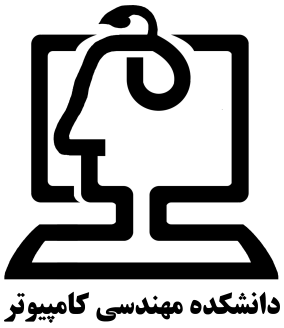
به نام خدا



دانشگاه صنعتی امیرکبیر
دانشکده مهندسی کامپیوتر

استاد درس: دکتر امین ناظرفرد

پاییز ۱۳۹۹

درس یادگیری ماشین

سروش مهدی
شماره دانشجویی: ۹۹۱۳۱۰۵۰

سری ۱ تمرین‌ها

فهرست مطالب

۱ سوالات تشریحی

۱.۱ تعاریف

۱.۱.۱ یادگیری نظارتی

در یادگیری نظارتی برای هر یک از نمونه ها در مجموعه داده های یادگیری خروجی صریح و یا برچسب مربوط به ان نمونه را داریم. در واقع الگوریتم از داده های برچسب گذاری شده برای عمل پیش بینی یک تابع استنباط میکند. خروجی مورد نظر هر یک از نمونه ها را یک ناظر مشخص کرده است.

۲.۱.۱ یادگیری نیمه نظارتی

یک رویکرد یادگیری ماشین میباشد که مقدار کمی از داده ها برچسب دارد و مقدار زیادی از ان ها بدون برچسب میباشد و در واقع خروجی برای تعداد محدودی از داده ها مشخص است . همانطورکه از اسم این روش پیداست روشی بین روش های یادگیری نظارتی و بدون نظارت میباشد.

۳.۱.۱ یادگیری بدون نظارت

در این رویکرد ما با داده های بدون برچسب رو به رو هستیم و الگوها و ساختار هایی جدید را از این داده ها بدست می اوریم.

۴.۱.۱ یادگیری تقویتی

یک رویکرد یادگیری ماشین است که در ان برای هر داده خروجی صریح ان را نداریم ولی در عوض یک مجموعه از خروجی های ممکن و یک معیار اندازه گیری برای میزان خوب بودن یا بد بودن این خروجی ها داریم. در واقع در این رویکرد عامل با سعی و خطا یاد میگیرد که چه عمل هایی برای بیشینه کردن تابع جایزه مناسب تر است.

۵.۱.۱ یادگیری عمیق

یک نوع یادگیری ماشین است که در ان برای پیدا کردن `feature` های پیچیده و یادگیری از شبکه های مصنوعی عمیق استفاده میکنیم همچنین در این روش رویکرد ما میتواند نظارتی یا نیمه نظارتی یا بدون نظارت باشد.

۶.۱.۱ رگرسیون

یک روش اماری است که در ان از فرایند های اماری برای تقریب رابطه بین یک متغیر وابسته و یک یا چندین متغیر مستقل استفاده میکنیم.

۷.۱.۱ یادگیری برخط

در این حالت ما جریانی از داده ها داریم که بصورت برخط وارد الگوریتم میشوند. در واقع ما تمام داده ها را از ابتدا نداریم و داده ها ممکن است به تدریج وارد الگوریتم شوند و الگوریتم باید مدل را نسبت به این داده ها به روزرسانی کند. همچنین از این روش میتوانیم هنگامی که محدودیت حافظه یا محدودیت محاسباتی داریم نیز استفاده کنیم.

۸.۱.۱ یادگیری فعال

یک مورد از روش های یادگیری ماشین است که در ان الگوریتم یادگیری میتواند به صورت فعالانه از ناظر سوال بپرسد و برای هر داده برچسب یا اطلاعات مورد نظر راجع به ان نمونه داده را به دست اورد. الگوریتم باید در مورد داده هایی سوال بپرسد که ارزش اطلاعات ان بالا باشد.

۹.۱.۱ دسته بندی

یک نوع یادگیری با نظارت است که در ان برای هر داده ورودی یک برچسب به عنوان خروجی مشخص میکنیم . در این حالت خروجی گسسته است.

