به نام یگانه خالق هستی بخش

پاسخنامه تمرین اول درس مبانی یادگیری ماشین

استاد

الهام قصرالدشتی

دستیاران آموزشی

مهرداد قصابی

مریم صفوی

گردآورنده

محمدامین نصیری

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

بهار ۱۴۰۴

فهرست

1. سوال اول: رگرسیون خطی

1. پیش‌پردازش داده‌ها

1. آشنایی با مجموعه داده

با استفاده از تابع *isna* تعداد داده‌های گم‌شده‌ی هر ستون و با استفاده از تابع *info* تعداد داده‌های موجود در هر ستون را نمایش می‌دهیم. همچنین با استفاده از تابع *describe* اطلاعات کلی داده‌های هر ستون مثل میانگین و کمترین و بیشترین مقدار را نشان می‌دهیم که نتایج استفاده از هر ۳ تابع در شکل ۱.۱ آمده است.

```
Missing values per column:
 SqFt            2
Bedrooms         3
Bathrooms        1
Offers           2
Brick            0
Neighborhood     0
```

الف) تابع isna

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117 entries, 0 to 116
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   SqFt          113 non-null    float64
 1   Bedrooms      112 non-null    float64
 2   Bathrooms     114 non-null    float64
 3   Offers        113 non-null    float64
 4   Brick         115 non-null    object
 5   Neighborhood  115 non-null    object
dtypes: float64(4), object(2)
memory usage: 5.6+ KB
```

ب) تابع info

```
Summary statistics:

