گزارش تمرین اول

(شناسایی آماری الگو)

تهیه شده توسط:

سارا لیمویی، ملیکا زارع

استاد:

دکتر عظیمی فر

پاییز ۱۴۰۰


بسم الله الرحمن الرحیم

فهرست مطالب

پیش گفتار

در این تمرین قصد داریم تا رگرسیون خطی را به کمک دو روش closed-form و الگوریتم gradient descent بر روی محموعه داده ی Data-Train و Data-Test پیاده سازی کنیم. در رگرسیون خطی به دنبال آن هستیم که تابعِ رابطه ی بین X و y را پیدا کنیم.

آماده سازی داده ها

مجموعه ی داده ی ما به دو بخش data-test و data-train تقسیم می شود. تعداد داده ی train، ۱۰۰۰ و تعداد داده ی test، ۳۰۰ می باشد. در ادامه، ۵ داده ی اول هر کدام از مجموعه ها را نشان داده-ایم.

```
In [5]: dataset.data_test.head()
```

Out[5]:

	x	y
0	77	79.775152
1	21	23.177279
2	22	25.609262
3	20	17.857388
4	36	41.849864

```
In [6]: dataset.data_train.head()
```

Out[6]:

	x	y
0	24	21.549452
1	50	47.464463
2	15	17.218656
3	38	36.586398
4	87	87.288984

شکل (۱)

همان طور که در شکل بالا آمده است، در هر کدام از فایل های data-train و data-test دو ستون با نام های x و y داریم. x ویژگی هر sample را نشان می دهد و y نیز نشان دهنده خروجی تابع به ازای هر sample می باشد.

برای خواندن دیتاست از روی فایل csv مدنظر کلاس Dataset در فایل "PreprocessData.py" نوشته شده است. با ساختن یک object از کلاس Dataset، می توان به هر کدام از مجموعه های ویژگی های sample ها و یا خروجی های y آنها، هم برای داده ی test و هم train دسترسی داشت. برای بدست آوردن دیتاست، به صورت زیر از کلاس تعریف شده استفاده می کنیم:

```
dataset = Dataset('Data-Train.csv', 'Data-Test.csv', 'y', normalization_method='zero_mean_unit_var')
```

در تابع __init__() کلاس Dataset، متغیر های data_train_file ، data_test_file، output_column_name و normalization_method را به عنوان ورودی می گیریم. data_train_file و data_test_file آدرس فایل های ورودی داده های test و train می باشد و output_column_name نام ستون خروجی در این تمرین یعنی "y" می باشد و در normalization_method، می توان نوع normalization را مشخص کرد، "zero_mean_unit_var" و یا "scale_0_1" و یا "none".

در این تابع __init__() که به هنگام ساختن object صدا زده می شود، داده های ورودی مدل رگرسیون خطی و خروجی ها را از هم جدا می کند.

همچنین تابع normalize در کلاس Dataset، داده های ورودی یعنی x_train و x_test را با توجه به روشی که ذکر شده نرمال می کند.

برای پیاده سازی رگرسیون به دو روش closed-form و gradient-descent که در ادامه توضیحات آن ها آورده شده است، باید یک ستون x0 با مقدار عددی ۱ به ابتدای تمام sample ها اضافه کنیم که در تابع add_vector_x0 پیاده سازی شده است.

قبل از اجرای هر کدام از مدل های رگرسیون خطی، باید یک object از کلاس Dataset بسازیم.

کلاس Regression

کلاس رگرسیون، یک کلاس base می باشد که هر کدام از روش های Closed-Form و Gradient-Descent از آن ارث بری می کنند.

این کلاس شامل یک متغیر theta می باشد که شامل همه ی θ ها برای هر کدام از ویژگی ها می باشد. همچنین، Regression دارای توابع زیر نیز می باشد که در ادامه هر کدام توضیح داده شده اند.

