**Machine Learning - Homework1**

Linear Regression

32184801 Ha Seung Won

0 Abstract

이번 과제에서는 Linear Regression 모델을 직접 구현하여 Linear Regression의 프로세스를 이해하는 것을 목표로 한다. 이론적인 부분을 기반으로 하여 직접 코드를 작성함으로써, Linear Regression의 기본 원리와 구현 방법을 더 깊이 이해할 수 있다. 이 과제를 통해 선형 회귀에 대한 이론과 코드 작성 능력을 향상시키고, 데이터 분석 및 예측 모델 구축에 필요한 기초를 다지는 데 도움이 될 것이다. 기본적인 이론을 설명하는 Relate Work, 코드 설명과 구현 결과는 Implementation에서 정리하였고 마지막으로 느낀 점으로 마무리하였다.

1 Relate Work

* 1. Linear Regression

Linear Regression에서 기본적인 프로세스는 아래와 같다.

1. **Hypothesis Function 지정**
2. **Model train**
3. **Predict**

Hypothesis Function 을 지정하면 처음에 훈련할 모델에 대한 알고리즘을 선택을 하고 파라미터를 초기화한다. 그 이후에는 그 모델을 학습을 해 나가면서 최적의 파라미터 값을 찾아나간다. 그리고 마지막으로 Predict로 모델을 활용하여 train 데이터가 아닌 새로운 데이터로 값을 예측한다. 이런 프로세스를 통해서 Linear Regression 과 전반적인 머신러닝 모델의 학습 과정이 진행된다.

* **Error**

도표, 라인, 그래프, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Figure 1 데이터 시각화

Error 란 모델이 예측한 값과 실제 값의 차 이를 Error 혹은 Cost라고 한다. 그리서 이런 값들을 최소화하는 게 모델 훈련의 목적이라고 이야기를 할 수 있다. 그렇다면 이런 Error를 구하는 방식은 Cost Function(J)에 의하여 정해지는데, Cost Function은 input은 모델의 파라미터 값을 통해서 구해지며 J(, )라고 표시한다. 이런 함수들의 예시로 MSE(Mean Squared Error), Cross-Entropy 등등이 있다.

* 1. Mean Squared Error

폰트, 화이트, 서예, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

Figure 2 MSE Formula

MSE는 Cost Function의 대표적인 예시로서 말 그대로 Error의 제곱의 평균이다. 식을 보자면 먼저 Hypothesis Function(h)를 이야기한다. 기존에 정했던 예측 모델 함수를 input 값 x를 통해서 현제 모델의 예측값이 나오고 실제 target 데이터와의 차이를 구한다. 그리고 이것을 음수 양수 관계없이 사용하기 위해서 제곱을 하며 이것을 모든 input 데이터에서 구하고 이것의 평균을 구하는 것이 MSE의 방식이다. 여기서 2로 나눠주는 이유는 나중에 계산을 편하게 하기 위해서인데 변화량을 구하려고 미분을 추가적으로 수행하기 때문이다.

폰트, 텍스트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

Figure 3 Cost Function

그래서 결국에는 이런 **Cost Function**의 값을 최소화 하는 파라미터 값을 찾는 게 모델 훈련의 과정이 된다. 이것을 **Recall이라고** 한다.

1.3 Learning Algorithms

* **Least Square**

도표, 텍스트, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

Figure 4 Hypothesis & Cost Function

이제 Error값을 계산하는 함수인 Cost Function에서 최솟값이 되도록 하는 파라미터를 구하는 방식에 대하여 알아보겠다. 먼저 일반적으로 생각하면 Cost Function을 표현한 Figure 4 오른쪽 그림을 보면 그래프의 모양이 아래로 볼록한 모양이다. 이건 Convex 한 모양이라고 할 수 있다. (반대는 Concave 하다고 한다.) 그렇다면 이런 함수적 특성을 본다면 값이 최소가 되려면 기울기가 0 인 즉 미분하여 값이 0이 되는 부분이 최소가 된다고 생각을 할 수 있다. 그래서 나온 첫 번째 방식이 편미분을 통해서 각각의 함수가 최소가 되는 파라미터의 값을 찾는 방법이다.

