به نام خدا

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

رگرسیون خطی چند متغیره

پدیدآورنده:

علی کثیری

استاد:

استاد حسین کارشناس

پاییز۱۴۰۲

رگرسیوین خطی چند متغیره

خواندن داده ها

تابع **`read_data()`**:

این تابع به طور کلی از دو بخش تشکیل شده است.

1- خواندن اطلاعات از فایل csv :

در این قسمت با استفاده از کتاب خانه pandas و تابع read_csv() اطلاعات را بیرون کشیده و ذخیره میکنیم ، سپس داده های مربوط به سطون price را که در تابع خطی ما حکم Y را دارد در متغیر Y ذخیره کرده و همچنین داده های دیگری که به صورت عددی هستند را در متغیر X ذخیره میکنیم.

2- تبدیل متغیر های دسته ای به متغیر های عددی:

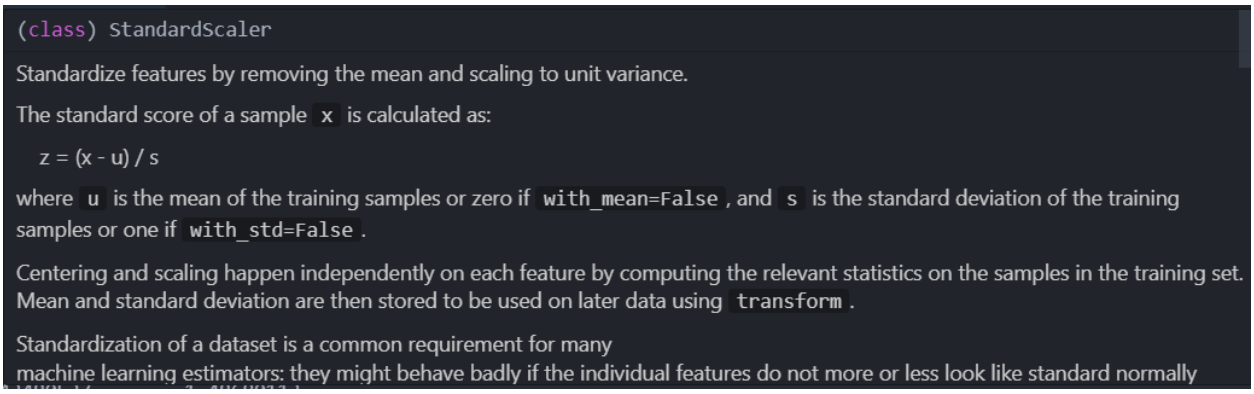
تابع get_dummies() :

این تابع از توابع کتاب خانه pandas است و داده های دسته ای را به داده های عددی به به فرم One-Hot ذخیره میکند ، در این قسمت داده های دسته ای که 4 داده ی departure_time و stops arrival_time و class بود را تبدیل به داده های عددی کرده و در متغیر X ذخیره کردیم.

نرمالیزه کردن داده ها

تابع `normalize_data()` :

در این تابع داده هارا نرمالیزه میکنیم به این صورت که هرنقطه را از میانگین نقاط کم و بر std که مخفف standard deviation است تقسیم میکنیم که این فرمول را ما به صورت دستی نوشتیم ولی کلاسی هم در کتابخانه sklearn.preprocessing به نام StandardScaler وجود دارد که همین کار را میکند:



و همچنین یک ستون با مقدار ثابت 1 به X اضافه میکنیم که در آینده نقش مقدار اولیه عرض از مبدا تابع خطی مارا ایفا میکند (در بخش gradient_desent() توضیح بیشتری درمود این ستون میدهیم) . در نهایت داده ها را که فرمت pandas دارند با تابع to_numpy() به ارایه ای از داده ها با فرمت numpy تغییر میدهیم تا عملیات ریاضی بر رویشان انجام دهیم.