تابع mse:

با گرفتن y اصلی و y-predict که توسط مدل بدست می آید، تابع هزینه (هدف) را به کمک روش زیر بدست می آورد:

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{m}\left(\hat{y}^j - y^j\right)^2$$

تابع predict:

در این تابع، با استفاده از x و theta ای که در کلاس Regression قرار دارد y-predict را به کمک فرمول زیر محاسبه می شود. مقدار theta در ادامه توسط هر یک از کلاس های ClosedForm و BGD مقداردهی خواهد شد.

$$\hat{y} = y\_predict = X\,\theta$$

تابع fit:

این تابع به صورت abstract است که در هر یک از کلاس های BGD و ClosedForm باید پیاده سازی شود.

تابع plot:

در این تابع با استفاده از کتابخانه ی matplotlib، داده های train و test و رگرسیون خطی بدست آمده به صورت نمودار plot می شوند. به عبارت دیگر با داشتن θ_0 و θ_1 به ترتیب به عنوان عرض از مبدا و شیب خط، خط بدست آمده را رسم می کنیم.

$$y = \theta_0 + \theta_1 X$$

تابع report:

با فراخوانی این تابع، اطلاعات مورد نیاز از جمله θ_0 و θ_1 و MSE بر روی هر کدام از داده های test و train چاپ می شود.

Closed-Form Solution

روش closed-form به روشی گفته می شود که در آن بتوان یک مساله را با یک روش مشخص و به کمک تابعی از ورودی های محدود حل کرد. به عنوان مثال جمع تعداد نامحدودی از اعداد، به عنوان یک closed-form شناخته نمی شود. در ادامه روش closed-form که برای Linear Regression-Least Square Error در این تمرین پیاده سازی می شود را بررسی می کنیم.

در این روش ابتدا $J(\theta)$ را به صورت زیر تعریف می کنیم:

$$J(\theta) = \sum_{j=1}^{m} \frac{1}{2} \left(h_\theta(X^j) - y^j\right)^2$$

که h_θ همان y-predict به ازای هر x-new ای است. برای پیدا کردن θ هایی که $J(\theta)$ را min کنند، از تابع فوق مشتق گرفته و برابر صفر قرار می دهیم تا θ ای را که نقطه ی مینیمم $J(\theta)$ است را بدست آوریم. برای این منظور داریم:

$$\nabla J(\theta) = 0$$

$$\nabla \frac{1}{2}(X\theta - y)^T(X\theta - y) = 0$$

در نتیجه ی حل عبارت فوق داریم:

$$\theta = (X^T X)^{-1} X^T y$$

از طریق محاسبه ی عبارت بالا که در تابع fit در کلاس ClosedForm پیاده سازی شده است، θ های مورد نیاز را پیدا کرده و سپس به کمک توابع report و plot، نتایج لازم را استخراج می کنیم.