* Partial Derivative

텍스트, 폰트, 친필, 라인이(가) 표시된 사진

자동 생성된 설명

Figure 5 MSE Partial derivative

먼저 Cost Function 이 있다고 가정하면 각각 파라미터 𝝧0, 𝝧1에 대하여 각각 편미분을 통해서 식을 전개한다. 그래서 이 계산 과정에서 생기는 식을 통해서 Figure 6 방식으로 전개한다. 이 부분에서 앞서 이야기했던 계산을 편하게 하기 위해 2를 나눠준 이유가 된다.

(argmin 은 어떤 함수를 최소로 만드는 변수를 말한다.)

텍스트, 폰트, 친필, 라인이(가) 표시된 사진

자동 생성된 설명

Figure 6 MSE Partial derivative calculate

편미분을 각각 파라미터를 기준으로 진행하고 그 식에서 convex 함수의 최솟값을 구하기 위해서 미분 값의 0 이 되는 미지수의 값을 구해야 한다. 그래서 이 값이 Error 가 최소가 되는 파라미터 값이 된다.

1.3.1 Normal Equation

스크린샷, 텍스트, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

Figure 7 Represent data & Parameters in matrix

Normal Equation 은 선형대수를 사용하여 최적 x에 대한 공식을 유도하는 방식입니다. Hypothesis Function 을 Matrix 형식으로 계수를 표현하면 위와 같이 표현이 가능하다. Figure 8 예시는 Hypothesis Function에서 단순히 1차 함수고 y 절편(bias)이 없는 상태이니 이것을 그냥 1로 채우고 matrix로 식을 쓰면 위의 행렬과 같다.

폰트, 텍스트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

Figure 8 Normal Equation formula

Ax=b 인 상황에서 A^T  즉 A 행렬의 transpose 한 값을 양변에 곱하고 그걸 다시 묶어서 invertible를 또 양변에 해주면 구하고자 했던 파라미터 값을 구하는 식이 완성된다.

* 증명

친필, 폰트, 화이트, 스케치이(가) 표시된 사진

자동 생성된 설명친필, 스케치, 라인 아트, 그림이(가) 표시된 사진

자동 생성된 설명

Figure 9 Error calculate in matrix

Matrix 방식으로 기존 Hypothesis 와 input 값을 포함하여 같이 표시하면 왼쪽 matrix 형식으로 나온다. 그 이후에 예측값과 target 값을 뺀 error 값을 표시를 하면 오른쪽 그림과 같다. 이제 error를 구하는 방식을 알았다면 이제 error를 통해서 Least Squared Method를 적용하는 게 가능하다. 이전에 matrix 가 아닌 다항 함수에서의 방식에서는 모든 데이터에서 error 값을 제곱을 더한 값을 구하였다. Matrix에서는 ∥Ax-b∥₂²로 그 값을 구하는 게 가능하다.

**∥Ax-b∥₂²** 를 먼저 설명하자만 Ax는 기존의 Hypothesis 함수를 말하고 b는 실제 데이터를 말한다. 그래서 이것을 뺀 것은 기존에 말했던 error를 표시한 것이다. ∥∥ 는 norm을 의미하며 2를 표시한 것은 아래의 2는 각각의 요소들을 전부 제곱을 해준다는 의미이고 위에 있는 2는 이것을 제곱을 해준다는 의미이다. 그래서 이것을 기존에서 MSE 방식에서 error의 제곱을 전부 더하는 부분과 같다.

하지만 , Matrix에서는 단순히 ∥x∥²₂ = x^T  \* x로 표시하는 게 가능하다. 이렇게 MSE를 Matrix 구성에서 구하는 방식은 아래와 같아.

