for x_val in X:
        outputs = calculate_rbf_outputs(x_val, centers, sigma)
        print(f"{x_val}\t{outputs}")

print("\nPolynomial RBF outputs:")
for sigma in sigma_values:
    print(f"\nσ = {sigma}")
    print("X\tPolynomial RBF outputs for each center")
    print("-" * 50)
    for x_val in X:
        outputs = calculate_rbf_outputs(x_val, centers, sigma, 'polynomial')
        print(f"{x_val}\t{outputs}")
```

در این بخش خروجی ها را برای هر ۲ تابع نشان میدهیم با مقادیر متفاوت سیگما و مراکز.

• بخش تحلیل خطا برای بخش ب

```python
print("\nPart B: MSE Analysis")
print("\nGaussian RBF MSE for different σ values:")
for sigma in sigma_values:
    predictions = []
    for x_val in X:
        rbf_outputs = calculate_rbf_outputs(x_val, centers, sigma)
        pred = np.mean(rbf_outputs) * 5
        predictions.append(pred)

    mse = calculate_mse(y, predictions)
    print(f"σ = {sigma}: MSE = {mse:.4f}")
```

در این بخش MSE را برای مقادیر مختلف سیگما محاسبه میکند. از میانگین RBF ضربدر ۵ برای پیشبینی استفاده میکند.

- **بخش نمایش تصویری تابع Gaussian:**

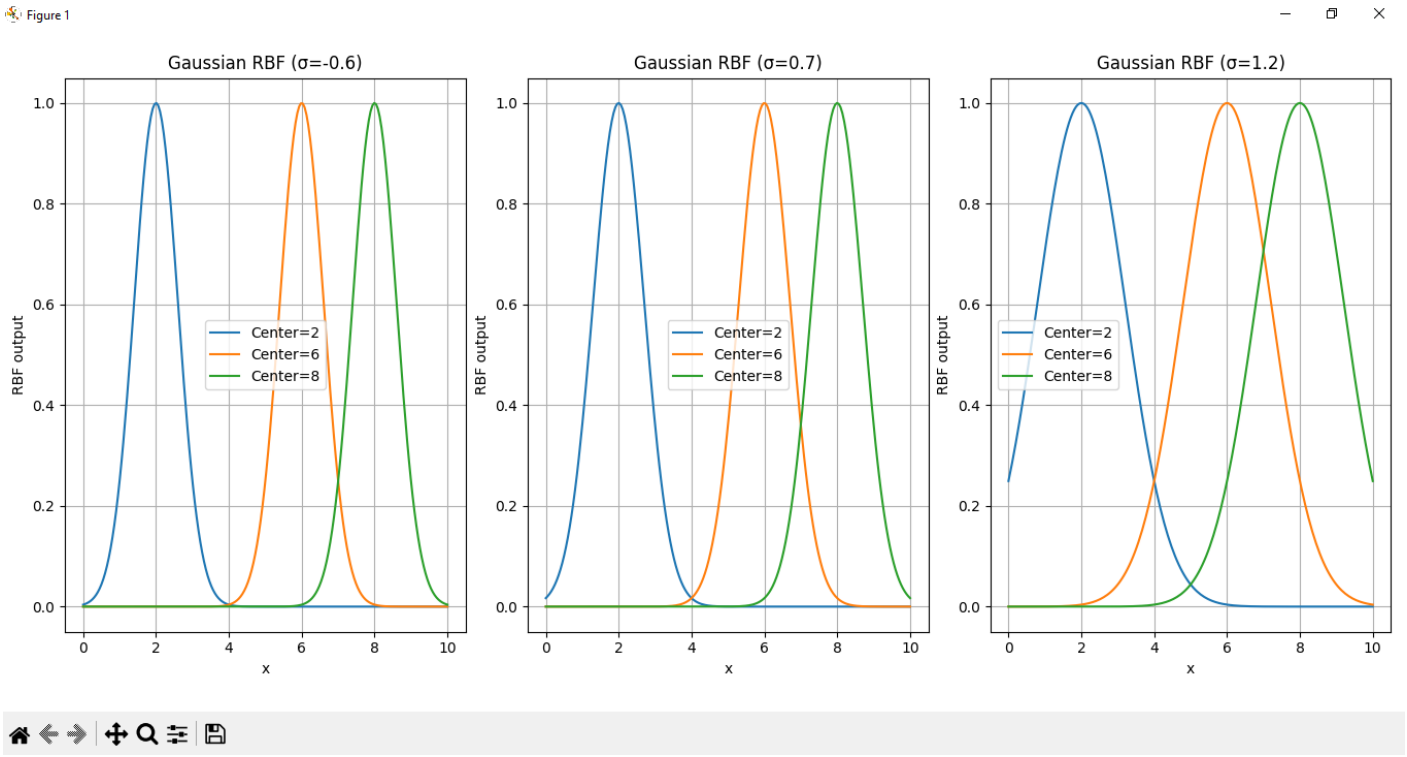
```
plt.figure(figsize=(15, 5))
x_plot = np.linspace(0, 10, 200)