نتایج Closed-Form Solution

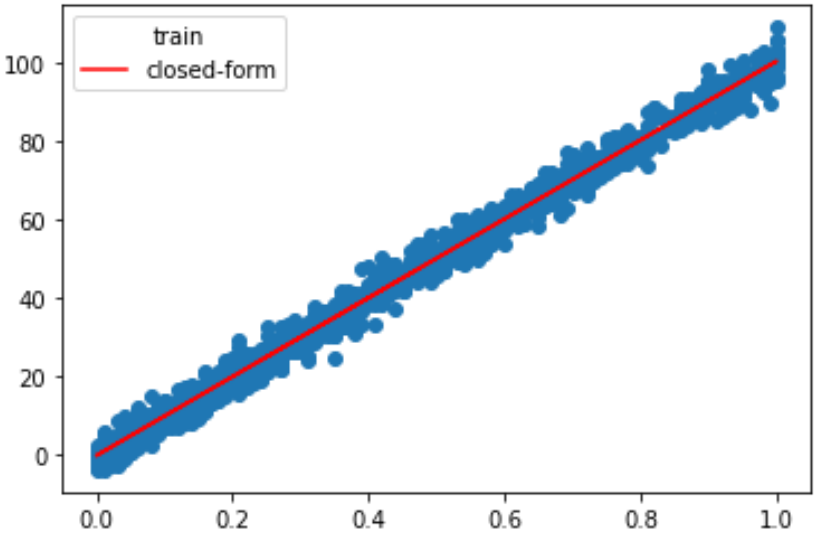
در ادامه نتایج مربوط به این روش آورده شده است:

```
dataset = PreprocessData.Dataset('Data-Train.csv', 'Data-Test.csv', 'y', normalization_method='scale_0_1')

closed_form = ClosedForm()
closed_form.fit(dataset.x_train, dataset.y_train)

closed_form.predict(dataset.x_test)

closed_form.report(dataset.x_train, dataset.y_train, dataset.x_test, dataset.y_test)
```

```
scaling to [0, 1] normalization.
theta_ 0 :  -0.21953583859344888
theta_ 1 :  100.48368423780036
MSE on train:  4164.006185786953
MSE on test:  1394.3540142796685
```

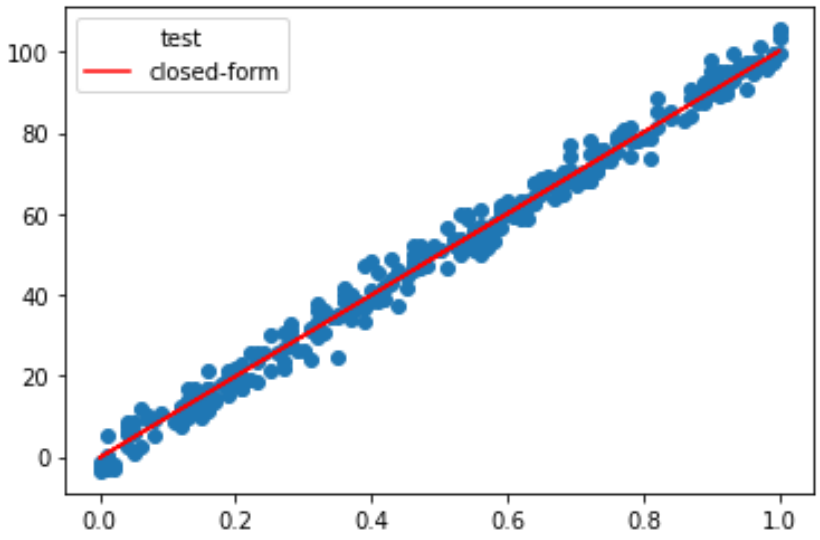
شکل (۲)

با استفاده از تابع plot میتوانیم نمودار داده های اموزشی و خط رگرسیون بدست آمده را مشاهده کنیم.



شکل (۳)

همچنین نمودار داده ی تست را نیز مانند قبل رسم می کنیم.



شکل (۴)

Batch Gradient Descent

متأسفانه محاسبه راه حل فرم بسته بسیار هزینه بر و زمان بر است. با این حال در موقعیت های زیر به مشکل نیز بر میخورد:

۱. برای اکثر مسائل رگرسیون غیر خطی، هیچ راه حل بسته ای وجود ندارد(به دلیل معکوس پذیر نبودن).
۲. حتی در رگرسیون خطی (یکی از معدود مواردی که راه حل شکل بسته در دسترس است)، ممکن است استفاده از فرمول غیرعملی باشد .به عنوان مثال،X یک ماتریس پراکنده بسیار بزرگ باشد.

Gradient Descent برای به حداقل رساندن تابع هدف (یا هزینه) با تکرار حرکت روی داده در جهت کاهش شیب مشتق تابع به نقطه بهینه می رسد.

این روش سه مدل مختلف دارد که ما در این تمرین Batch Gradient Descent را پیاده سازی می-کنیم.