              SqFt    Bedrooms   Bathrooms      Offers
count   113.000000  112.000000  114.000000  113.000000
mean   2008.761062    3.026786    2.464912    2.592920
std     214.247574    0.728565    0.518333    1.090848
min    1450.000000    2.000000    2.000000    1.000000
25%    1890.000000    3.000000    2.000000    2.000000
50%    2000.000000    3.000000    2.000000    3.000000
75%    2150.000000    3.000000    3.000000    3.000000
max    2590.000000    5.000000    4.000000    6.000000
```

ج) تابع describe

شکل ۱-۱ نتایج استفاده از ۳ تابع isna، info و describe

2. مدیریت داده‌های از دست رفته

- داده‌های عددی: داده‌های گمشده‌ی ستون‌های *SqFt* و *Bedrooms* و *Bathrooms* و *Offers* که شامل داده‌های عددی هستند را با استفاده از میانگین داده‌های موجود هر ستون بازیابی می‌کنیم. داده‌های از دست رفته‌ی تست را نیز با همین متد بازیابی می‌کنیم با این تفاوت که مرجع میانگین را همان داده‌های آموزشی درنظر می‌گیریم.
- داده‌های کلاس‌بندی: داده‌های گمشده‌ی ستون‌های *Brick* و *Neighborhood* که شامل داده‌های کلاس‌بندی هستند را با استفاده از *mode* داده‌های موجود هر ستون بازیابی می‌کنیم. *mode* یک ستون از داده برابر است با داده‌ای که بیشتر از بقیه‌ی مقدارها ظاهر شده است. داده‌های از دست رفته‌ی تست را نیز با همین متد بازیابی می‌کنیم با این تفاوت که مرجع میانگین را همان داده‌های آموزشی درنظر می‌گیریم. البته که با توجه به نتایج آورده شده در شکل ۱.۱ داده‌ی گم‌شده‌ی دسته‌ای نداریم و به بازیابی داده‌های کلاس‌بندی نیازی نیست.

3. حذف داده‌های پرت

- داده‌های عددی: با استفاده از متد *IQR* داده‌های پرت عددی همه‌ی ستون‌ها به جز ستون *Bedrooms* حذف شدند. حذف نکردن داده‌های پرت این ستون به این خاطر است که با توجه به نمودار توزیع نامتوازن این ویژگی همه‌ی مقدارهای غیر "3" این ویژگی باید حذف شوند که این کار منجر به حذف تعداد زیادی از داده‌ها شده و مقادیر این ویژگی را یکسان می‌کند. در اصل به دلیل کواریانس پایین این ویژگی با *price* بهتر است این ویژگی به کلی در آموزش مدل استفاده نشود تا باعث کاهش دقت مدل نشود ولی به‌علت خواست مسئله این ستون از داده را دست نخورده باقی می‌گذاریم.

- داده‌های کلاس‌بندی: برای حذف داده‌های کلاس‌بندی می‌توان از متدهایی مانند *rare category handling* که داده‌هایی که کمتر از k بار تکرار شده‌اند را حذف می‌کند استفاده کرد اما باتوجه به کم بودن مجموعه داده‌ی آموزشی این کار سنخیت ندارد پس داده‌های کلاس‌بندی را نیز دست نخورده باقی می‌گذاریم.

درمجموع از میان ۱۱۵ داده‌ی آزمایشی ۷ تای آن حذف و ۱۰۸ داده باقی می‌ماند.

4. رمزگذاری ویژگی‌های دسته‌ای

در این بخش از پیش‌پردازش داده‌ها، هدف ما تبدیل ویژگی‌های دسته‌ای (categorical features) به داده‌های عددی است، به طوری که برای مدل‌های یادگیری ماشین قابل فهم و پردازش باشند. بسته به نوع داده‌های دسته‌ای، از دو روش مختلف استفاده شده است.