تقسیم بندی مجموعه ها

تابع `train_test_split()`:

همانطور که در بخش مقدمات پروژه هم گفته شد با استفاده از کتابخوانه scikit-learn میتوان از این تابع استفاده کرد و داده ها را به دو بخش 80 درصد برای یادگیری مدل و 20 درصد برای تست گرفتن از مدل استفاده کرد تا مدل را صحت سنجی کرد ، درواقع این تکنیک 80،20 برای مدل از این جهت مفید است که مدل با کل داده ها اصطلاحا over fit نمیشود و میتوانیم بسنجیم که اگر داده ی جدیدی به مدل ما داده شود تا چه حد خوب عمل میکند.

الگوریتم gradient_desent:

الگوریتم کمینه‌سازی گرادیان یک روش بهینه‌سازی است و استفاده می‌شود تا بهینه‌ترین مقادیر پارامترهای یک مدل را پیدا کند. این الگوریتم به خصوص در مسائل یادگیری ماشین، از جمله رگرسیون خطی چند متغیره، بسیار مفید است.

برای رگرسیون خطی چند متغیره، فرض کنید که مدل خطی شما به شکل زیر باشد:

$$f(x_0,x_1,\ldots,x_n)=w_0\cdot x_0+w_1\cdot x_1+\ldots+w_n\cdot x_n+b$$

در اینجا:

$f(x_1,x_2,\ldots,x_n)$: مقدار پیشبینی شده توسط مدل

w0,w1,...w1 : پارامتر های مدل که باید یادگرفته شوند

X1,x2,...,xn : ویژگی های ورودی(متغیر های وابسته) مورد نظر

کلاس Gradient Descent یک تابع fit() دارد که به صورت مراحل زیر عمل می‌کند:

1. محاسبه خطا ها و پیشبینی ها (errors and predictions):
پیشبینی Yیا همان (prediction) با پارامتر های فعلی و محاسبه مقدار خطا با اختلاف پیشبینی و مقدار اصلی:

```
predictions = np.dot(X, self.coefficients)

errors = predictions - Y
```

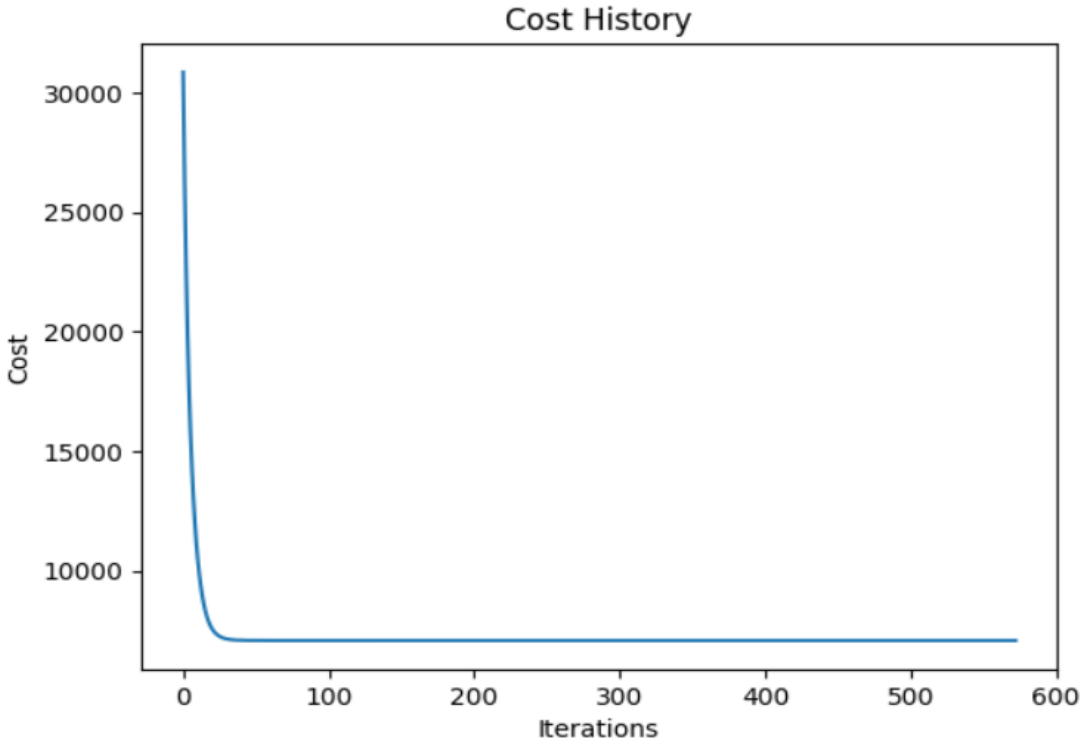
2. محاسبه گرادیان (Gradient Calculation): محاسبه مشتق جزئی تابع هزینه نسبت به هر یک از پارامترها. گرادیان نشان‌دهنده جهت افزایش سرعت کاهش تابع هزینه است.