۱۰.۱.۱ خوشه بندی

یک نوع یادگیری بدون نظارت است که در ان داده ها به خوشه ها یا زیرمجموعه هایی تقسیم میشوند که داده های در یک خوشه ویژگی های شبیه به هم دارند.

۱۱.۱.۱ بیش برازش و کم برازش

بیش برازش هنگامی است که مدل یادگیری بیش از حد جزییات از داده ها یاد میگیرد و در نتیجه نویز ها را نیز یاد میگیرد اما داده های جدید این نویز ها را ندارند و در این حالت الگوریتم برای داده های جدید خوب کار نمیکند. این مشکل میتواند به دلیل پیچیدگی بیش از حد مدل بوجود بیاید.
کم برازش هنگامی اتفاق می افتد که مدل نتواند الگوهای موجود در داده ها را به خوبی یاد بگیرد و در نتیجه نتواند روی داده های جدید پیشبینی خوبی داشته باشد.

۲.۱ همبستگی بین ویژگی ها

کواریانس بین دو ویژگی نوع همبستگی خطی بین دو ویژگی را مشخص میکند اما کواریانس مقدار همبستگی را نمیتواند مشخص کند و فقط به ما میگوید بین دو دسته از ویژگی ها رابطه خطی وجود دارد یا خیر و اگر وجود دارد مثبت یا منفی است ولی همبستگی که در واقع مقدار نرمال شده کواریانس به وسیله واریانس دو ویژگی است همیشه بین ۱ و -۱ میباشد و میتواند میزان همبستگی را نیز مشخص کند.
از روی نمودار `scatter plot` دو ویژگی میتوان تشخیص داد که بین دو ویژگی همبستگی وجود دارد یا خیر به این صورت که اگر بتوانیم با یک خط با شیب مثبت یا منفی رابطه این دو ویژگی را تقریب بزنیم یعنی همبستگی دارند و اگر نه یعنی همبستگی بین ان ها صفر است.

۳.۱ مقایسه معیار های ارزیابی

`MSE` برابر میانگین مربعات خطا ها میباشد.
`RMSE` برابر ریشه دوم میانگین مربعات خطا ها میباشد.
`MAE` برابر میانگین قدر مطلق خطاها میباشد.
در برابر داده های پرت `MAE` بهتر از همه عمل میکند زیرا میزان جریمه کمتری نسبت به دو معیار دیگر دارد. `RMSE` نیز بهتر از `MSE` عمل میکند چون میزان جریمه برای خطا را کمتر در نظر میگیرد. و همچنین به دلیل استفاده نکردن از قدر مطلق مشتق پذیر نیز میباشد.

۴.۱ مقایسه گرادیان نزولی و معادله نرمال

۱

گرادیان نزولی	معادله نرمال
نیاز داریم نرخ یادگیری را انتخاب کنیم	نیازی به مشخص کردن نرخ یادگیری نیست.
پیچیدگی زمانی الگوریتم نسبت به تعداد ویژگی ها از مرتبه دو است	پیچیدگی زمانی الگوریتم نسبت به تعداد ویژگی ها از مرتبه سه است.
به مشکل معکوس پذیر نبودن برنمیخوریم	ممکن است به مشکل معکوس ناپذیری بربخوریم
برای سریع تر شدن الگوریتم میتوان از Feature scaling استفاده کرد	نیازی به Feature scaling نیست
یک رویکرد تکراری است	یک رویکرد تحلیلی است
با تعداد Feature های زیاد خوب کار میکند.	با تعداد Feature های کم خوب کار میکند

جدول ۱: مقایسه گرادیان نزولی و معادله نرمال

۵.۱ رگرسیون Lasso

این نوع رگرسیون مانند رگرسیون خطی است با این تفاوت که در عبارتی که میخواهد ان را کمینه کند یک ترم اضافه دارد.
این ترم اضافه جمع مقادیر ضریب های پارامتر ها میباشد که در یک ضریب ثابت ضرب شده اند. در واقع این عبارت اضافه به Regularization کمک میکند و میتواند ضرایب پارامتر هایی که مهم نیستند را صفر کند یا بسیار کوچک کند و در نتیجه مدل ساده تر میشود و از بیش برازش جلوگیری میشود.