۱. رمزگذاری برچسبی (Label Encoding) برای داده‌های دودویی (Binary Categorical Data): برای ویژگی‌هایی که فقط دو مقدار ممکن دارند مثل ("Yes"/"No") یا ("True"/"False") از *Label Encoding* استفاده شده است. در این پروژه، ویژگی *Brick* از این نوع است.

۲. رمزگذاری *One-Hot* برای ستون‌های اسمی (Nominal Categorical Columns): برای ویژگی‌هایی مانند *Neighborhood* که دارای چندین مقدار منحصربه‌فرد هستند (و ترتیب خاصی بین آن‌ها وجود ندارد)، از رمزگذاری *One-Hot* استفاده شده است. این روش برای هر دسته یک ستون جدید می‌سازد و مقدار آن را با *true* یا *false* مشخص می‌کند.

5. نرمال‌سازی داده‌ها

در این پروژه، از *Min-Max Scaling* استفاده شده است تا مقادیر هر ویژگی عددی بین بازه $[۰, ۱]$مقیاس‌بندی شود. این روش طبق فرمول زیر کار می‌کند:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

مهم است که برای داده‌های آزمون، فقط از transform استفاده شود نه fit_transform، تا از نشت داده یا data leakage جلوگیری شود. مقادیر مینیمم و ماکسیمم باید فقط از داده‌های آموزش استخراج شوند.

2. تحلیل داده‌ها و نمودارها

برای بررسی ارتباط بین ویژگی‌های مختلف و متغیر هدف (قیمت ملک)، از دو نوع نمودار استفاده شده است:

1. نمودارهای پراکندگی

چهار نمودار پراکندگی روابط بین برخی از ویژگی‌ها و قیمت را به‌صورت بصری نمایش می‌دهند:

الف) رابطه‌ی بین مساحت (SqFt) و قیمت:

- این نمودار نشان می‌دهد که با افزایش مساحت خانه، قیمت نیز معمولاً افزایش می‌یابد.
- الگوی صعودی در داده‌ها مشاهده می‌شود که نشان‌دهنده همبستگی مثبت بین این دو ویژگی است.
- این رابطه در ماتریس همبستگی نیز تأیید می‌شود (ضریب همبستگی: ۰.۵۶).

ب) رابطه‌ی بین تعداد اتاق خواب (Bedrooms) و قیمت:

- رابطه مستقیم اما با پراکندگی زیاد دیده می‌شود.
- قیمت‌ها برای تعداد مشخصی از اتاق خواب‌ها متنوع هستند که نشان می‌دهد تنها تعداد اتاق خواب عامل تعیین‌کننده قیمت نیست.
- ضریب همبستگی متوسط: ۰.۵۱

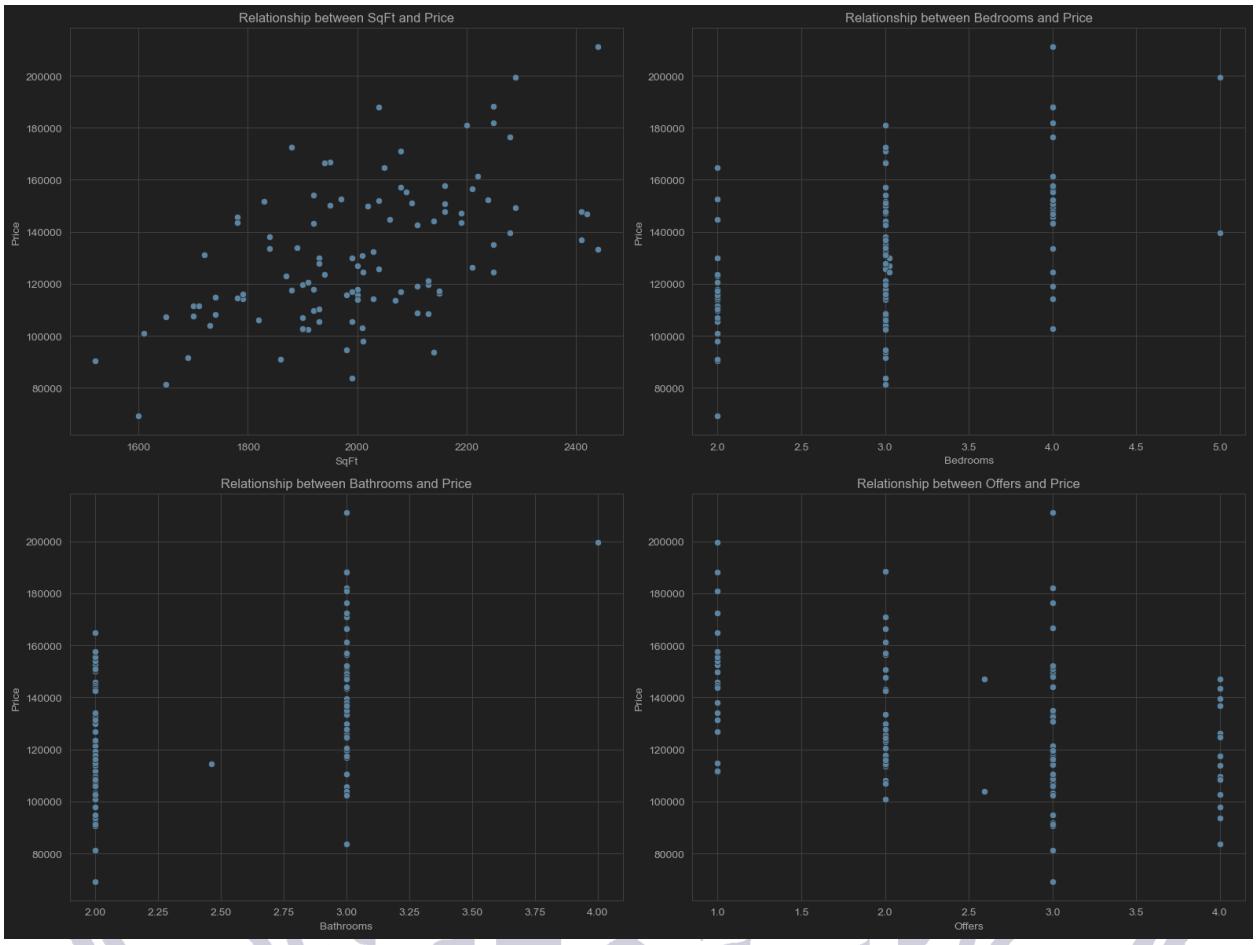
ج) رابطه بین تعداد سرویس بهداشتی (Bathrooms) و قیمت:

- رابطه مشابه با اتاق خواب‌ها دیده می‌شود. قیمت‌ها در سطوح مختلف سرویس بهداشتی متفاوت هستند.
- اما روندی نسبی به سمت بالا وجود دارد.

- ضریب همبستگی: ۰.۴۹

د) رابطه‌ی بین تعداد پیشنهادات (Offers) و قیمت:

- برخلاف موارد قبلی، رابطه معکوس ضعیفی بین تعداد پیشنهادات و قیمت دیده می‌شود.
- قیمت‌های بالاتر ممکن است پیشنهادهای کمتری داشته باشند یا بالعکس.
- ضریب همبستگی منفی: ۰.۳۷-



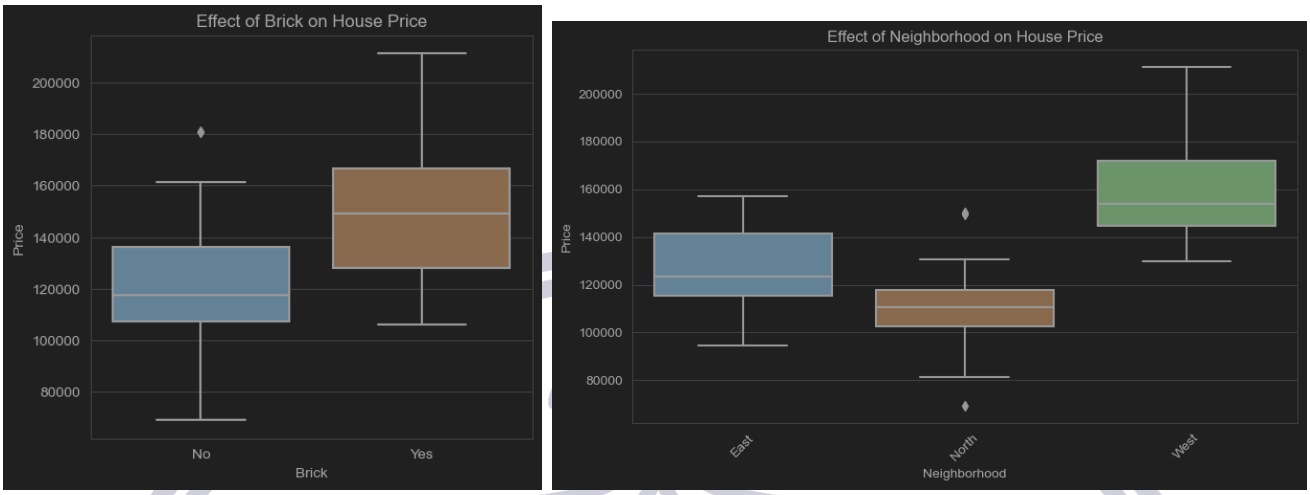
شکل ۲-۱ نمودار روابط بین ویژگی‌ها و قیمت با استفاده از تابع scatterplot

ه) رابطه‌ی بین آجر (Brick) و قیمت:

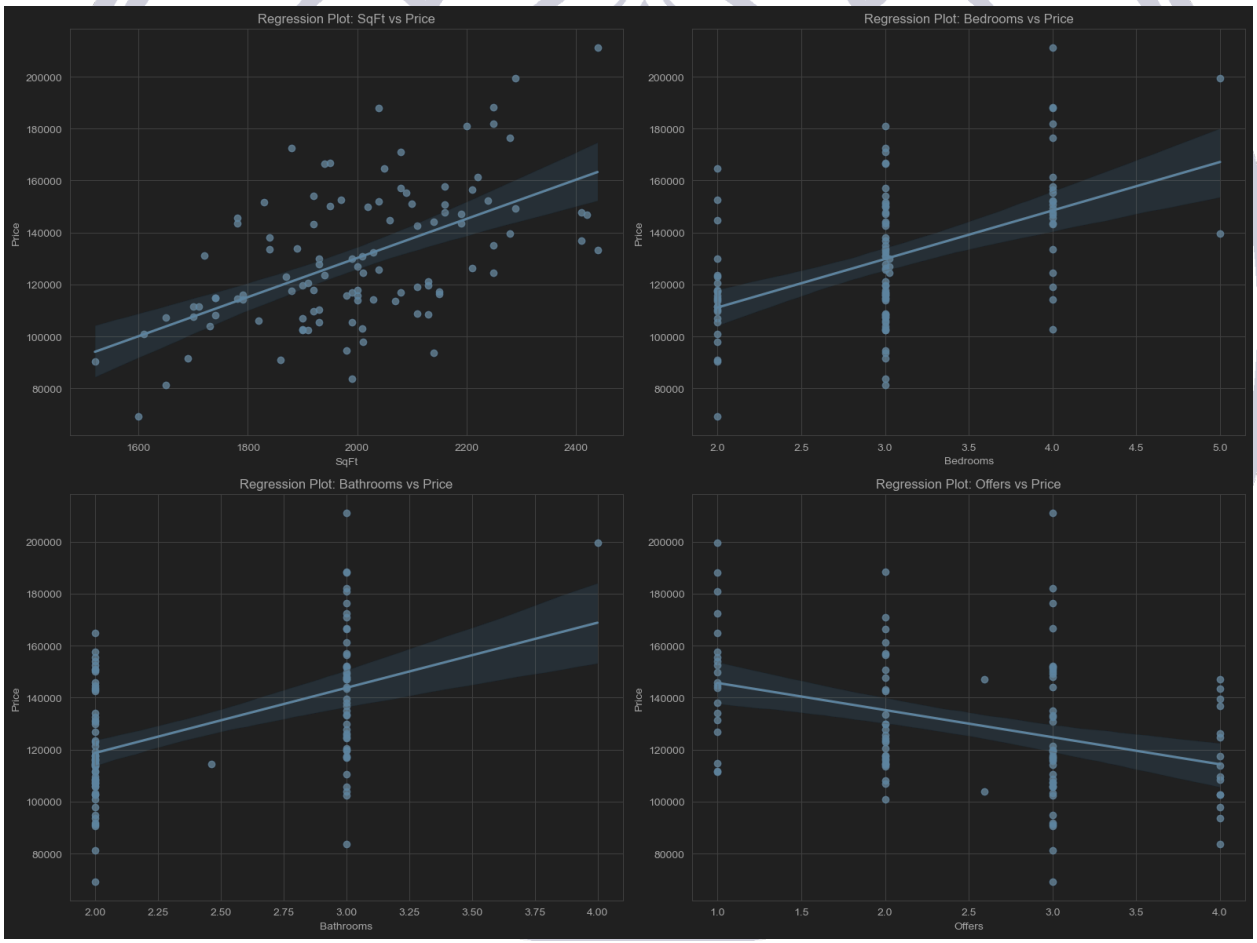
- نمودار جعبه‌ای نشان می‌دهد که خانه‌هایی که با آجر ساخته شده‌اند، به‌طور متوسط قیمت بالاتری دارند.
