

به نام خدا

نویسنده:

محمد علی مجتهد سلیمانی

توضیحات پیاده سازی سوال ۷:

• بخش اضافه کردن کتابخانه:

```
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from scipy.spatial.distance import cdist
```

با توجه به اینکه بنده از IDE، pycharm استفاده میکنیم نمیتوانستیم از خود کتابخانه matplotlib بصورت مستقیم استفاده کنم در نتیجه مجبور شدم خط دوم را هم وارد کنم.

طبق کد، از کتابخانه sklearn به ۲ منظور استفاده کردیم: ۱. ابتدا برای استفاده از تابع MSE برای محاسبه خطا خروجی واقعی نسبت به خروجی desired. ۲. Linear regression برای fit کردن مدل.

از کتابخانه `scipy` بخاطر استفاده از تابع `cdist` استفاده کردیم تا فاصله داده ورودی نسبت به `center` های RBF استفاده کردیم.

• بخش تولید داده:

```
N = 20
x = np.linspace(0, 2, N).reshape(-1, 1)

def main_function(x):
    return 0.3 * np.cos(3 * np.pi * x) + 0.7 * np.sin(np.pi * x) + 0.5

out_y = main_function(x)
rand_noise = np.random.normal(0, np.sqrt(0.01), N).reshape(-1, 1)
y = out_y + rand_noise
```

$N=20$ برای تعداد نقاط ورودی که می‌خواهیم تولید کنیم. از `linspace` که یک تابع `numpy` است برای ساختن یک بردار مقادیر بین ۰ تا ۲ انتخاب کردیم که به تعداد ۲۰ تا نمونه تولید میکند. تابع `main` همان تابعی است که سوال از ما خواسته است تا استفاده بکنیم این تابع ارتباط بین x و y را بدون در نظر گرفتن نویز می‌سازد. `Out_y` خروجی واقعی که ما بدون در نظر گرفتن نویز خواهد بود. `Rand_noise` با استفاده از `numpy` یک نویز تصادفی با میانگین صفر تولید میکند که باعث تصادفی شدن y خواهد شد. Y در نهایت حاصل جمع خروجی واقعی با نویز تولید شده در خط قبلی خواهد بود.

• بخش شبکه RBF:

```
def rbf_model(x, centers, variances):  
    distances = cdist(x, centers)  
    rbf_values = np.exp(-distances ** 2 / (2 * variances))  
    return rbf_values
```

این تابع جایی است که برای هر نقطه x ، RBF را محاسبه میکند. Distance فاصله اقلیدوسی را بین هر center و نقطه x را محاسبه میکند. Rbf_values تابع gaussian را به هر distance اعمال میکند و distance را با یک واریانس به یک مقداری تبدیل میکند که این مقدار به فاصله تا center ها گفته میشود. در واقع این مقدار نشان میدهد که به چه شدتی هر center تاثیر روی ورودی گذاشته است. در نهایت return را داریم که ماتریس RBF برای همه نقاط x را محاسبه میکند.

• بخش ارزیابی شبکه RBF:

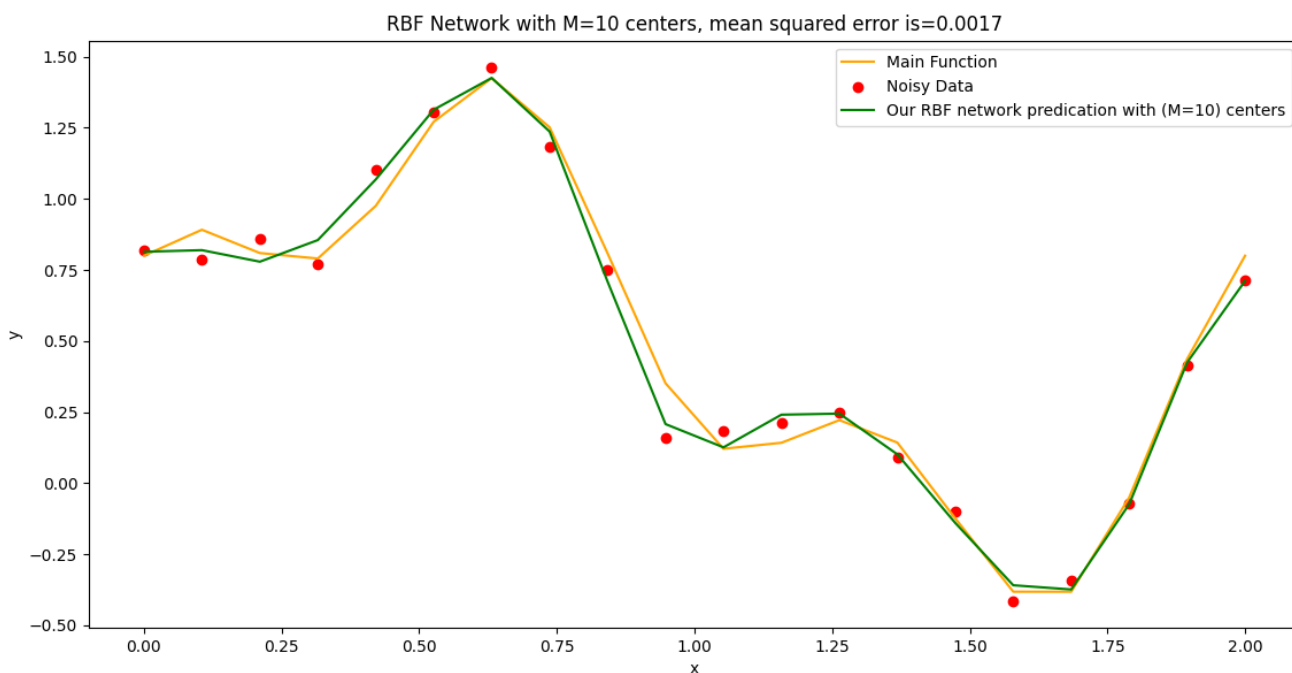
```
def fit_rbf_network(x, y, M):  
    centers = np.random.choice(x.flatten(), M).reshape(-1, 1)  
    variances = np.random.rand(M)  
  
    rbf_features = rbf_model(x, centers, variances)  
  
    model = LinearRegression()  
    model.fit(rbf_features, y)  
    y_pred = model.predict(rbf_features)  
  
    mse = mean_squared_error(y, y_pred)  
    return y_pred, mse
```