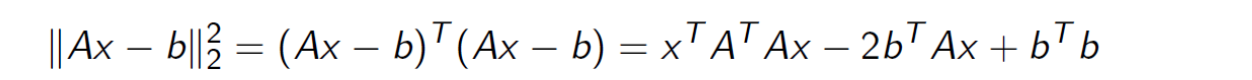


Figure 10 Square Error Calculate

* **Minimize Cost Function**

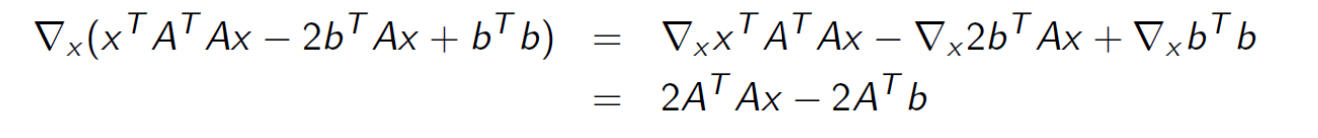


Figure 11 Partial Derivative in Square Error

텍스트, 친필, 폰트, 스케치이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 화이트, 대수학이(가) 표시된 사진

자동 생성된 설명

Figure 12 Matrix derivative formula

LSM 방식으로 Cost Function을 Figure 11 과 같이 만들었다. 이제 값이 최소가 되는 파라미터 값을 찾아야 한다. 그래서 Figure 12에서 파라미터 값을 기준으로 미분을 진행하였고 이후에는 Figure 13의 Matrix의 미분에 대한 특성을 활용하여 식을 정리하면 결국에는 Figure 12에서 나오는 최종의 식이 도출된다. 그렇게 미분을 통해서 최적의 파라미터 값을 찾으려면 결국 미분 값이 0 이 되어야 하므로 식을 간단히 하게 되면 결국에는 Figure 9에서 나왔던 식이 결국에는 도출된다.

* **한계**

Normal Equation에서 10,000 개의 데이터 보다 작은 잡업을 수행하는 데 여기서 많은 수행 시간이 필요하지 않다. 하지만 그 이상은 너무 많은 연산 시간을 필요로 한다.

Normal Equation을 이용하다 보면 non-invertible 문제가 발생하기도 한다. 이런 문제는 보통 중복된 feature 가 존재하거나 데이터의 총 양보다 feature의 양이 너무 많을 때 발생한다. 이런 문제를 해결하기 위해서는 feature를 중복된 feature를 제거하거나 regularization을 통해서 문제를 해결한다.

1.3.2 Gradient Descent

기본적인 프로세스는 cost Function에서 일정 파라미터에 대하여 특정 시작점을 지정하고 점점 이동하면서 목표로 하는 지점까지 이동하는 방식이다. 한국어로 해석하면 경사하강법이라고 이야기를 한다.

라인, 도표, 그래프, 스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Figure 13 Gradient Descent visualization

* Go right, Go left : Θ += - gradient \* a

식을 통해서 알 수 있듯이 파라미터의 값을 일정 gradient 값과 알파 값을 곱한 값을 빼서 파라미터를 업데이트한다. 여기서 생기는 의문은 왼쪽이던 오른쪽이던 값은 방식으로 적용이 된다는 것인데 그림을 보면 왼쪽에서 값이 업데이트된다고 하면 gradient 값이 음수이므로 결국 더해지게 되고 오른쪽에서 시작하면 gradient 값은 양수이므로 값이 줄어든다. 결국 왼쪽이든 오른쪽이든 같은 방식으로 적용이 가능하다.

* **Gradient Descent Process**

텍스트, 폰트, 친필, 라인이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

Figure 14 Gradient Descent Formula

**Iteration** : Iteration은 파라미터가 업데이트되는 횟수를 말한다.

 : 알파값은 learning rate라고도 이야기하고 학습되는 정도를 이야기하며 파라미터가 조정되는 정도를 지정된다.

이제 파라미터를 업데이트할 때는 Iteration을 반복하여 0에 가까워질 때까지 반복한다. 여기서 파라미터값이 여러 개 있다면 각각의 파라미터에 대하여 Partial derivative 편미분을 통해서 각각의 파라미터를 업데이터 해준다. 이때 업데이트는 각각 따로 하는 게 아니라 동시에 업데이트시켜 준다.

