텍스트이(가) 표시된 사진

자동 생성된 설명

Logistic Regression

Homework #2

32181928 박찬호

Dankook University

Mobile Systems Engineering

2022 Fall

**Contents 목차**

**Ⅰ 서론 01**

**Ⅱ 본론 03**

**Ⅲ 결론 09**

Logistic Regression 01

Cost Function 03

행렬 곱으로의 전환 03

코드 설명 05

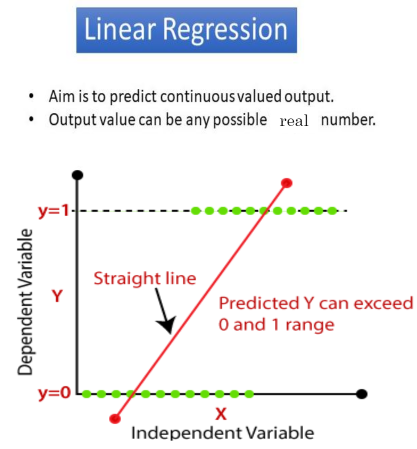
결과 분석 07

최종 결과 09

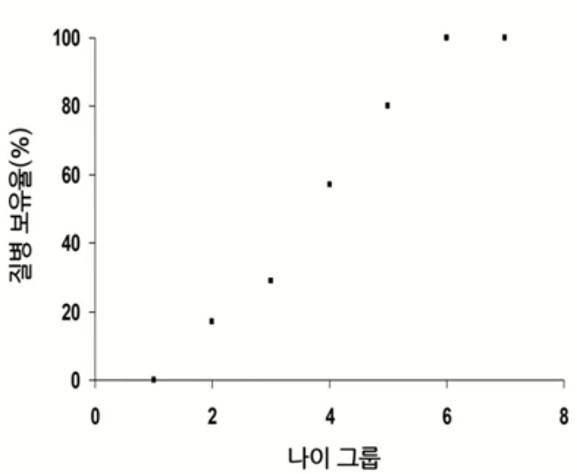
실행방법 09

서론

Logistic Regression

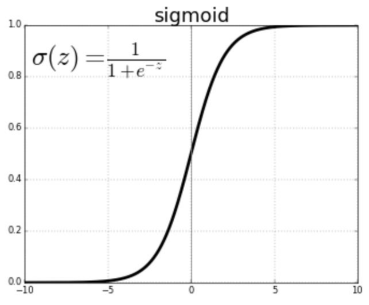
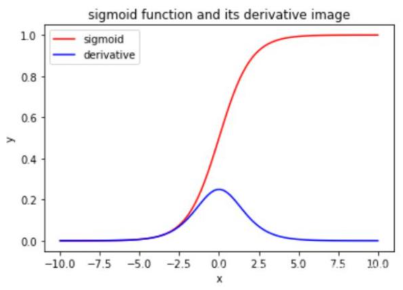
Logistic Regression이란 무엇일까? 값을 예측하는 regression문제에서 연속적인 label이 존재하고 이들이 linear한 관계를 가질 경우 linear regression은 훌륭한 예측 모델이자 분석 방법이 되어준다. 그러나 label이 연속적이지 않고 어떠한 class에 속한다면 linear regression은 적용하기 어려워진다. 즉, classification을 위해서는 다른 방식이 필요하다.

출처 : 강의자료

테이블이(가) 표시된 사진

자동 생성된 설명예를 들어 가장 간단한 binary classification을 생각해보자. Y가 0 또는 1이라는 이산적인 값들로 정해져 있기 때문에, 위의 그림처럼 linear regression의 선형 모델을 직접 적용시킬 수 없다. 이때, 각 데이터의 feature에 의해 어떤 클래스가 결정되는 확률을 살펴보면 아래와 같다. 나이라는 feature를 기준으로 Y=1에 속할 확률을 그려보면 우측과 같은 그래프를 만드는 것을 알 수 있다.