۲ سوالات پیاده سازی

توجه شود که دیتاست ها باید درمسیری باشند که کد اجرا میشود. در تمامی مراحل هفتاد درصد داده ها به مجموعه اموزش و سی درصد به مجموعه ارزیابی اختصاص داده شده اند.

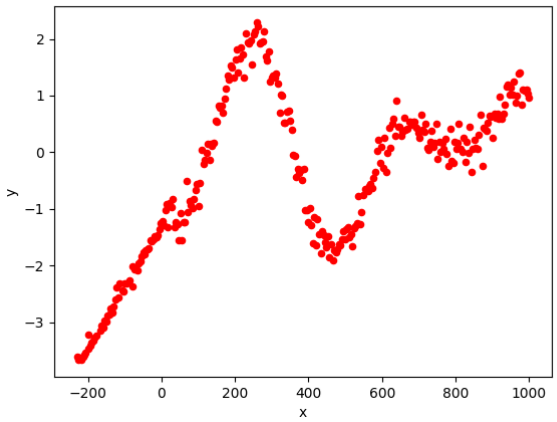
۱.۲ بخش اول

۱.۱.۲ نمودار داده ها

این نمودار با نام output1 ذخیره میشود.

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# reading data from csv file
firstData = pd.read_csv('Dataset1.csv')

#plotting the data and saving it
firstData.plot(kind = 'scatter' , x = 'x' , y = 'y' , color = 'red')
plt.savefig('output1.png')
```

Listing : ۱



شکل ۱: نمودار مجموعه داده اول

۲.۱.۲ شافل کردن و نرمال سازی

شافل کردن داده ها به این منظور انجام میشود که مجموعه داده های اموزش و تست و validation توزیع یکنواختی از داده ها داشته باشند و مثلا در داده های اموزش فقط یکسری داده ها با یک ویژگی خاص نباشند. در واقع در هر مجموعه داده ها به طور تقریبا یکسان توزیع شده باشند. نرمال سازی داده ها به این منظور انجام میشود که محدوده همه ویژگی هم یکسان باشد . در نتیجه این کار الگوریتم نزول گرادیان زودتر به نقطه کمینه میرسد همچنین اگر محدوده ویژگی ها خیلی تفاوت داشته باشد ممکن است در رگرسیون به ویژگی ای که مقادیر بزرگ تری دارد وزن بیشتری اختصاص دهیم . این مجموعه از داده ها چون فقط یک ویژگی دارد به نرمال سازی احتیاج ندارد اما به شافل کردن احتیاج دارد چون داده ها دارای ترتیب خاصی هستند بر اساس x مرتب شده اند.

اما اگر توان های بالاتر x را به عنوان ویژگی اضافه کنیم در ان صورت به نرمال سازی هم احتیاج داریم چون در این صورت محدوده ویژگی ها تفاوت زیادی خواهد داشت. کد های شافل کردن و نرمال سازی در ادامه امده است. توضیحات مربوط به کد ها در کامنت ها امده است.

```python
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# reading data from csv file
firstData = pd.read_csv('Dataset1.csv')

#shuffling data
firstData = firstData.sample(frac = 1).reset_index(drop=True)
```

```python
# a function for scaling features between -0.5 and 0.5
def scaling(x,d):
    """[x(i) := (x(i) - mean) / range]

    Args:
        x ([numpy array (m * d+1)]): [input matrix for scaling]
        d ([integer]): [degree of the polynomial model]

    Returns:
        [numpy array]: [scaled martix except the first colmun that is 1]
    """

    # slicing the first column that is 1 from matrix
    temp = x[: , 1:]
    # range should not be zero (max - min != 0)
    # scaling
    temp = [(temp[:,i] - temp[:,i].mean())/(temp[:,i].max() - temp[:,i].min()) for i in range(d)]