- میانه (Median) قیمت برای خانه‌های آجری بالاتر است.
- بازه interquartile (IQR) گسترده‌تر است که نشان می‌دهد قیمت این خانه‌ها پراکندگی بیشتری دارد.
- در گروه خانه‌های بدون آجر، مقدار پرت (outliers) در بالاتر از بازه دیده می‌شود.

و) رابطه‌ی بین محله (Neighborhood) و قیمت:

- محله *West* دارای بالاترین میانه قیمت و بیشترین بازه قیمتی است.
- محله *North* کمترین قیمت‌ها را دارد و همچنین چند مقدارهای پرت (outliers) بالا و پایین نیز در آن دیده می‌شود.
- محله *East* در میانه قرار دارد اما نسبت به *West* قیمت‌های پایین‌تری دارد.
- درکل خانه‌های واقع در محله‌ی *West* بسیار گران‌تر هستند.



شکل ٤-١نمودار جعبه‌ای تاثیر آجری بودن خانه و محله‌ی آن بر قیمت خانه



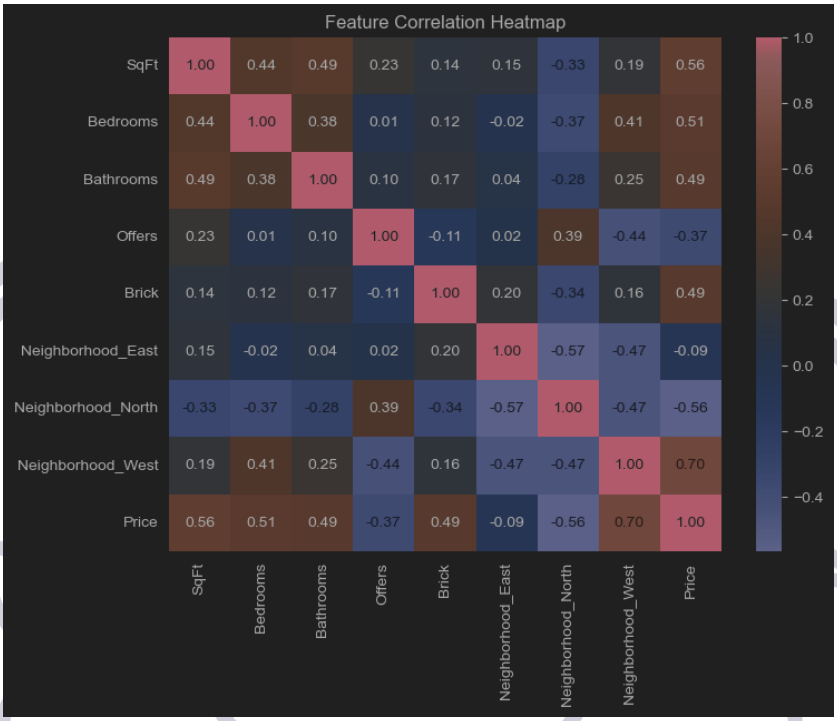
شکل ۳-۱ نمودار خطی تغییرات قیمت با هر ویژگی با استفاده از تابع regplot

2. ماتریس همبستگی

این نمودار به‌صورت عددی و بصری نشان می‌دهد که کدام ویژگی‌ها بیشترین رابطه را با قیمت دارند:

- بیشترین همبستگی مثبت با قیمت مربوط به محله غربی (Neighborhood_West) است که می‌تواند نشان‌دهنده ارزش‌گذاری بالای این محله باشد.

- همچنین محله شمالی (Neighborhood_North) رابطه منفی قوی با قیمت دارد.
- مساحت، تعداد اتاق و سرویس بهداشتی هم نقش مؤثری در تعیین قیمت دارند، اما نه به اندازه‌ی محله.



شکل ۵-۱ نمودار همبستگی بین ویژگی‌ها و قیمت با استفاده از تابع heatmap

با توجه به نمودارها و ماتریس همبستگی، می‌توان ویژگی‌های زیر را برای مدلسازی قیمت خانه به‌عنوان مهم‌ترین عوامل در نظر گرفت:

- Neighborhood_West
- SqFt
- Bathrooms
- Brick

ویژگی‌هایی مانند Offers و Neighborhood_North نیز می‌توانند به‌عنوان عوامل منفی تأثیرگذار در مدل لحاظ شوند.

3. پیاده‌سازی و آموزش مدل

برای فرآیند آموزش مدل با استفاده از گرادیان کاهشی(Gradient Descent)، یک کلاس به نام LinearRegressionScratch طراحی شده است. این کلاس تمام مراحل آموزش، پیش‌بینی، و رسم نمودار خطای آموزش را به‌صورت دستی انجام می‌دهد.

پارامترهای کلاس LinearRegressionScratch:

- learning_rate: نرخ یادگیری برای گرادیان کاهشی.
- epochs: حداکثر تعداد دوره‌های آموزشی.
- Tolerance: آستانه‌ای برای تشخیص تغییرات جزئی در خطا جهت توقف زودهنگام.
- Patience: تعداد دوره‌هایی که اگر تغییر خطا کمتر از tolerance باشد، آموزش متوقف می‌شود.
- W: وزن‌ها (پارامترهای مدل)، به‌صورت آرایه صفر مقداردهی می‌شوند.
- b: بایاس (عدد ثابت).
- loss_history: لیستی برای ذخیره مقدار RMSE در هر دوره.

1. compute_rmse(y_true, y_pred)

تابعی برای محاسبه ریشه میانگین مربع خطا (RMSE) بین مقادیر واقعی و پیش‌بینی‌شده.

2. fit(X_train, y_train)

تابعی برای آموزش مدل با استفاده از الگوریتم گرادیان کاهشی:

۱. ابتدا وزن‌ها و بایاس مقداردهی اولیه می‌شوند.
۲. در هر دوره(epoch) :
- مقادیر پیش‌بینی‌شده y_pred محاسبه می‌شوند.