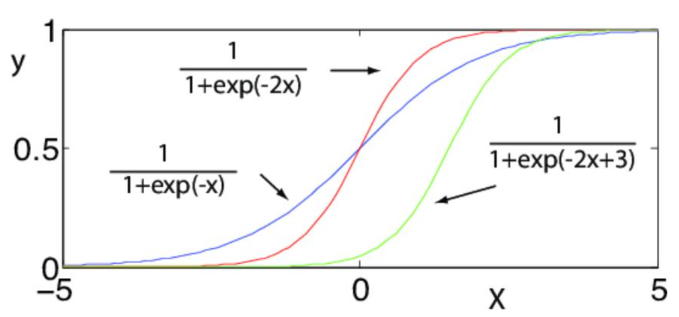
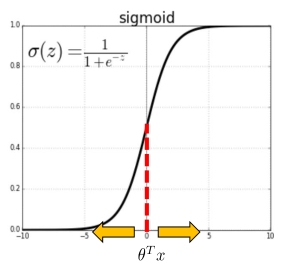
출처: 유튜브[핵심 머신러닝] 로지스틱회귀모델1 (로지스틱함수, 승산) https://youtu.be/l\_8XEj2\_9rk

이러한 형태를 띠는 함수 중, sigmoid function은 그 first derivative가 bell shaped이고, first derivative가 sigmoid 자기 자신을 포함하는 식으로 나타나기 때문에 많이 사용된다.

출처: 강의자료

정리하면, sigmoid 함수는 binary classification에서 feature에 따라 label에 속할 확률을 설명할 수 있는 모델이고, 따라서 이 모델을 그 확률에 맞게 조정하면 label이 구분되는 decision boundary를 정해 예측할 수 있다. 이때, sigmoid는

꼴로 표현되는 Logistic function의 특이 케이스이기 때문에 우리는 이러한 방식을 Logistic Regression이라고 부른다. 실제 logistic regression에서는 sigmoid(x)의 x에 를 사용하여

라는 모델을 만들고, 이를 통해 linear decision boundary를 찾는다.

Cost Function

Linear regression에서 수행했던 과정과 마찬가지로, 우리는 decision boundary를 찾기 위해서, cost function (에 대해 를 통해 를 학습해야 한다. 이를 위해 linear regression에서 사용한 MSE는 logistic regression에 적합하지 않다. Logistic regression에서는 Cross Entropy를 cost function으로 사용한다.

Cross Entropy는

이다.

따라서,

로 나타낼 수 있다. 이 함수 에 대해 gradient descent를 수행하면 를 만족하는 방향으로 를 학습할 수 있다.

이므로 최종적으로 우리가 향할 방향은 아래와 같다.

본론

행렬 곱으로의 전환

Cost function과 gradient descent를 여러 행의 데이터에 대해 구하는 과정을 iteration을 취하거나, 또는 sum과 multiplication을 지원하는 함수를 사용할 수도 있다. 그러나, 두 함수에서 모두 multiplication의 합 형태가 나타나는데, sum of multiplication은 곧 행렬 곱과의 관계가 있다는 것에서 착안하면 더 수식을 간단하게 만들고, 구현 상의 편의도 챙길 수 있다.

먼저 를 살펴보자. 의 내부를 시그마에 대해 분리하면,

의 꼴로 나타낼 수 있다. 이때 이를 A와 B의 행렬 곱이 C일 때, 임에서 착안하면, 라는 새로운 차원을 도입했을 때,

로 생각할 수 있을 것이다. 입력 데이터의 형태를 고려하여,

라면, 결과적으로

로 나타낼 수 있을 것이다. 물론 이러한 결과는 함수나 scalar의 broadcasting을 고려한 rough한 결과이다.

마찬가지의 방법으로, gradient descent의 역시

로 나타낼 수 있다.

코드 설명

코드 설명은 전체 부분을 다 설명하지 않고, 수식과 연관되는 부분 위주로 설명하도록 하겠다.

텍스트이(가) 표시된 사진

자동 생성된 설명먼저 와 를 정의하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명이어서 를 아래와 같이 정의하였다

수식과 색깔 박스로 비교하면 아래와 같다.



텍스트이(가) 표시된 사진

자동 생성된 설명Gradient descent의 theta update 부분은 아래와 같이 구현하였다.