for i, sigma in enumerate(sigma_values):
    plt.subplot(1, 3, i + 1)
    for center in centers:
        y_rbf = gaussian_rbf(x_plot, center, sigma)
        plt.plot(x_plot, y_rbf, label=f'Center={center}')
    plt.title(f'Gaussian RBF (σ={sigma})')
    plt.xlabel('x')
    plt.ylabel('RBF output')
    plt.legend()
    plt.grid(True)

plt.tight_layout()
plt.show()
```

در این بخش تابع Gaussian را برای ۳ مقدار مختلف سیگما نشان میدهیم. و خروجی ها را با توجه به مراکز نمایش دادیم.

*خروجی این بخش در صفحه بعد وجود دارد.



♦ بخش الف

• خروجی تابع Gaussian با توجه به نقاط:

```
σ = -0.6
X   Gaussian RBF outputs for each center
-----------------------------------------------------
1   [2.49352209e-01 8.32396968e-16 2.77873902e-30]
3   [2.49352209e-01 3.72665317e-06 8.32396968e-16]
5   [3.72665317e-06 2.49352209e-01 3.72665317e-06]
7   [8.32396968e-16 2.49352209e-01 2.49352209e-01]
9   [2.77873902e-30 3.72665317e-06 2.49352209e-01]
```

خروجی برای سیگما 0.6-

```
σ = 0.7
X   Gaussian RBF outputs for each center
-------------------------------------------------
1   [3.60447789e-01 8.33794709e-12 1.92874985e-22]
3   [3.60447789e-01 1.02702565e-04 8.33794709e-12]
5   [1.02702565e-04 3.60447789e-01 1.02702565e-04]
7   [8.33794709e-12 3.60447789e-01 3.60447789e-01]
9   [1.92874985e-22 1.02702565e-04 3.60447789e-01]
```

خروجی برای سیگما 0.7

```
σ = 1.2
X   Gaussian RBF outputs for each center
-------------------------------------------------
1   [7.06648278e-01 1.69856677e-04 4.08283604e-08]
3   [7.06648278e-01 4.39369336e-02 1.69856677e-04]
5   [0.04393693 0.70664828 0.04393693]
7   [1.69856677e-04 7.06648278e-01 7.06648278e-01]
9   [4.08283604e-08 4.39369336e-02 7.06648278e-01]
```

خروجی برای سیگما 1.2

- خروجی تابع **Polynomial** با توجه به نقاط:

```
σ = -0.6
X   Polynomial RBF outputs for each center
-------------------------------------------------
1   [ 2.38888889 35.72222222 69.05555556]
3   [ 2.38888889 13.5        35.72222222]
5   [13.5         2.38888889 13.5       ]
7   [35.72222222  2.38888889  2.38888889]
9   [69.05555556 13.5         2.38888889]
```

خروجی برای سیگما -0.6

```
σ = 0.7
X   Polynomial RBF outputs for each center
-------------------------------------------------
1   [ 2.02040816 26.51020408 51.        ]
3   [ 2.02040816 10.18367347 26.51020408]
5   [10.18367347  2.02040816 10.18367347]
7   [26.51020408  2.02040816  2.02040816]
9   [51.         10.18367347  2.02040816]
```

خروجی برای سیگما 0.7