ابتدا باید تابع هزینه را مشخص کنیم. برای این سوال همانند فرم بسته از تابع MSE بهره برده ایم. این روش به تعداد دفعات با عبور از روی داده های آموزشی مقدار شیب و عرض از مبدا را به روز می کند. فرمول آپدیت کردن θ به صورت زیر می باشد:

$$\theta_i = \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$$

بنابراین لازم داریم تا مشتق جزیی تابع هزینه را بر حسب $\partial \theta_i$ بدست آوریم:

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{\partial}{\partial \theta_i} \sum_{j=1}^{m} \frac{1}{2} \left(h_\theta(X^j) - y^j \right)^2$$

$$= \sum_{j=1}^{m} \left(h_\theta(X^j) - y^j \right) x_i^j$$

که با جایگذاری در فرمول قبل روش مدنظر بدست می آید.

در فایل کد داده شده یک کلاس BGD تعریف کرده ایم که از کلاس Regression ارث بری میکند تا توابع مورد نیازی مانند mse، predict، plot را استفاده کند. اما تابع fit را باید پیاده سازی کنیم.

تابع fit با گرفتن x_train, y_train و همچنین اپسیلونی اختیاری، مقدار θ را بدست می آورد. برای این کار ابتدا یک ماتریس با بُعد (n,1) از اعداد رندوم برای θ اختیار می کنیم. سپس مطابق فرمول بدست آمده در بالا به تعداد دفعاتِ max_iter، مقدار ماتریس θ را آپدیت می کنیم. در هر دوره با استفاده از θ بدست آمده هزینه ی mse در آن iteration مشخص می شود. اگر تفاوت هزینه هر θ با دوره قبلی آن کمتر از مقدار اپسیلون باشد به این معنی است که الگوریتم converged کرده و دیگر بهبودی در سیستم نداریم پس حلقه for را به اتمام می رسانیم.

در نهایت با فراخوانی تابع predict از کلاس ارث بری شده با مقدار x_test می توانیم y حدس زده شده را مشاهده کنیم و با فراخوانی تابع report از همان کلاس مقدار θ نهایی و مقدار هزینه MSE بر روی داده-های آموزشی و داده های تست چاپ می شود.

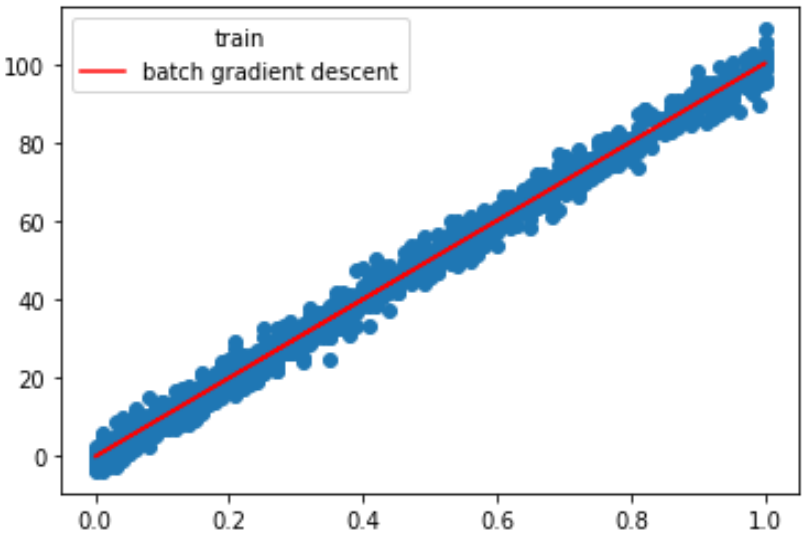
```
dataset = PreprocessData.Dataset('Data-Train.csv', 'Data-Test.csv', 'y', normalization_method='scale_0_1')

bgd= BGD(alpha=1e-3, max_iter=4000)

bgd.fit(dataset.x_train, dataset.y_train)

bgd.predict(dataset.x_test)

bgd.report(dataset.x_train, dataset.y_train, dataset.x_test, dataset.y_test)
```

```
scaling to [0, 1] normalization.
Iteration 214 with difference 9.47602529777214e-09 converged.
theta_ 0 :  -0.2195145894964904
theta_ 1 :  100.48364472932899
MSE on train:  4164.006185852577
MSE on test:  1394.3548444604237
```