    temp = np.array(temp)
    temp = temp.T

    return np.append(np.ones(( x.shape[0], 1)) , temp , axis = 1) #adding [1] column
```

Listing :۲

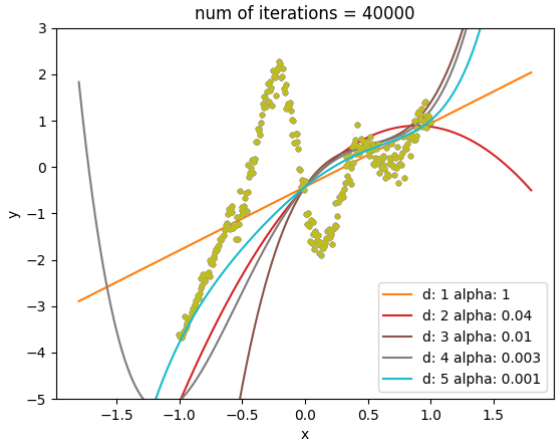
تابع scaling به عنوان ورودی ماتریس ویژگی ها و درجه مدل را دریافت میکند و به جز ستون اول ماتریس که همه مقادیر ان یک هست مقادیر بقیه ویژگی ها را نرمال میکند. این تابع برای هر مقدار ان را منهای میانگین ان ویژگی میکند و حاصل را به محدوده ان ویژگی تقسیم میکند این کار هر ویزگی را به محدوده ۰/۵ و -۰/۵ map میکند.

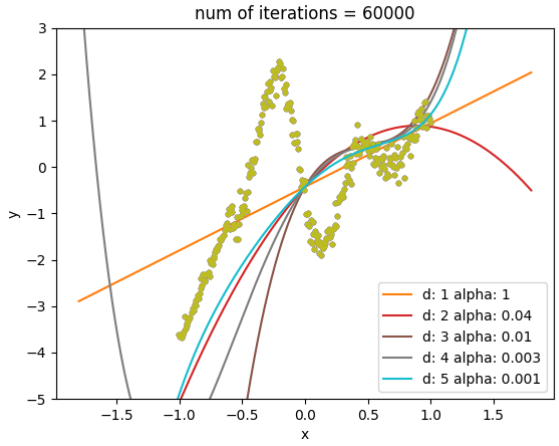
۳.۱.۲ پیاده سازی الگوریتم گرادیان نزولی

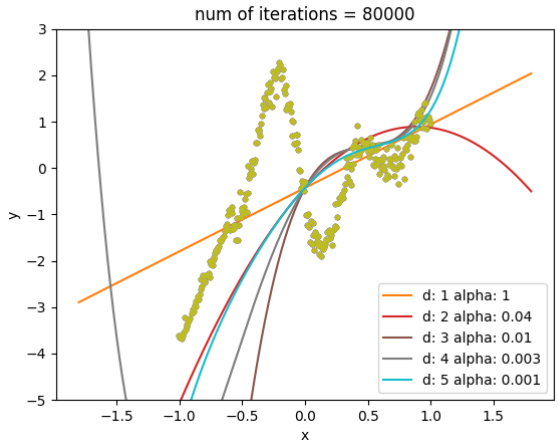
```python
def GD(target , features, reg , d=1 , iter_num=1000 , alpha =0.0001  ):
    """[gradiant decent]

    Args:
        target ([numpy array (m * 1)]): [target vector]
        features ([numpy array (m * 1)]): [raw features vector]
        reg ([float]): [regularization factor]
        d (int, optional): [degree of model]. Defaults to 1.
        iter_num (int, optional): [number of iterations]. Defaults to 1000.
        alpha (float, optional): [learning rate]. Defaults to 0.0001.

    Returns:
        [numpy array (d+1 * 1)]: [theta vector]
    """

    theta = np.zeros((d+1,1)) # initializing theta to zero vector
    theta_temp = theta

    m = features.shape[0] # m = num of samples
    y = target

    x = make_featurs(features , d) # adding polynomial features to features matrix
    x = scaling(x , d) #scaling features

    for i in range(iter_num):
        #theta = theta - alpha * d (J) + lambda * theta power 2
        theta_temp = theta*(1 - (alpha *reg) / m) - alpha * (1 / m) * np.dot(x.T , np.dot(x , theta) - y)
        #this line is becuase of regularization , we should not regularize theta0
        theta_temp[0] = theta_temp[0] + (alpha * reg)/m * theta[0]

