

WEEK1-2

㉟ 자료 정보

작성자	김민섭
강의	Coursera 2주차

㉟ Binary Classification (이진 분류)

이진 분류는 말 그대로 두 개로 분류하는 것이다. 일반적으로 프로그램에서는 0 혹은 1 으로 분류한다.

'왜 이것을 배우는 가?'는 뉴런을 보면 이해 가능하다.

뉴런에서 '흥분의 전도 현상'에서 '역치'라는 현상이 발생한다. 역치란 간단히 말해 특정 수치 이상 값이 들어와야 흥분이 전도되는 상황을 의미한다.

다시 프로그래밍적 관점으로 보면, 노이즈를 제거하는 과정이라고 볼 수 있다. 이진 분류를 통해 의미가 없는 값이면, 0 혹은 0에 근접한 수를 사용한다.

㉟ Logistic Regression (로지스틱 회귀)

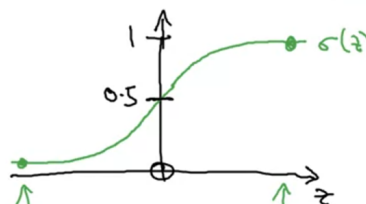
로지스틱 회귀 개념

Logistic Regression

Given x , want $\hat{y} = P(y=1|x)$
 $x \in \mathbb{R}^{n_x}$
 $0 \leq \hat{y} \leq 1$

Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$.

Output $\hat{y} = \sigma(\underbrace{w^T x + b}_z)$



$$x_0 = 1, \quad x \in \mathbb{R}^{n_x+1}$$
$$\hat{y} = \sigma(\theta^T x)$$
$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \begin{matrix} \} b \leftarrow \\ \} w \leftarrow \end{matrix}$$
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

If z large $\sigma(z) \approx \frac{1}{1+0} = 1$

If z large negative number

$$\sigma(z) = \frac{1}{1 + e^{-z}} \approx \frac{1}{1 + \text{Big num}} \approx 0$$

로지스틱 회귀 비용함수

- 비효율적인 손실함수

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

이러한 손실 함수를 사용할 수 있지만, 제공 때문에 최적화에서 문제가 발생한다.

- **효율적인(최적X) 손실함수**

$$L(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

다음과 같은 손실 함수는 **log** 를 사용하기 때문에 제공보다는 효율적이다.

수식 앞에 -(음의 기호)를 붙이는 이유는 \hat{y} 는 **sigmoid** 함수로 범위는 (1, -1) 이다.

즉, (1, 0)에서 **log** 값은 음수가 나오기 때문에 -(음의 기호)를 붙인다.

예제

y = 1 일 경우:

$$L(\hat{y}, 1) = -(1 \times \log \hat{y} + (1 - 1) \log(1 - \hat{y})) = -\log \hat{y}$$

다음과 같이 나옴. 손실함수 값이 작다는 것은 모델 정확도가 높아진다는 의미이다.

따라서 위 수식의 경우, **log \hat{y}** 값이 커져야 한다.

여기서 \hat{y} 는 **sigmoid** 함수로 범위가 (1, -1) 이다. 따라서 \hat{y} 값은 커져야 한다.

이것은 모델 학습 시, 고려해야 할 부분 중 하나이다.

y = 0 일 경우:

$$L(\hat{y}, 0) = -(0 \times \log \hat{y} + (1 - 0) \log(1 - \hat{y})) = -\log(1 - \hat{y})$$

같은 이유로 이 경우에는 \hat{y} 가 작아야 손실함수 값이 작아진다.

㉞ Gradient Descent (경사 하강법)

$$\text{Recap : } \hat{y} = \sigma(w^T x + b), \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

정확한 설명은 아니지만 미분 개념과 유사함. 때문에 '정확한' 값이 아닌 '근사값'을 찾는 알고리즘이다.

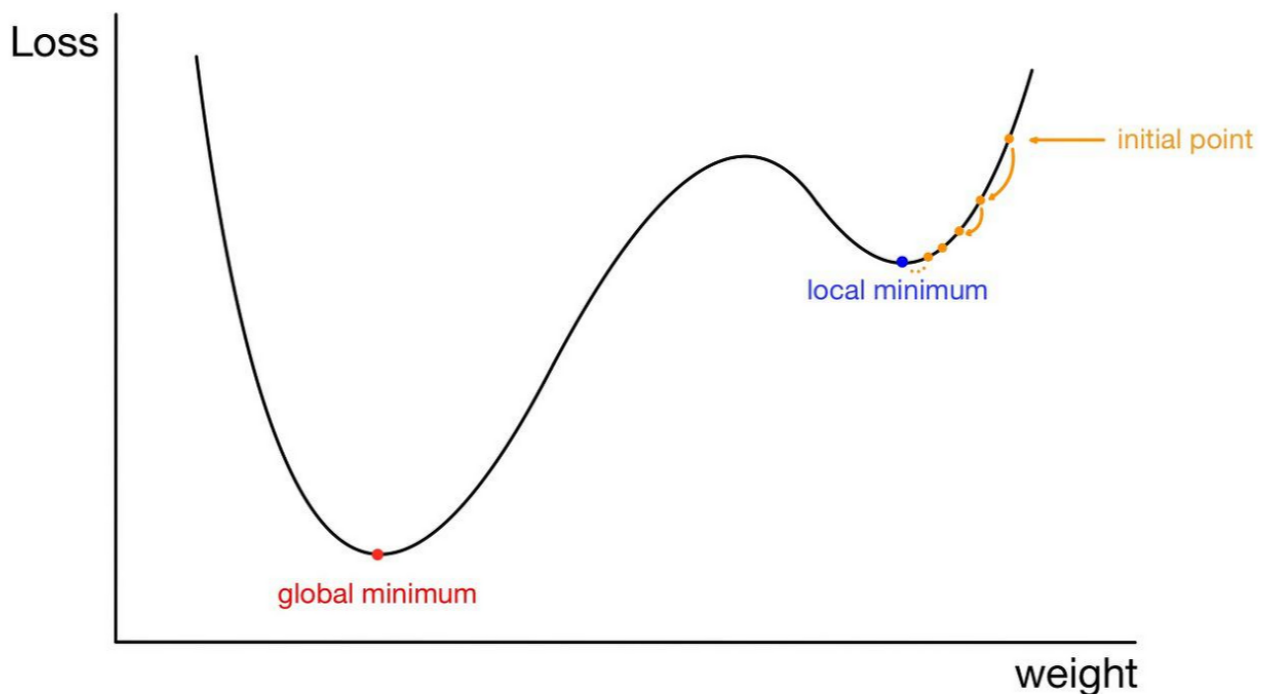
위 $J(w, b)$ 를 반복하는 형태이다.