1.3.2.1 Gradient Descent Types

원, 텍스트, 폰트, 도표이(가) 표시된 사진

자동 생성된 설명

Figure 15 Different Gradient Descent

**-Batch Gradient Descent**

데이터셋의 전부를 탐색하는 방식으로 정확한 방향을 찾는 게 가능하지만 조금 느리고 Local Minimum 문제가 발생하기 쉽다.

**-Stochastic Gradient Descent**

한 개 한 개 탐색하는 방식으로 데이터를 한 개씩 탐색을 하기 때문에 정확하게 탐색하기 힘들다

**-Mini-Batch Gradient Descent**

가장 많이 쓰이는 방식으로 일정량의 데이터로 분할하여 탐색하는 방식이다. 용어를 정리하자면 **Batch는** 분할한 데이터 샘플을 말하고 **Iteration**은 그 데이터를 통해서 파라미터를 학습한 횟수를 말한다. 그리고 **Epoch**는 이런 한 프로세스를 반복한 횟수를 말한다.

2 Implementation

2-1 Code Comment

Task 1

* Data & Initial Parameter setting

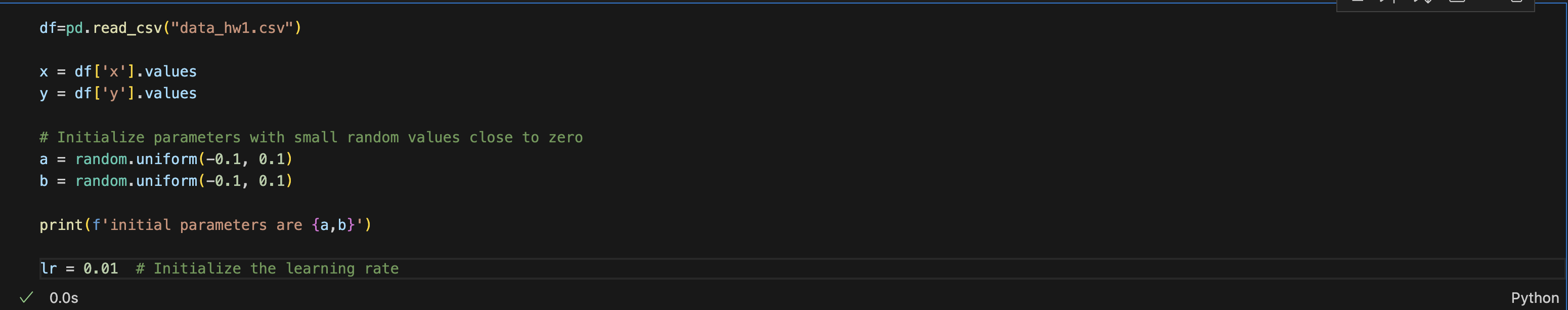
****

Figure 17 Data & Initial Parameter setting code

Pandas 라이브러리를 활용하여 데이터를 DataFrame 형식으로 파일을 읽고 input data ‘x’ 와 target data ‘y’ 변수에 각각 저장하였다. 그리고 초기 파라미터 값은 랜덤하게 초기 설정을 해주었다.

* Model Definition

텍스트, 소프트웨어, 멀티미디어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

2

1

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

3

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

4

Figure 18 Task1 model class define

Figure 17 코드는 Task 1 y=ax+b 모델을 구현한 코드이다. 선언 당시 초기값은 파라미터 두개와 Gradient descent learning 에 필요한 learning rate 로 class 선언이 가능하다.

1. **Function**

Hypothesis Function을 구현한 코드로 클래스의 파라미터를 기준으로 input 값 x 을 기준으로 현재 모델의 예측값을 계산하기 위한 함수이다.

1. **Calculate\_gradient**

Figure 5 그림에서 나왔던 식을 기반으로 각 파라미터의 Gradient 값을 계산하기 위해 필요한 함수이다. 또한 input 데이터를 기반으로 error^2 의 평균을 구하는 MSE 도 이 부분에서 계산이 가능하게 하였다.