이를 수식과 색깔 박스로 비교하면 아래와 같다텍스트이(가) 표시된 사진

자동 생성된 설명

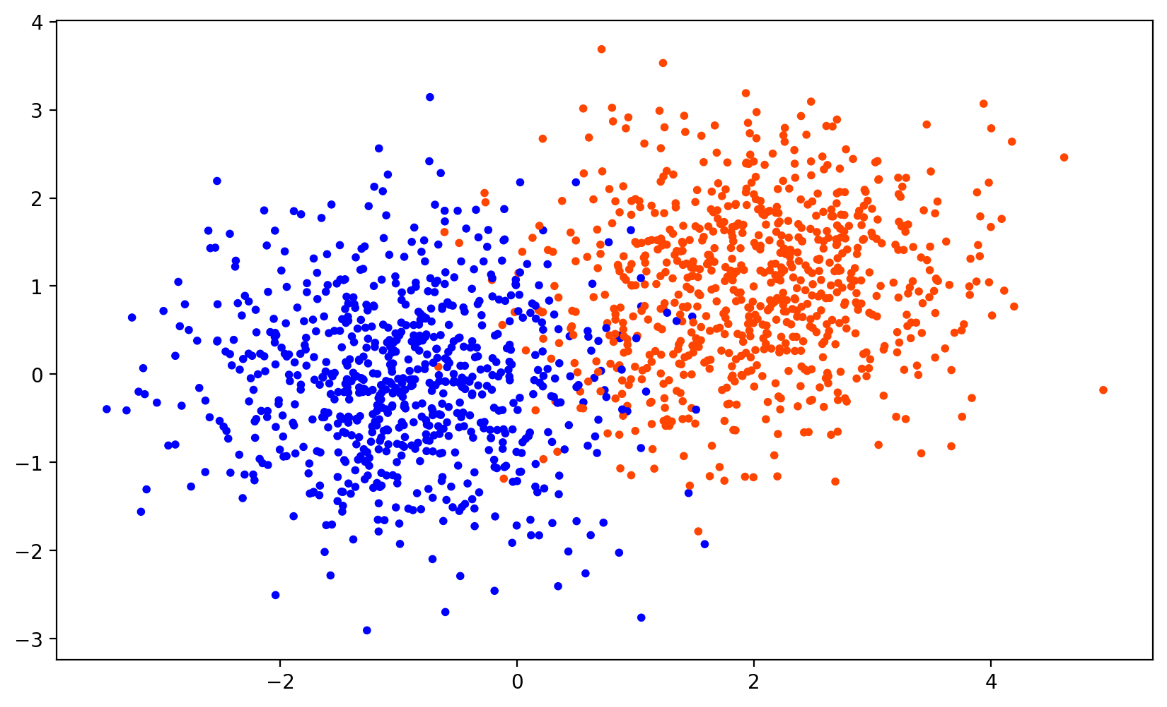
텍스트이(가) 표시된 사진

자동 생성된 설명이 외에 추가적으로 논의할 부분은 grid search이다. 의 다항식 꼴에서 최대 차수 depth, 그리고 batch size, epoch, alpha 등의 값을 list로 후보군을 만들고, 그 grid 안에서 최적의 하이퍼 파라미터를 찾도록 반복하여 개별 grid 마다 학습을 진행했다. 이 부분에서 cross validation을 구현하지 못한 점이 아쉽지만, grid search의 구현은 성공적이었다. 구현 결과는 아래와 같다

그러나 grid search의 구현이 성공적인 것과는 별개로, grid search를 진행했을 때, depth나 epoch, batch 등에 따라 minimum cost와 score의 차이가 거의 없었다.

따라서, 최종적으로 depth 1(= )에 대해서 반복을 수행했다.