        theta = theta_temp

    return theta
```

Listing :۳

تابع کاهش گرادیان که کد ان در بالا اورده شده با استفاده از عملیات ماتریسی پیاده سازی شده است. این تابع ابتدا وکتور تتا را با مقدار صفر مقداردهی اولیه میکند. سپس به ماتریس ویژگی که در ابتدا فقط یک ویژگی دارد ویژگی های توانی را اضافه میکند و بعد نرمال سازی را انجام میدهد. سپس عملیات تکراری گرادیان شروع میشود و در هر مرحله به مقادیر تتا مقادیری را اضافه یا از ان ها کم میکند طبق فرمول گرادیان کاهشی. همچنین در این فرمول رگولاریزیشن نیز در نظر گرفته شده است در ادامه تصاویر نمودار برازش شده به ازای هر درجه امده است. همچنین در بالای هر تصویر مقادیر خواسته شده گزارش شده است. دقت کنید که ویژگی ها در محدوده ۱ و -۱ نرمال شده اند. تمامی کد های این قسمت در فایل c.py امده است.







شکل ۲: نمودار های برازش شده بوسیله گرادیان نزولی

خطاها به صورت زیر میباشند.

```
d= 1 iter=40000train err: 0.6320809148208427 test err: 0.7389574360705377
d= 2 iter=40000train err: 0.5601652305424956 test err: 0.5594651061003666
d= 3 iter=40000train err: 0.40496860552396596 test err: 0.3991620454947855
d= 4 iter=40000train err: 0.4680041766304053 test err: 0.48845446846049106
d= 5 iter=40000train err: 0.5507781317597473 test err: 0.6115189684199582
d= 1 iter=60000train err: 0.6320809148208427 test err: 0.7389574360705377
d= 2 iter=60000train err: 0.5601652305424956 test err: 0.5594651061003666
d= 3 iter=60000train err: 0.37179590995223255 test err: 0.3643389326500257
d= 4 iter=60000train err: 0.42631992827998877 test err: 0.43332035910912353
d= 5 iter=60000train err: 0.5137197195068379 test err: 0.558099712325108
d= 1 iter=80000train err: 0.6320809148208427 test err: 0.7389574360705377
d= 2 iter=80000train err: 0.5601652305424956 test err: 0.5594651061003666
d= 3 iter=80000train err: 0.3554537832803988 test err: 0.3504845524506728
d= 4 iter=80000train err: 0.40007545454018 test err: 0.3990354679785045
d= 5 iter=80000train err: 0.48490613284346556 test err: 0.5172078786063234
```

میبینیم که با درجه یک یک مدل خطی برازش شده است و چون داده ها نرمال شده الفا نیز خیلی کوچک نیست و الگوریتم با تعداد گام های کمی به نقطه بهینه میرسد. همچنین توجه شود که الگوریتم با ۴۰۰۰ گام نیز به نقطه بهینه رسیده و خط ما با افزایش گام ها تغییری نمیکند و در نتیجه خطا ها هم تغییری نمیکنند. خطای ارزیابی و خطای اموزش نزدیک به هم هستند که نشان میدهد خط برازش شده عملکرد مناسبی دارد و روند کلی داده ها را یاد گرفته است و بیش برازش نیز نداریم.

در مورد مدل درجه دو میبینیم که میزان خطای اموزش نسبت به مدل خطی کمتر شده همچنین میبینیم که در این حالت با افزایش تعداد گام ها میزان خطاها تغییر نمیکند که نشان میدهد مدل بهتر نمیتواند عمل کند همچنین میزان الفا نیز نسبت به مدل خطی کوچکتر شده است.

در مدل مدل های درجه سه میبینیم که الفا کوچکتر شده همچنین مقدار خطا نیز نسبت به مدل های قبلی کمتر شده و همچنین با افزایش گام ها نیز خطا کاهش میابد در مدل درجه چهار نیز الفا کوچکتر شده و با

افزایش گام ها نیز خطا ها کمتر میشوند اما خطا از مدل درجه ۳ بیشتر است که نشان میدهد مدل درجه ۳ بهتر داده ها را یاد گرفته در مدل درجه ۵ نیز با افزایش تعداد گام ها خطا کاهش میابد اما خطای این مدل نیز از

مدل درجه ۳ و ۴ بیشتر است.

۴.۱.۲ رگولاریزیشن

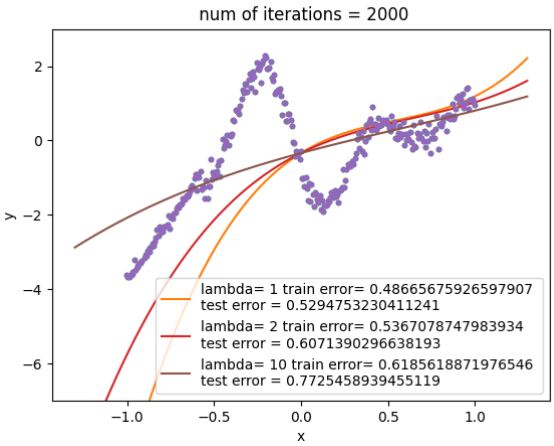
بهترین مدل در بخش قبل مدل درجه سه با تعداد تکرار ۸۰۰۰۰ بود مشاهده میشود که با افزایش ضریب رگولاریزیشن نمودار برازش شده smooth تر میشود و خطای هر دو فاز اموزش و ارزیابی نیز بیشتر میشود بردار تتا به ازای ضرایب مختلف رگولاریزیشن به صورت زیر است.