3. به‌روزرسانی پارامترها (Parameter Update): به‌روزرسانی پارامترها با استفاده از گرادیان و یک نرخ یادگیری (learning rate). این نرخ تعیین‌کننده اندازه گامی است که در جهت مختصات گرادیان حرکت می‌کنیم.

```
gradient = np.dot(X.T, errors) / len(Y)
self.coefficients -= self.learning_rate * gradient
```

4. محاسبه تابع هزینه(cost) :

دراینجا ما از Root Mean squared Error(rmse) استفاده میکنیم که میانگین مربعات ریشه دار خطا ها است را بدست می اوریم و سعی داریم با هرچرخش در لوپ این مقدار را به کمینه برسانیم با به عبارتی مقدار پیش‌بینی شده توسط مدل به مقدار واقعی نزدیکتر شود. اگر نمودار تغییرات این تابع هزینه را نمایش دهیم میبینیم که در پیشبینی های اولیه اختلاف زیاد و شیب تند است اما با چرخش بیشتر در لوپ و بروزرسانی پارامتر ها ، اختلاف خیلی کم میشود.



5. تکرار (Iteration): مراحل 1 تا4 را تا زمانی که تابع هزینه به اندازه کافی کاهش یابد یا تا رسیدن به تعداد مشخصی تکرار انجام دهید.

همچنین دو شرط در این الگوریتم به کار بردیم تا به زمان و نتیجه بهتری برسیم که بدین شرح هستند:

1. **تغییر نرخ یادگیری(learning rate) :**

در ابتدای یادگیری که تابع هزینه با شیب تندی درحال تغییر است الگوریتم به اصطلاح با گام های بلند در جهت مختصات گرادیان حرکت میکنیم اما پس از چندین بار چرخش نیاز به گام های کوتاه تری داریم تا به نرمی به سمت مختصات گرادیان حرکت کنیم پس برای همین اختلاف تابع هزینه با مقدار قبلی اش را محاسبه میکنیم و اگر از یک حدی کمتر بود نرخ یادگیری را در یک دهم ضرب میکنیم تا با گام های کوتاه تری حرکت کنیم.

```
if iteration > 0 and abs(self.cost_history[iteration - 1] - rmse) < 1e-3 and self.learning_rate > 0.01:
    self.learning_rate = self.learning_rate / 10
    print(abs(self.cost_history[iteration - 1] -rmse))
    print('learning_rate *= 0.1 ' ,self.learning_rate )
```