1. **Train**

Gradient descent 방식을 활용하여 최적을 파라미터를 찾아나가는 모델 훈련의 과정을 구현한 부분이다. Figure 13 번에 나왔던 식에 따라 gradient 값에서 초기 설정한 learning rate 값을 곱하여 기존의 파라미터 값에서 빼준다.

여기서 조건문을 추가한 이유는 Gradient descent 방식이 Gradient 값이 0에 가까워 질때까지 반복을 이여나간다는 이론적 부분을 직접 확인하기 위해서 처음으로 0 에 가까워지는 연산횟수인 iteration 을 확인하기 위하여 추가를 해주었다.

1. **Mini-batch train**

Mini-batch 훈련 방식은 1.5.1 Gradient Descent Types 에서 설명을 했듯이 batch size 로 데이터를 나눠서 학습하는 코드를 추가로 구현을 해보았다.

* Train code & output
* Batch model train

스크린샷이(가) 표시된 사진

중간 신뢰도로 자동 생성된 설명

Figure 19 Batch model train code

스크린샷, 텍스트, 패턴, 디자인이(가) 표시된 사진

자동 생성된 설명

Figure 20 Batch model train output

* Mini-Batch model train

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Figure 21 Mini-Batch model train code

스크린샷, 텍스트, 패턴이(가) 표시된 사진

자동 생성된 설명

Figure 22 Mini-Batch model train output

* Library model

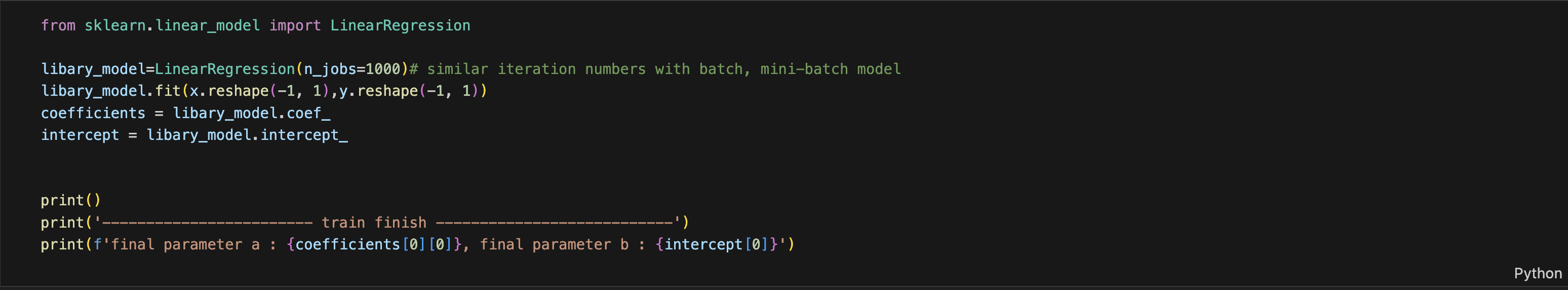


Figure 23 Scikit-learn model train code

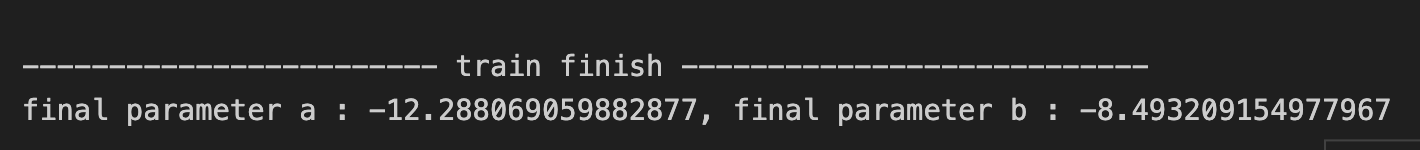


Figure 24 Sciki-learn model train output

Result

|  |  |
| --- | --- |
| Model | Plot |
| Batch | 텍스트, 그래프, 라인, 스크린샷이(가) 표시된 사진  자동 생성된 설명  First zero iterate : 360 |
| Mini-Batch | 그래프, 텍스트, 스크린샷, 라인이(가) 표시된 사진  자동 생성된 설명  First zero iterate : 294 |
| Scikit-learn | 텍스트, 그래프, 스크린샷, 라인이(가) 표시된 사진  자동 생성된 설명 |