$$\begin{aligned} \alpha &= \text{learning rate} \\ \text{Repeat } \{ \\ &\quad w := w - \alpha \frac{\partial J(w)}{\partial w} \\ &\} \end{aligned}$$

alpha 는 **learning rate**. 한국어로 학습도 일반적으로 **ML** 에서 **lr** 로 축약해 사용한다.

말 그대로 한 번에 '학습하는 정도'를 의미함. 헷갈리는 개념인데, 학습도가 높다고 정확도가

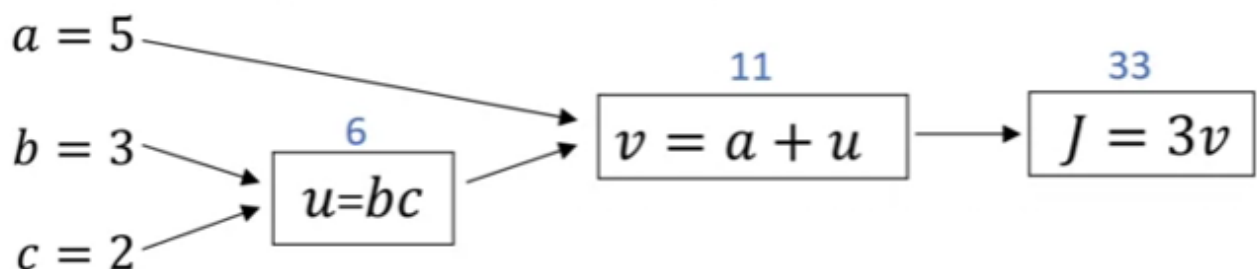
높아지는 것이 아니다.
이를 이해하기 위해 다음 그림을 보며 이해해 보자.



위 예제에서 `learning rate`가 작다면 위 그림과 같이 `local minimum`만 찾게 될 것이다.
반대로 `learning rate`가 크다면 위 그림에서 `global minimum`에 근접할 수 있겠지만, 최소 값을 못찾을 수 있다.

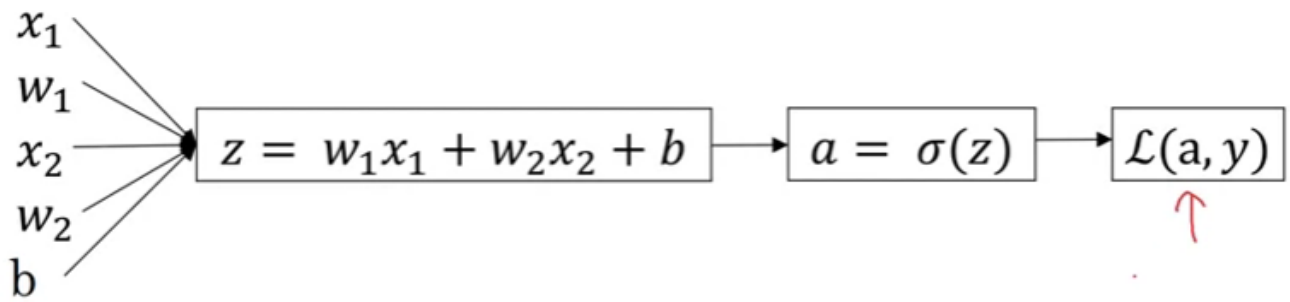
때문에 실제로 `learning rate`는 너무 크지도 작지도 않은 중간 지점을 찾아야 한다. (결국.. 노가다.)

Computing derivatives



위에서 배운 내용을 종합적으로 사용하는 모습이다.

이 모델의 기초 개념은 생물의 뉴런에서 따왔다고 알고 있다. 실제로는 뉴런을 모방한 것도 아닌 새로운 개념이라고 창시자가 말했지만요. (이거 때문에 철학적으로 인공지능을 사람으로 보아야 하는 가?라는 질문에서 논쟁이 존재하는 걸로 앎.)



Vectorization (벡터화)

행렬 && Numpy 최고

Vectorization 쓰는 이유

행렬에 있는 값을 수정할 때, 2중 for문을 사용하는 방법으로 해결할 수 있다. 하지만, 파이썬 `numpy`에는 내장된 함수가 존재하고 이게 성능이 좋기 때문에 권장한다.