```
σ = 1.2
X   Polynomial RBF outputs for each center
-------------------------------------------------
1   [ 1.34722222  9.68055556 18.01388889]
3   [1.34722222 4.125      9.68055556]
5   [4.125      1.34722222 4.125     ]
7   [9.68055556 1.34722222 1.34722222]
9   [18.01388889  4.125       1.34722222]
```

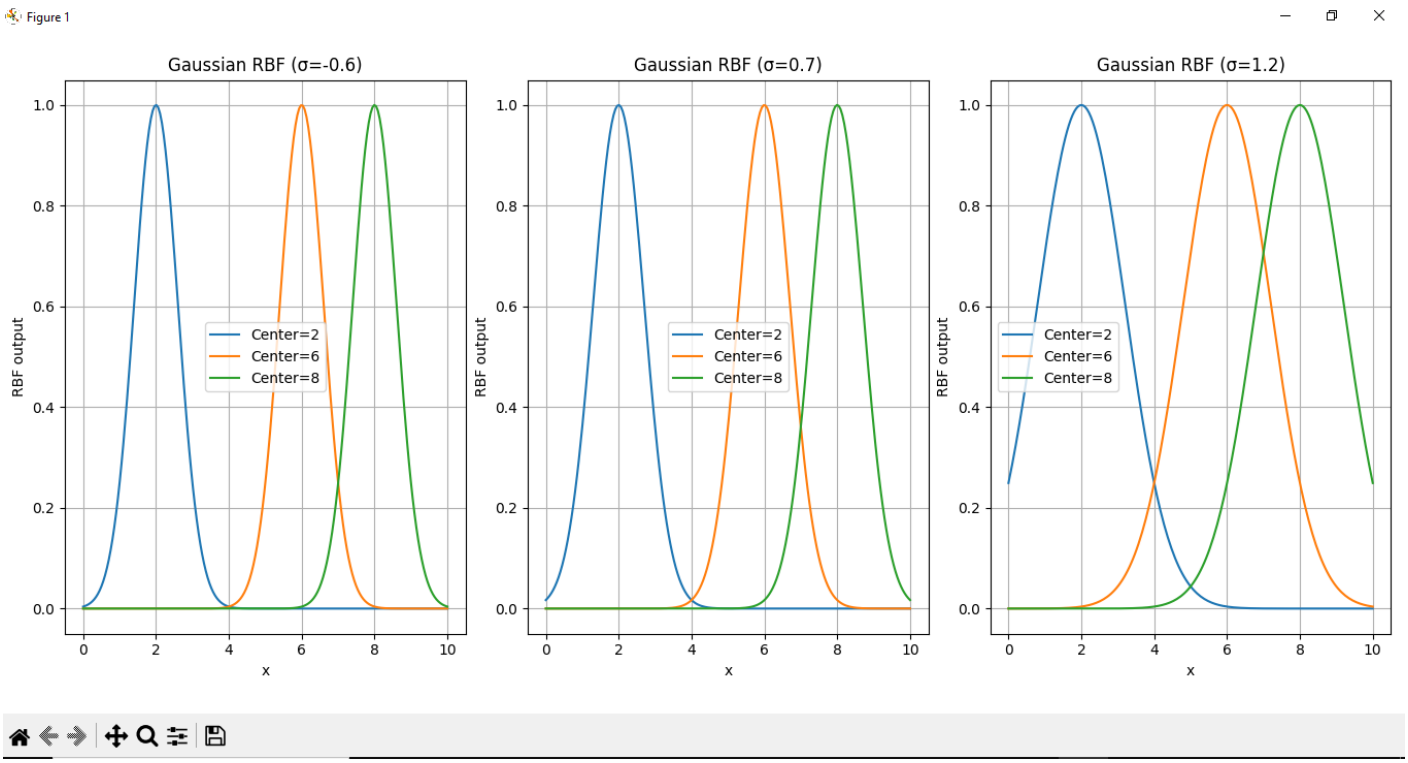
خروجی برای سیگما 1.2

• بخش ب

```
Gaussian RBF MSE for different σ values:
σ = -0.6: MSE = 9.2259
σ = 0.7: MSE = 8.0005
σ = 1.2: MSE = 4.6432
```

بخش محاسبه خطا با ۳ مقدار سیگما مختلف

همچنین نمایش تصویری این مورد در صفحات قبلی آمده است. (در بخش آخر توضیح پیاده سازی کد)



• نکات:

برای سیگما 0.6- به سبب استفاده از مقادیر مثبت مانند سیگما 0.6 رفتار میکند اما پاسخ باریک تری یا محدود تری تولید میکند.

برای سیگما 0.7 تاثیری با پراکندگی متوسط در اطراف مرکز ایجاد کرده است.

برای سیگما 1.2 پاسخ های وسیع تر و با همپوشانی بیشتر تولید میکند.

هر چه مقدار سیگما بزرگتر باشد بنظر میرسد پاسخ های روان تر و نرم تری با همپوشانی بیشتر تولید میکند. هر چه این مقدار کمتر باشد پاسخ های محلی تر و قله های تیز ایجاد میکند. مقادیر منفی از طریق قدر مطلق مدیریت میشوند. موقعیت مراکز هم نسبت به نقاط بر پیشبینی تاثیر میگذارد.