- RMSE محاسبه شده و در لیست loss_history ذخیره می‌شود.
- در صورتی که تغییر RMSE نسبت به دوره قبل کمتر از tolerance باشد، شمارنده no_change_count افزایش می‌یابد.
- اگر این شمارنده به patience برسد، توقف زودهنگام (early stopping) رخ می‌دهد.
- سپس گرادیان وزن‌ها و بایاس محاسبه شده و به‌روزرسانی می‌شوند.
- هر ۱۰۰ دوره، مقدار RMSE نمایش داده می‌شود.

3. predict(X_test)

تابعی برای پیش‌بینی قیمت با استفاده از وزن‌ها و بایاس آموزش‌دیده:

$$y\hat{} = X \cdot W + b$$

4. plot_loss()

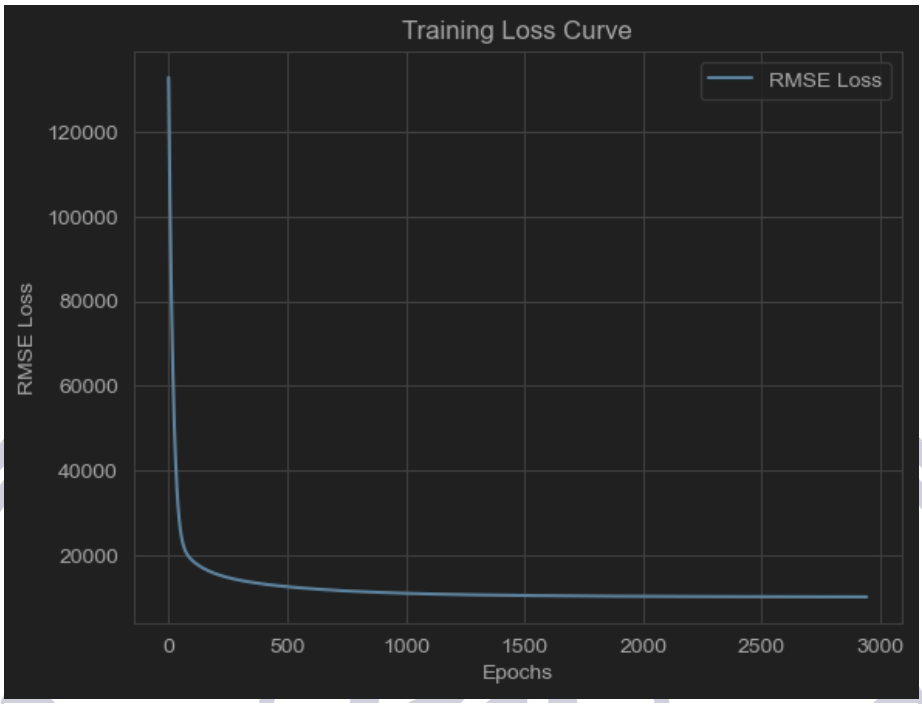
نموداری از تغییرات RMSE در طول دوره‌های آموزشی رسم می‌کند.

مدل با نرخ یادگیری ۰.۰۱ و تحمل خطای ۰.۰۷ آموزش داده می‌شود. حداکثر ۱۰,۰۰۰ دوره ولی با شرط توقف زودهنگام پس از ۲۰ دوره‌ی بی‌تغییر در RMSE. تنظیم این پارامترها با آزمون و خطا و اجرای چندباره‌ی الگوریتم بدست آمده و بهترین جواب را در پی داشت. در نهایت مدل بدست آمده روی داده‌های تست آزمایش شده و مقدار RMSE بر روی داده‌های آزمون نمایش داده می‌شود.

4. نتیجه‌ی نهایی

مدل رگرسیون خطی پیاده‌سازی‌شده با استفاده از الگوریتم گرادیان کاهشی، توانست به‌صورت پیوسته و مؤثر مقدار خطای RMSE را در طول آموزش کاهش دهد. همان‌طور که در نمودار شکل ۱.۶ مشاهده می‌شود، منحنی خطا روندی نزولی و هموار دارد و از مقدار اولیه حدود ۱۳۲,۸۳۶ در اولین تکرار، به حدود ۱۰,۱۶۵ در انتهای آموزش رسید.

- استفاده از مکانیسم توقف زودهنگام (Early Stopping) باعث شد که مدل به‌جای اجرای تمام ۱۰,۰۰۰ دوره، در دوره ۲۹۴۳ آموزش را متوقف کند. این موضوع از افزایش بیش از حد پیچیدگی مدل و Overfitting جلوگیری می‌کند و زمان آموزش را کاهش می‌دهد.
- مقدار خطای RMSE بر روی داده‌های تست برابر با ۹۵۸۴.۲۷ به‌دست آمده که نشان‌دهنده‌ی عملکرد قابل قبول مدل در تعمیم به داده‌های جدید است.
- کاهش شدید و سریع RMSE در ابتدای آموزش، و سپس کاهش آهسته‌تر، نشان می‌دهد که مدل به خوبی به بهینه‌ی محلی نزدیک شده است.



شکل ٦-١ نمودار تغییرات خطای RMSE در طول آموزش مدل

2. سوال دوم: حدس تابع

هدف این پروژه، آموزش یک مدل رگرسیون چندجمله‌ای سه‌بعدی برای پیش‌بینی مقدار تابع $f(x, y, z)$ بر اساس داده‌های موجود در فایل اکسل train.xlsx است. این مدل از جمله‌های چندجمله‌ای مرتبه سوم و عبارات تعاملی (interaction terms) استفاده می‌کند.

1. ساختار داده‌ها

داده‌های ورودی شامل ۴ ستون هستند:

x, y, z: متغیرهای مستقل

F: مقدار هدف (تابع وابسته‌ای که باید پیش‌بینی شود)

2. فرم تابع رگرسیون

مدل ما از یک تابع چندجمله‌ای از درجه ۳ استفاده می‌کند که شکل کلی آن به صورت زیر است:

$$
\begin{aligned}
f(x, y, z) = b &+ w1*x + w2*x^2 + w3*x^3 + \\
&w4*y + w5*y^2 + w6*y^3 + \\
&w7*z + w8*z^2 + w9*z^3 + \\
&w10*xy + w11*x^2y + w12*xy^2 + \\
&w13*xz + w14*x^2z + w15*xz^2 + \\
&w16*yz + w17*y^2z + w18*yz^2 + \\
&w19*xyz
\end{aligned}
$$

3. پیاده‌سازی

1. بارگذاری مجموعه داده