결과적으로 Batch, Mini-Batch 모델 모두 파라미터 학습이 잘 이루어진다는 사실을 확인할 수 있다. 하지만 Batch Gradient Descent와 Mini-Batch Gradient Descent의 First zero iterate 횟수를 비교를 해보면 Mini-Batch 모델이 더욱 빨리 수렴한다는 것을 알 수 있다. 이는 batch 사이즈를 적절하게 나누면 모델의 파라미터를 더욱 빠르게 수렴하게 하는 게 가능하다는 것을 보여준다. 그래서 데이터 셋이 큰 경우에는 보통 Mini-Batch 방식을 더 많이 사용하는 이유가 된다.

Task 2

* Normal Equation

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

2

1

Figure 25 Normal Equation Model

Normal Equation 방식을 통해서 파라미터 값을 구하는 방식을 구현한 코드이다. 먼저 데이터를 Matrix 구조로 변경하여서 데이터를 처리하고 그 이후에는 Matrix 의 연산을 통해서 진행하였다.

1. **Convert to Matrix Form**

먼저 구하고자 하는 polynomial regression 함수의 차수가 2차수인 함수를 이야기함으로 matrix 로 값을 표현하면 [1, x, x^2] 값으로 표현을 해야한다 그래서 모들 input 값을 기준으로 matrix 형식으로 input 값을 조정하여 2차원 배열로 구성하였다.

1. **Normal Equation Calculation**

폰트, 텍스트, 화이트, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

Normal Equation 공식의 식을 그대로 구현한 방식으로 Matrix 의 곱 연산을 통해서 계산된 파라미터 Matrix X 값을 theta 변수에 저장하는 형식으로 코드를 구현하였다. Matrix 형식으로 변환된 x 값은 .T 로 transpose를 진행하였고 @ 연산을 통해서 Matrix의 곱을 구현하였으며 numpy 라이브러리에서 제공하는 inverse 함수를 통해서 inverse 도 구현하여서 식을 직접 구현하였다.

* Library model

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Figure 26 Scikit-Learn Library Model

|  |  |
| --- | --- |
| Model | Plot |
| My Model | 텍스트, 스크린샷, 그래프, 라인이(가) 표시된 사진  자동 생성된 설명 |
| Scikit-learn model | 텍스트, 스크린샷, 그래프, 라인이(가) 표시된 사진  자동 생성된 설명 |

2-3 Run Code

Step 1 : Move to the Same Folder

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* + 데이터 파일과 코드를 같은 파일에 위치시킨다.

Step 2 : Define the Python Base Kernel

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* + 커널을 설정하여 코드를 작동할 환경을 설정한다.

Step 3 : Install the Libraries to Run the Code

* Pandas : 데이터를 읽을때 필요하다.
* Random : 첫 파라미터 지정하기 위해서 필요.
* Numpy : Matrix 연산에 필요
* Matplotlib : 결과 시각화를 하기 위해서
* Sklearn : 구현한 모델과 비교하기 위해서
  + 코드를 실행하기 위한 외부 라이브러리를 미리 설치한다.

Step 4: Execute the Cells one by one from top to bottom

스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

* + .ipynb 파일의 특성상 코드를 cell 단위로 실행하는 것이 가능함으로 위에서부터 차례로 cell을 실행하여 전체 코드를 실행한다.

3느낀점

이번 과제를 통해서 Linear Regression 함수를 직접 구현을 해보니 함수가 어떻게 실행되는지 흐름을 정확하게 이해하는 게 가능해서 좋았다. 하지만 코드를 구현하면서 생기는 반복문으로 이론적으로 data set이 늘어나면 Normal Equation의 비효율성이 증가한다는 단점을 임의로 랜덤하게 데이터를 넣어서 파악하기 위해서 노력했지만 데이터가 직접적으로 확인하지 못한 건 조금 아쉬웠다.