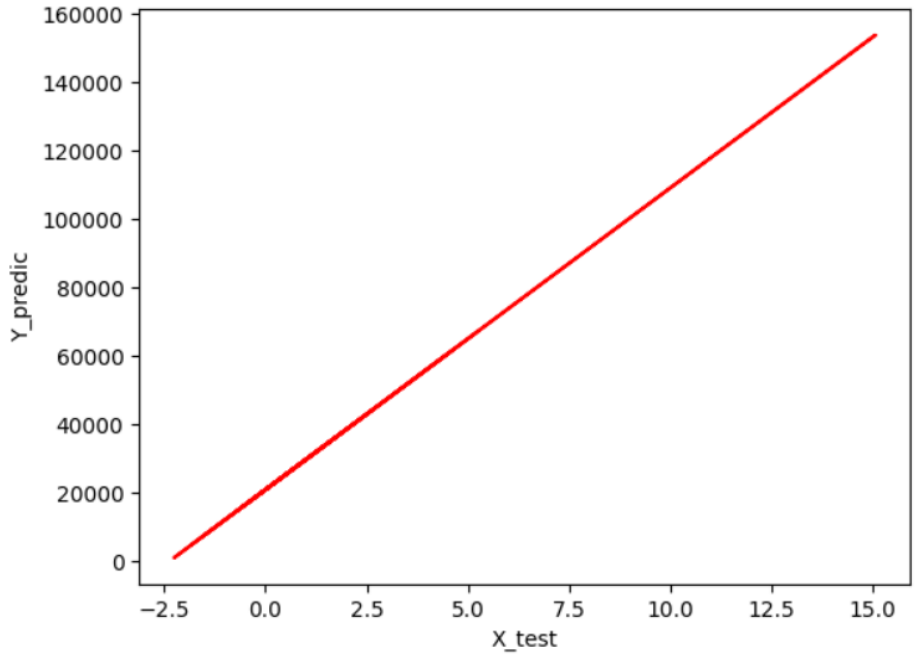
2. **رسیدن به حد مناسب:**

در الگوریتم گرادیان ما به تعداد مشخصی(iteration) در حلقه چرخش میکنیم اما اگر در تعدادی کمتر از این مقدار، تابع ما به پارامتر های مناسب رسید ، چرخش های بعدی ضروری نیست پس با یک شرط چک میکنیم که اگر اختلاف تابع هزینه(cost_history) با مقدار قبلی اش از یک مقدار خیلی نزدیک به صفر کمتر بود از حلقه بیرون بپریم.

```
if iteration > 0 and abs(self.cost_history[iteration - 1] - rmse) < self.tolerance:
    print(abs(self.cost_history[iteration - 1] - rmse))
    self.num_iterations = iteration + 1
    print(f"Converged after {iteration} iterations.")
    break
```

پیشبینی Y ها

کلاس Gradient Descent یک تابع دیگر به نام predic() دارد که در آن پارامتر های بدست آمده در تابع قبلی را در متغیر های وابسته(X) ضرب میکند و باهم جمع میکند که این مقدار همان مقدار پیشبینی شده برای Y است. برای ارزیابی مدلمان همان 20 درصد از داده ها(X_test) که جدا کرده بودیم برای ارزیابی را به تابع predeic() میدهیم تا به ما Y های پیشبینی شده(Y_predic) را بدهد.



ارزیابی

تابع output() :

این تابع از دو بخش 1-محاسبه و ارزیابی و 2-ذخیره اطلاعات تشکیل شده است:

1-محاسبه و ارزیابی:

در این بخش مقدار های MSE , RMSE , MAE , R2 را با استفاده از کتابخانه sklearn بدست می آوریم که هرکدام با فرمول ها و معیار های متفاوتی میزان خطا بر روی مجموعه تست را بدست می اورند.

2-ذخیره اطلاعات :

در نهایت پارامتر های بدست آمده و خطاهای MSE , RMSE , MAE , R2 و مدت زمان مورد نیاز برای آموزش را در یک فایل ذخیره میکنیم.

نمونه ای از خروجی :

```
PRICE = (205.21 * [duration]) + (-1786.53 * [days_left]) + (-291.90 *
+ (-150.01 * [arrival_time_Late_Night]) + (-33.58 * [arrival_time_Mor
Training Time: 6.324s

Logs:
MSE: 49691276.72451339
RMSE: 7049.203978075354
MAE: 4654.571519279775
R2: 0.9037245576230007
```

منابع

- Gradient Descent, Step-by-Step - YouTube
- **https://chat.openai.com/chat** **chat gpt**
- ویدیوی آموزشی مقدمه یادگیری ماشین با پایتون - مکتب‌خونه بخش رگرسیون