```
df = pd.read_excel('train.xlsx')

x = df['x'].values
y = df['y'].values
z = df['z'].values
f = df['F(x, y, z)'].values  # Target
```

داده‌ها از فایل اکسل بارگذاری شده و به آرایه‌های NumPy تبدیل می‌شوند.

2. ساخت ماتریس ویژگی‌ها

```
X = np.column_stack([
    np.ones(len(x)),    # Bias term (b)
    x, x**2, x**3,      # x terms
    y, y**2, y**3,      # y terms
    z, z**2, z**3,      # z terms
    x*y, x**2*y, x*y**2,  # xy interaction
    x*z, x**2*z, x*z**2,  # xz interaction
    y*z, y**2*z, y*z**2,  # yz interaction
    x*y*z  # xyz interaction
])
```

در این بخش، برای هر نمونه‌ی آموزشی، یک بردار ۲۰-بعدی از ویژگی‌ها ساخته می‌شود که شامل تمام ترکیب‌های ممکن از x, y, z تا درجه سوم است. این فرآیند به صورت دستی انجام شده است تا از کتابخانه‌هایی مانند PolynomialFeatures استفاده نشود.

3. آموزش مدل با استفاده از معادله نرمال(Normal Equation)

در اینجا وارد قسمت اصلی حل مسئله می‌شویم که از یکی از مفاهیم جبر خطی به نام عبارت نرمال پیروی می‌کند:

```
W = np.linalg.inv(X.T @ X) @ X.T @ f
```

توضیح معادله نرمال:

در مدل رگرسیون خطی، می‌خواهیم ضرایب W را طوری بیابیم که حاصل ضرب $X.W$ به f نزدیک باشد. این مسئله به صورت ریاضی به صورت کمینه‌سازی تابع هزینه (error) تعریف می‌شود:

$$\min_{W} \|XW - f\|^2$$

برای حل این مسئله‌ی بهینه‌سازی، مشتق تابع هزینه را صفر می‌کنیم و به فرمول معروف Normal Equation می‌رسیم:

$$W = (X^T X)^{-1} X f$$

این معادله مستقیماً یک جواب تحلیلی برای ضرایب W ارائه می‌دهد، بدون نیاز به الگوریتم‌های تکراری مانند گرادیان نزولی.

4. مزایای استفاده از معادله نرمال

- تحلیلی است، یعنی برخلاف گرادیان نزولی، نیازی به تنظیم نرخ یادگیری یا تعداد تکرار ندارد.
- برای تعداد ویژگی‌های کم (کمتر از چند هزار) بسیار سریع و دقیق است.
- در تمرین ما که فقط ۲۰ ویژگی دارد، کاملاً مناسب است.

5. نتیجه‌ی نهایی

در این تمرین، یک مدل رگرسیون چندجمله‌ای مرتبه سوم برای پیش‌بینی f(x,y,z) پیاده‌سازی شد. مدل با استفاده از معادله نرمال و بدون استفاده از کتابخانه‌های یادگیری ماشین آموزش داده شد.

معادله نرمال، با حل مستقیم دستگاه معادلات، ضرایب بهینه را محاسبه کرد و دقت بالایی در پیش‌بینی‌ها حاصل شد. ضرایب به‌دست‌آمده ذخیره شدند تا در آینده بدون نیاز به بازآموزی، بتوان از آن‌ها استفاده کرد.

این پروژه نشان داد که با ابزارهای پایه‌ای مانند NumPy و مفاهیم جبر خطی می‌توان یک مدل دقیق و کاربردی برای رگرسیون چندمتغیره ساخت.

3. سوال سوم: کوهنوردان در جستجوی مینیموم جهانی

در این پروژه، فرض بر این است که دو کوهنورد قصد دارند از یک کوه (سطح تابع هزینه) پایین بیایند تا به پایین‌ترین نقطه یا مینیمم جهانی برسند. هر کدام از این کوهنوردان نشان‌دهنده یکی از الگوریتم‌های بهینه‌سازی در یادگیری ماشین هستند:

- کوهنورد اول *SGD (Stochastic Gradient Descent)*: که قدم‌های تصادفی، پراکنده و سریع برمی‌دارد و با اطلاعات محدود از اطراف حرکت می‌کند.
- کوهنورد دوم *Batch Gradient Descent*: که ابتدا محیط اطراف را کامل بررسی کرده، سپس با قدم‌هایی آهسته اما دقیق حرکت می‌کند.

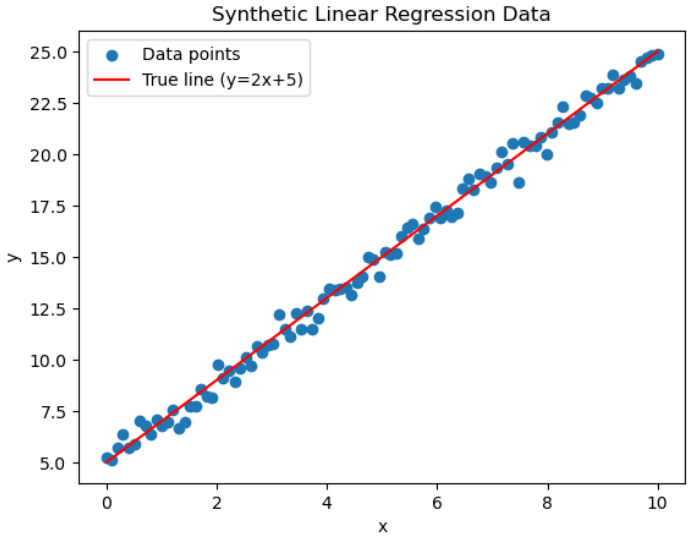
هدف این پروژه، شبیه‌سازی مسیر حرکت این دو کوهنورد و مقایسه عملکرد آن‌ها در رسیدن به مینیمم جهانی است.

1. پیاده‌سازی مرحله‌ای

1. ساخت داده مصنوعی (Data Creation)

برای شبیه‌سازی یک مسئله رگرسیون خطی ساده، داده‌هایی با نویز کم تولید می‌کنیم:

- رابطه اصلی: $y = 2x + 5 + noise$
- داده‌ها شامل ۱۰۰ نقطه هستند که با نویز کمی پراکنده شده‌اند.
- این داده‌ها به دسته‌های کوچک(۱۰ تایی) تقسیم می‌شوند تا برای Mini-Batch GD آماده شوند.
- تابع هزینه مورد استفاده، میانگین مربعات خطا (MSE) است.



شکل ۱-۳ نمایش داده‌های تولید شده روی نمودار

2. پیاده‌سازی سه الگوریتم بهینه‌سازی

۱. Batch Gradient Descent:

- در هر تکرار، گرادیان بر اساس کل داده‌ها محاسبه شده و وزن‌ها به‌روزرسانی می‌شوند.
- w بدست آمده: 2.43774127778603
- b بدست آمده: 2.048176441154291
- زمان اجرا: ۰.۶۳ میلی ثانیه

۲. Stochastic Gradient Descent (SGD):

- در هر تکرار، تنها از یک نمونه تصادفی برای محاسبه گرادیان و به‌روزرسانی وزن‌ها استفاده می‌شود.
- w بدست آمده: 2.668692260632476
- b بدست آمده: 1.9757374621105868
- زمان اجرا: ۰.۷۹ میلی ثانیه

۳. Mini-Batch Gradient Descent:

- داده‌ها به گروه‌های کوچکتر (مینی‌بچ) تقسیم می‌شوند.
- در هر تکرار، گرادیان بر اساس یک مینی‌بچ محاسبه و وزن‌ها به‌روزرسانی می‌شوند.
- w بدست آمده: 1.9776948446169698
- b بدست آمده: 4.981939614734239
- زمان اجرا: ۵.۶۱ میلی ثانیه

3. ثبت معیارها

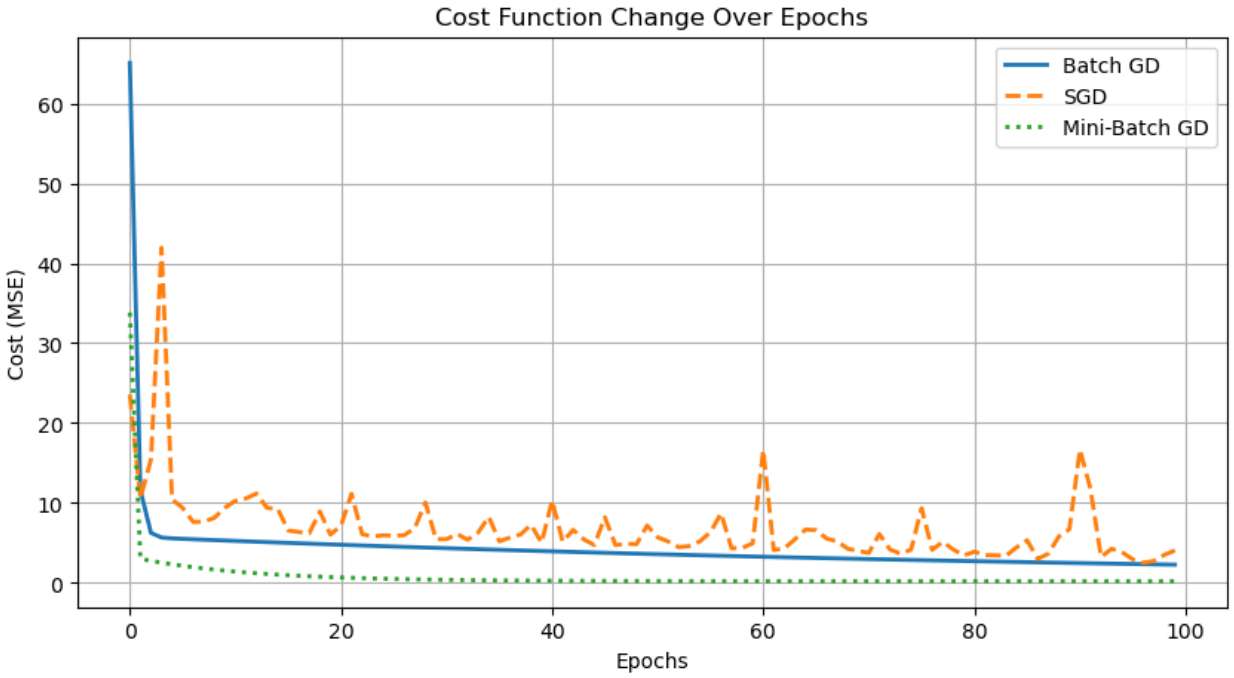
برای هر الگوریتم، موارد زیر ذخیره می‌شوند:

- مقدار تابع هزینه در هر epoch
- مسیر تغییر وزن‌ها (برای ترسیم روی نمودار کانتور)
- تعداد کل به‌روزرسانی‌ها

2. تحلیل بصری و عددی

1. نمودار تابع هزینه

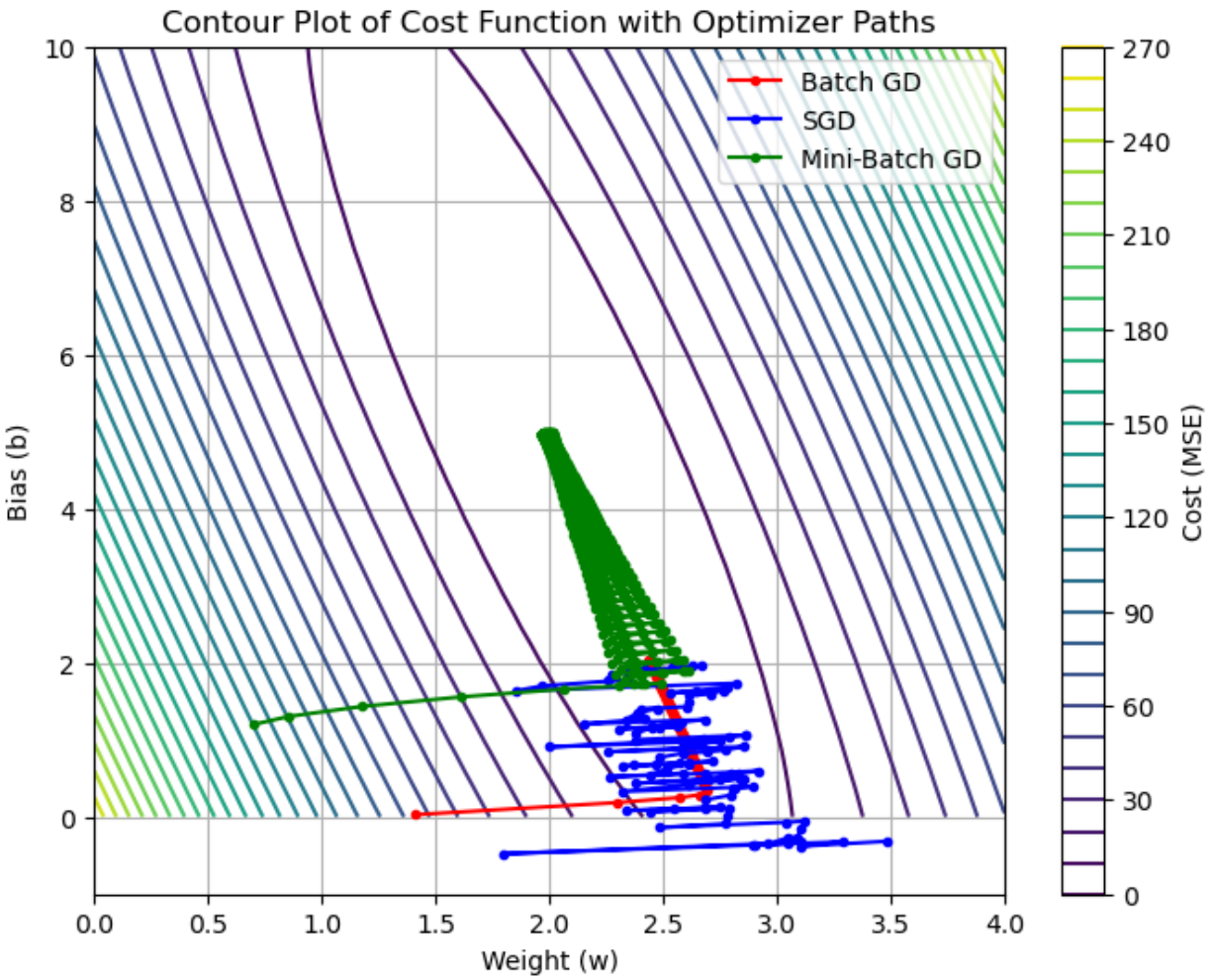
- با استفاده از matplotlib، تغییرات تابع هزینه (MSE) برای هر سه الگوریتم در طول epoch ها ترسیم شده است.