شکل (۵)

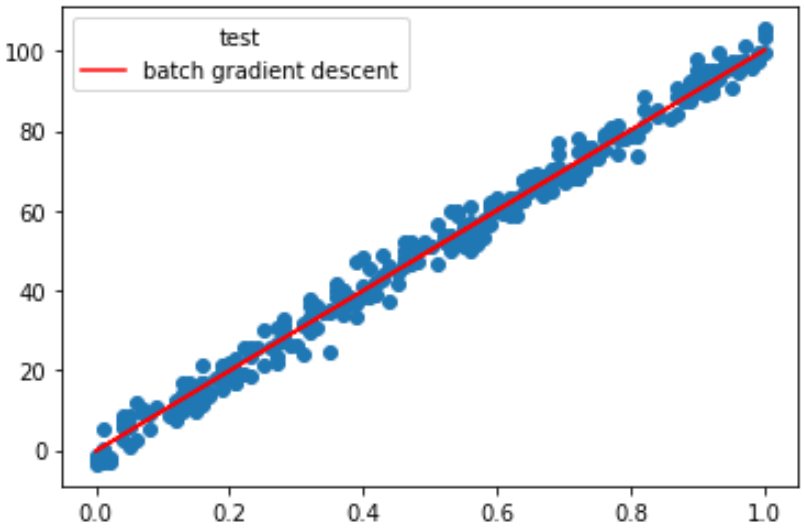
نتایج Batch Gradient Descent Solution

با استفاده از تابع plot میتوانیم نمودار داده های اموزشی و خط رگرسیون بدست آمده را مشاهده کنیم.



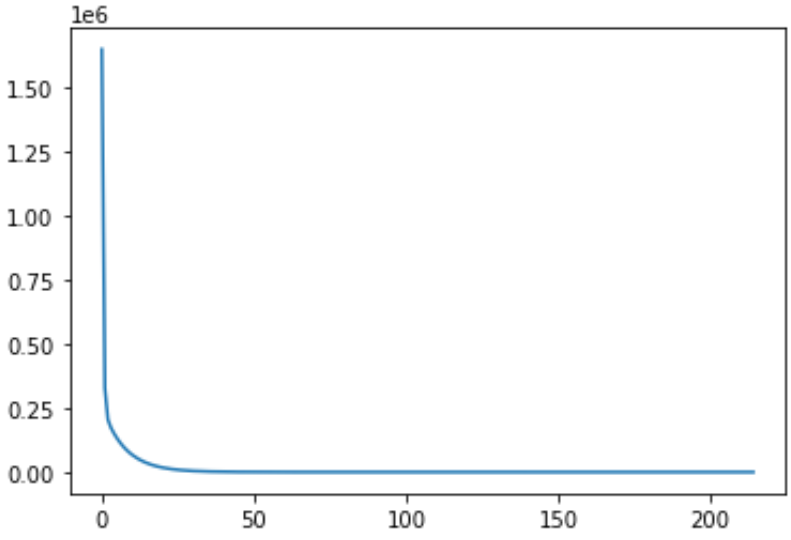
شکل (۶)

همچنین نمودار داده ی تست را نیز مانند قبل رسم می کنیم.



شکل (۷)

در نهایت با فراخوانی تابع plot_cost نمودار تغییرات هزینه در هر دوره (iteration) را داریم.



شکل (۸)