```
regularization=  1
[[-0.33663563]
 [ 2.89829935]
 [-3.54469009]
 [ 2.17263767]]

regularization=  2
[[-0.33663563]
 [ 2.32040692]
 [-2.02467311]
 [ 1.06934294]]

regularization=  10
[[-0.33663563]
 [ 1.26535928]
 [-0.30031485]
 [ 0.17517285]]
```

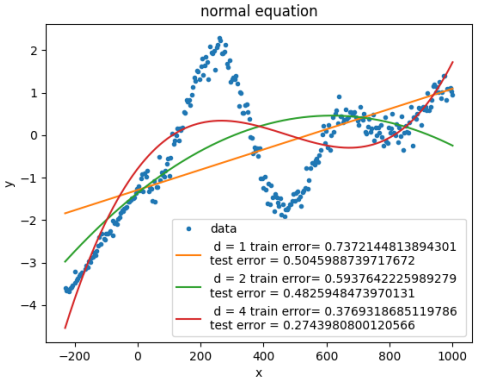
شکل ۳:

مشاهده میشود که تتای صفر به ازای ضرایب مختلف تغییری نمیکند چون ضریب رگولاریزیشن برای تتا صفر را صفر در نظر گرفتیم. همچنین ضرایب ویژگی های با درجه بیشتر با بزرگ تر شدن ضریب رگولاریزیشن به صفر نزدیک تر میشوند چون با بزرگ تر شدن ضریب رگولاریزیشن میزان جریمه برای اندازه ضرایب بیشتر میشود و در نتیجه این ضرایب اگر بیش از حد بزرگ شوند باعث میشوند تابع هزینه زیاد شود.

در نمودار های مدل نیز مشاهده میشود با ضریب یک نمودار smooth تر شده و خطای هر دو فاز افزایش میابد و هر چه ضریب رگولاریزیشن بیشتر شود همین روال ادامه دارد و برای ضریب ده مشاهده میشود نمودار دچارکم برازش شده است و نتوانسته الگوی داده ها را به خوبی یاد بگیرد. نمودار های این بخش در صفحه بعد امده اند.



شکل ۴: مدل با درجه۳ به همراه رگولاریزیشن



شکل ۵: مدل های برازش شده به وسیله معادله نرمال

۵.۱.۲ معادله نرمال

```python
def normal_eq(x , y , d):
    """[summary]

    Args:
        x ([numpy array (m * 1)]): [raw feature vectores]
        y ([numpy array (m * 1)]): [targets vector]
        d ([int]): [degree of model]