با استفاده از یک مقدار M مشخص تعداد center ها را مشخص میکنیم. Centers به صورت تصادفی به اندازه m تا از x برای center ها انتخاب میکند. Center محل RBF را مشخص میکنند. Variance به

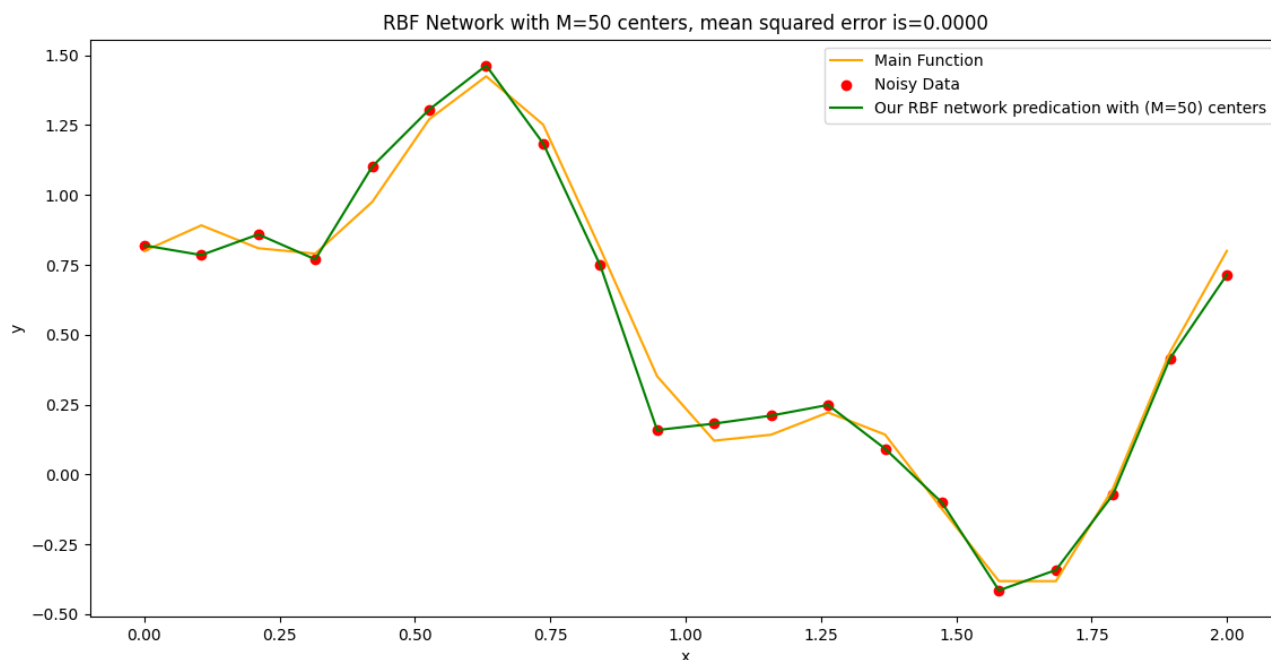
صورت تصادفی m تا واریانس تصادفی بین مقادیر ۰ و ۱ تولید میکند. هر $center$ واریانس مخصوص به خودش را دارد که به همگرایی زودتر RBF با $distance$ کمک میکند. $Rbf_feature$ جایی هست که مقدار RBF را برای همه نقاط X حساب میکنیم (با توجه به $center$ و واریانس مخصوص به آن). اینجا $feature$ $space$ که یک بعد بالاتر است ساخته میشود که هر $feature$ یک خروجی RBF هستند. $model$ یک مدل با $linear regression$ ساخته ایم. بعد مدل را با RBF آموزش میدهم تا مقدار y را تقریب بزنیم. Y_pred مقدار $prediction$ تابع خواهد بود. MSE مقدار خطا را تخمین میزند که به فاصله بین خروجی $desired$ و خروجی پیشبینی شده ما خواهد بود. و در نهایت این ۲ مقدار را $return$ کرده ایم.

در بخش آخر خروجی را $plot$ کرده ایم به کمک $matplotlib$.

• خروجی به ازای مقدار $M=10$:



• خروجی به ازای مقدار $m=50$:



• تحلیل تغییر عملکرد شبکه:

وقتی که مقدار m برابر با ۱۰ بود مدل فضای کافی برای ذخیره تمام جزئیات **target** نداشت و تمام الگو را مدل نمیتوانست یاد بگیرد با توجه به اینکه در بعضی قسمت ها تغییرات زیاد بود اما وقتی m برابر با ۵۰ شد مدل فضای کافی برای ذخیره تمام جزئیات داشت و مدل به اصطلاح **overfit** شد و حتی به موارد **noise** هم سازگار شد که این امر تاثیر بسیار منفی در تعمیم پذیری مدل ما دارد نسبت به داده های جدیدی که تا الان مشاهده نکرده است. از طرف دیگر وقتی تعداد m بالاتر رفته است مدل حتی به نویز های بسیار کوچک هم بسیار حساسیت نشان داد که باعث ناهمواری در روند شد.