- هدف: بررسی سرعت همگرایی و دقت هر الگوریتم در کاهش خطا.



شکل ۲-۳ نمودار تغییرات تابع هزینه در اجرای سه الگوریتم

2. نمودار کانتور (Contour Plot)

- تابع هزینه بر حسب وزن (w) و بایاس (b) ترسیم شده است.
- مسیر حرکت وزن‌ها برای هر الگوریتم با رنگ متفاوت نشان داده شده:
  - قرمز: Batch GD
  - آبی: SGD
  - سبز: Mini-Batch GD



شکل ۳-۳ نمودار کانتور تابع هزینه‌ی هر الگوریتم همراه با مسیر بهینه‌ساز

- این نمودار نشان می‌دهد که هر الگوریتم چگونه در فضای پارامترها به سمت مینیمم حرکت می‌کند.

3. مقایسه سرعت اجرا

- تعداد کل به‌روزرسانی‌ها در هر الگوریتم:
  - **SGD:** بیشترین تعداد به‌روزرسانی اما سریع‌ترین اجرا
  - **Batch GD:** کمترین به‌روزرسانی ولی سنگین‌ترین پردازش در هر مرحله
  - **Mini-Batch GD:** تعادل بین دقت و سرعت

3. بررسی عملکرد در تابع رزنبروک(Rosenbrock Function)

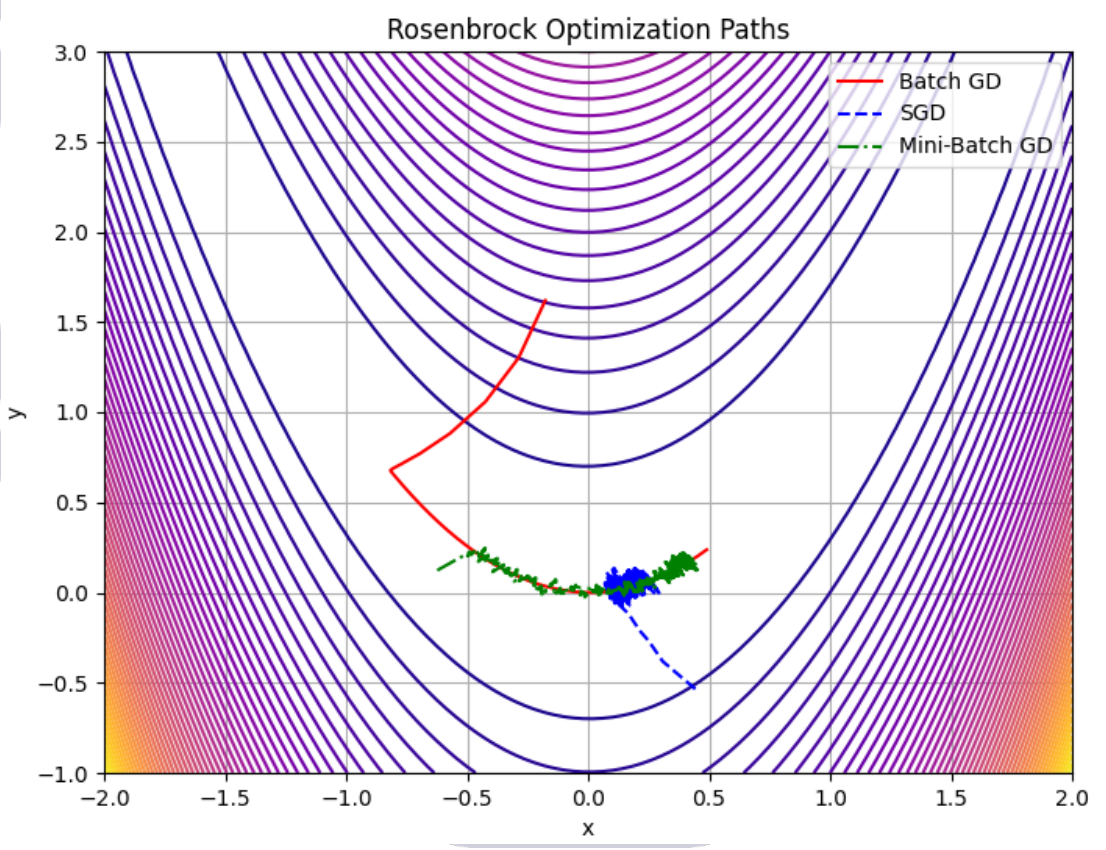
1. تابع Rosenbrock

- تابعی غیر محدب با یک دره باریک که بهینه‌سازی در آن بسیار دشوارتر از حالت خطی است.
- $f(x,y)=(1-x)^2+100(y-x^2)^2$
- گرادیان این تابع برای بهینه‌سازی استفاده شد و سه الگوریتم روی آن اجرا شدند.

2. نتایج

- **Batch GD:** آهسته و پایدار ولی گاهی در نقاط نامناسب گیر می‌کند.
- **SGD:** سریع ولی مسیر بسیار نوسانی دارد.
- **Mini-Batch GD:** با تنظیم درست نویز، بهترین تعادل را بین سرعت و دقت نشان داد.

نمودار مسیر هر الگوریتم در فضای تابع *Rosenbrock* نشان‌دهنده چالش‌های این بهینه‌سازی و تفاوت عملکرد الگوریتم‌ها است.



شکل ٤-٣ مسیر بهینه‌سازی با استفاده از تابع رزنبروک

4. نتیجه‌گیری کلی

- *SGD* برای داده‌های بزرگ یا محیط‌های نویزی مناسب‌تر است و سرعت بالایی دارد.
- *Batch GD* دقت بیشتری دارد اما در داده‌های بزرگ، زمان‌بر است.

- *Mini-Batch GD* بهترین انتخاب برای تعادل بین سرعت و دقت، به‌ویژه در مدل‌های یادگیری عمیق است.

این پروژه نشان می‌دهد که بسته به نوع داده و هدف مسئله، باید الگوریتم مناسب را انتخاب کرد و هیچ الگوریتمی به‌طور مطلق برتر نیست.

5. پاسخ به سوالات

1. توضیح دهید چه زمانی و چرا SGD بهتر از Batch GD عمل می‌کند.

زمانی که داده بسیار بزرگ و پیچیده باشد یا داده به صورت online برسد، روش SGD با به‌روزرسانی سریع و استفاده کم از حافظه عملکرد بهتری دارد.

2. در مورد مزایا و معایب سرعت و پایداری هر دو روش توضیح دهید.

- Batch GD: پایدار ولی کند
- SGD: سریع ولی ناپایدار و نوسان‌دار

3. در چه موقعیت‌هایی ترجیح می‌دهید از SGD به جای Batch GD استفاده کنید؟

در حجم داده بزرگ، پردازش real-time یا زمانی که حافظه محدود باشد از SGD استفاده می‌کنیم.

4. چگونه گرادیان نزولی مینی‌بچ می‌تواند این دو را متعادل کند؟

Mini-Batch با استفاده از دسته‌های کوچک داده هم سرعت مناسبی دارد (نسبت به Batch) و هم نوسان زیادی ندارد (نسبت به SGD).

4. لینک‌های مرتبط

ریپازیتوری گیت‌هاب:

https://github.com/AMIN-nsri/machine-learning-regression.git

5. منابع

1. یادگیری ماشین و طبقه‌بندی (Machine Learning & Classification)

[1] T. M. Mitchell, *Machine Learning*, New York, NY, USA: McGraw-Hill, 1997.

[2] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2019.

[3] Andrew Ng, "Machine Learning," Coursera, Online Course, 2011. Available: https://www.coursera.org/learn/machine-learning

2. مدیریت داده‌های نامتوازن و انتخاب ویژگی (Handling Imbalanced Data & Feature Selection)

[4] H. He and Y. Ma, *Imbalanced Learning: Foundations, Algorithms, and Applications*, Hoboken, NJ, USA: Wiley, 2013.

[5] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

3. پردازش داده و کار با پایتون (Data Processing & Python Programming)

[6] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*, Sebastopol, CA, USA: O'Reilly Media, 2016.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York, NY, USA: Springer, 2006.

[8] ChatGPT