    Returns:
        [numpy array (d+1 * 1)]: [theta vector]
    """

    x = make_featurs(x , d) # adding polynomial features to features matrix

    temp = np.linalg.inv(np.dot(x.T , x)) #(X^T * X) ^ -1
    theta = np.dot(np.dot(temp , x.T) , y) # temp * X^T * y

    return theta
```

Listing :۴

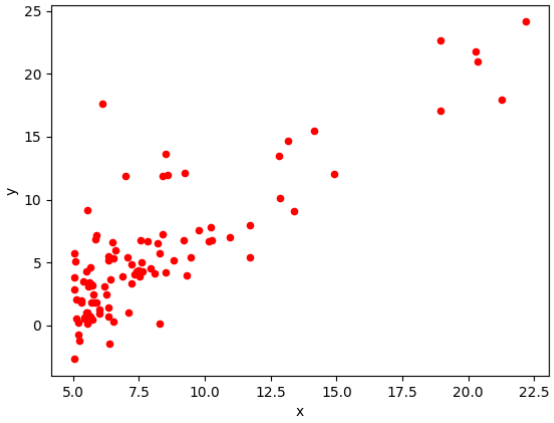
کد معادله نرمال در بالا امده است. این تابع به عنوان ورودی بردار ویژگی ها و نتایج و درجه مدل را میگیرد. سپس ویژگی های با درجه بالا تر را به ویژگی ها اضافه میکند. و بعد بردار تتا را طبق فرمول معادله نرمال محاسبه میکند.

این روش نیازی به نرمال سازی داده ها ندارد. شکل نمودار های برازش شده با این مدل در بالای صفحه امده است. همانطورکه مشاهده میشود با پیچیده تر شدن مدل میزان خطا ها کاهش میابد و همچنین میبینیم که مدل درجه یک به خوبی نمیتواند داده ها را یاد بگیرد و دچارکم برازش شده است ولی مدل درجه ۴ به خوبی عمل کرده و الگوی داده ها را به خوبی یاد گرفته است. این نمودار با نام 5e ذخیره شده است.

۲.۲ بخش دوم

۱.۲.۲ رسم نمودار داده ها

این نمودار با نام output2 ذخیره شده است.

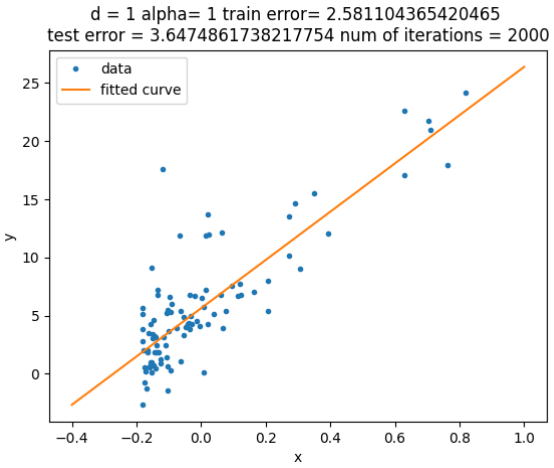


شکل ۶: نمودار مجموعه داده دوم

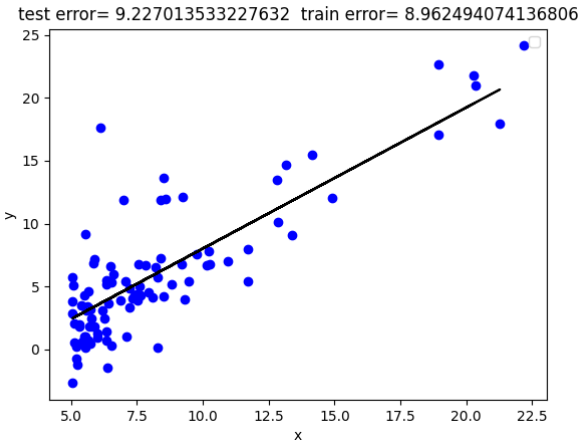
۲.۲.۲ اجرای گرادیان نزولی روی داده ها

۳.۲.۲ رگرسیون خطی با استفاده از کتابخانه scikit-learn

نمودار های این دو بخش در صفحه بعد امده اند
باید توجه کنیم که در این کتابخانه در محاسبه MSE مخرج کسر تعداد نمونه ها میباشد و نه دوبرابر ان ها و در نتیجه برای مقایسه این خطا ها با خطای مدل های قبل باید این خطا ها را تقسیم بر دو کنیم.



شکل ۷: اجرای گرادیان نزولی روی مجموعه داده دوم



شکل ۸: رگرسیون خطی با استفاده از یک کتابخانه اماده