결과 분석

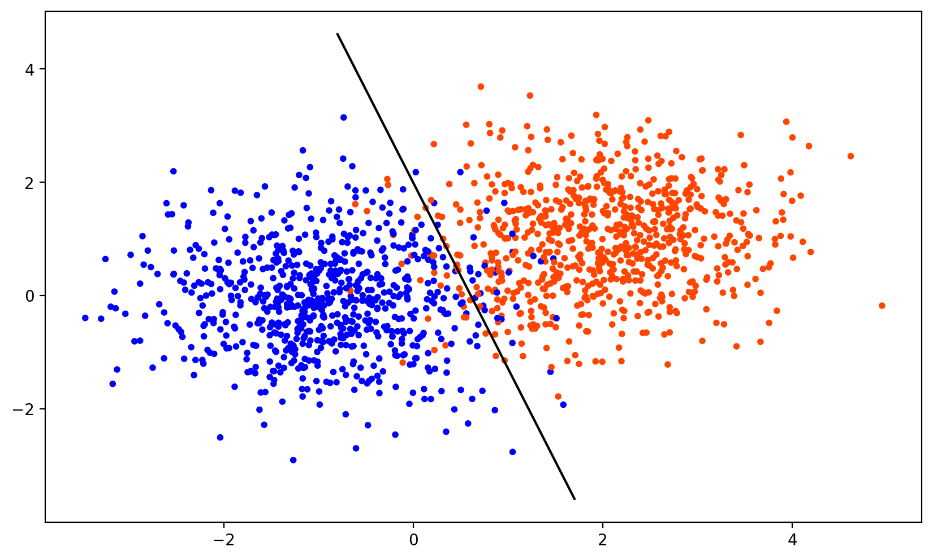
먼저 training dataset은 아래와 같이 구성되어 있다.

학습 과정에서 loss는 아래와 같이 감소하였다.

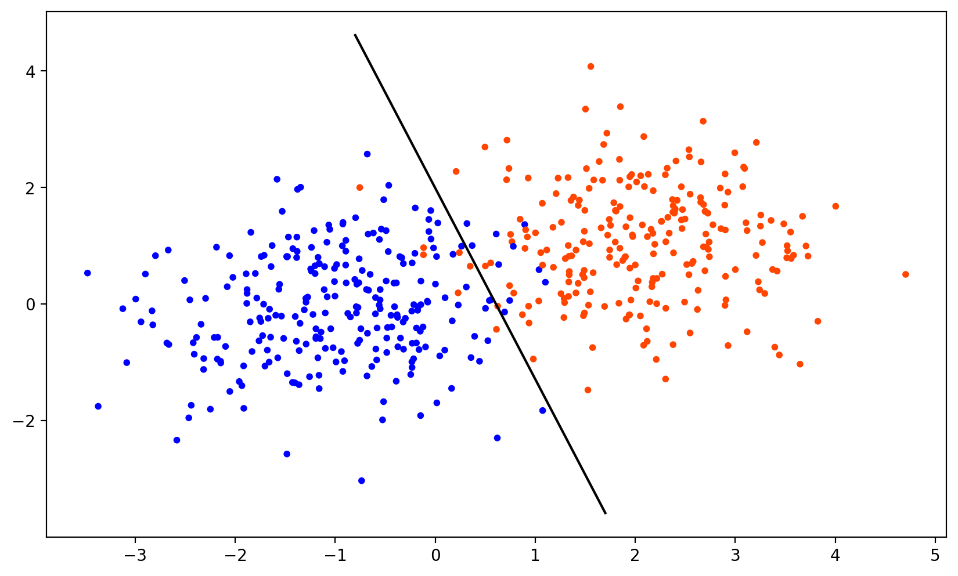
위의 과정을 통해 얻어낸 최종 theta는 아래와 같다.

optimal depth:1, optimal batch:0, optimal epoch:15000, optimal alpha:0.1

optimal theta:[[2.28917126], [-3.77578216], [-1.15149672]]

이 theta에 대해 training data set의 decision boundary는 아래와 같이 그려진다

텍스트이(가) 표시된 사진

자동 생성된 설명또한 test data에 대해 prediction을 수행한 결과는 아래의 캡쳐와 같다. 최종 결과는 마지막에 표로 정리하였다.

이러한 결과를 통해 알 수 있는 것은, 데이터셋의 label이 혼재된 영역 때문에 생기는 irreducible error로 인해 logistic regression이 충분히 학습 되었음에도 loss가 irreducible error에 수렴하여 더 이상 작아지지 않음을 알 수 있다.

결론

최종 결과

A. for

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Task1 | 2.28917126 | -3.77578216 | -1.15149672 |

B.

|  |  |  |
| --- | --- | --- |
|  | Cost | Accuracy |
| Task1 (train) | 0.10635371 | 0.958 |
| Task2 (test) | 0.08810934 | 0.966 |

실행 방법

HW2.ipynb를 jupyter notebook이나 colab에서 열고 하나씩 셀을 실행하면 됩니다. Numpy, matplotlib이 설치되어 있어야합니다.