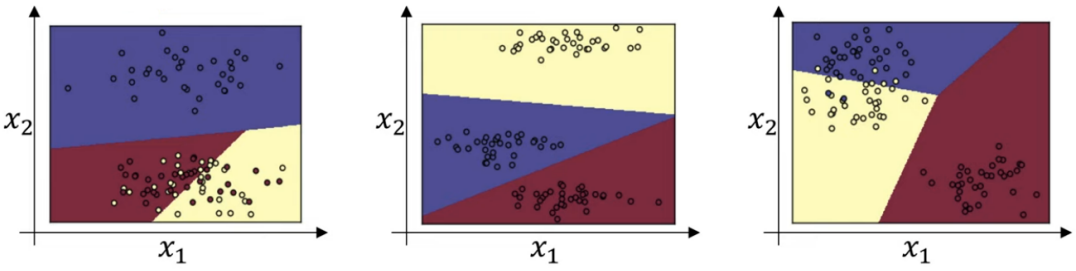
[Week12]_이원주

다중 클래스 분류

소프트맥스 회귀 (Softmax Regression)

개념

- *Aha*
  - 로지스틱 회귀(이진 분류)를 다중 클래스 분류로 일반화
  - 클래스가 많아져도 클래스 사이의 경계는 항상 선형임.



- **출력층**
  - 활성화 함수로 **softmax 함수** 사용
    - **softmax 층**이라고도 함.
  - 각 노드의 a = 각 클래스에 속할 확률
    - 노드 개수 = 클래스 개수
    - 모든 a의 합 = 1

- **Softmax 함수**
  - 활성화 함수의 일종

ReLU, sigmoid 등	실수 입력 → 실수 출력

Softmax 함수	벡터 입력 → 벡터 출력

- 수식

$$a = \frac{e^z}{\sum e^z}$$

그래서 softmax 층에서는 모든 a의 합 = 1이 됨.

- ↔ Hardmax 함수

Softmax 함수	결과값 = **각 클래스**에 속할 **확률**	[0.3 , 0.5 , 0.2]
Hardmax 함수	가장 가능성 높은 클래스 **하나만 정함**	[1 , 0 , 0]

적용

↓데이터 하나만 고려했을 때

손실함수

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

- 로지스틱 회귀와 다르게 출력층에서 값을 여러 개 출력
- 출력층의 모든 노드 → 비용 함수

↓데이터 set을 고려했을 때

비용함수

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

- 모든 데이터 → L의 평균
- 이제 이거 가지고 경사하강법 쓰면 됨

$$\hat{y} = \begin{pmatrix} 0.3 \\ 0.5 \\ 0.2 \end{pmatrix}$$

벡터화

$$\hat{Y} = \begin{pmatrix} 0.3 & \cdots & 0.6 \\ 0.5 & \cdots & 0.1 \\ 0.2 & \cdots & 0.3 \end{pmatrix}$$

$|\quad \cdots \quad|$ *이렇게 쌓음*

역전파

- 출력층에서 $dz^{[L]} = \hat{y} - y$

프로그래밍 프레임워크 소개

지역 최적값 문제

- 용어
  - **국소 최적값 → local optima**
    - 전체 말고 어느 local에서의 최적값
  - **안장점 → saddle point**
    - 어느쪽에서 보면 ∪ 모양, 다른 쪽에서 보면 ∩ 모양인 점
    - 말에 얹는 안장 같아서 붙은 이름
  - **안정지대 → plateaus**

        - 안장점으로 향하는 구간
        - 이 지역에서는 미분값이 아주 오랫동안 0에 가깝게 유지됨.

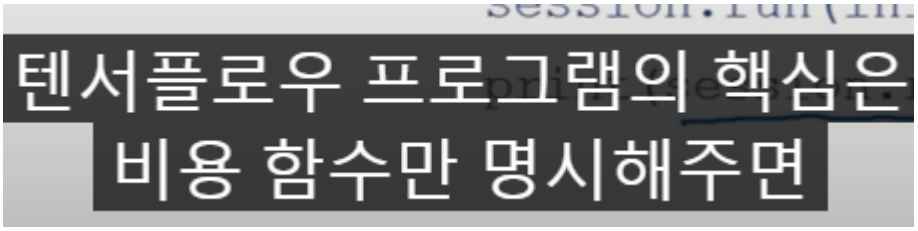
- 문제
    - 충분히 큰 Network에서 경사가 0인 경우는 대부분 → 지역 최적값 (X) / 안장점 (O)
    - 안정지대에서는 경사가 거의 0이므로 학습속도가 느려짐.
    - 방향 변환이 없다면 안정지대에서 벗어나기 어려움.
- 해결방법
    - Adam, RMSprop 등 알고리즘이 해결해줌.

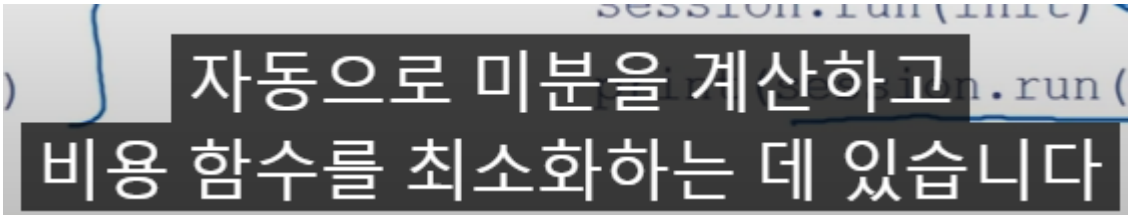
Tensorflow

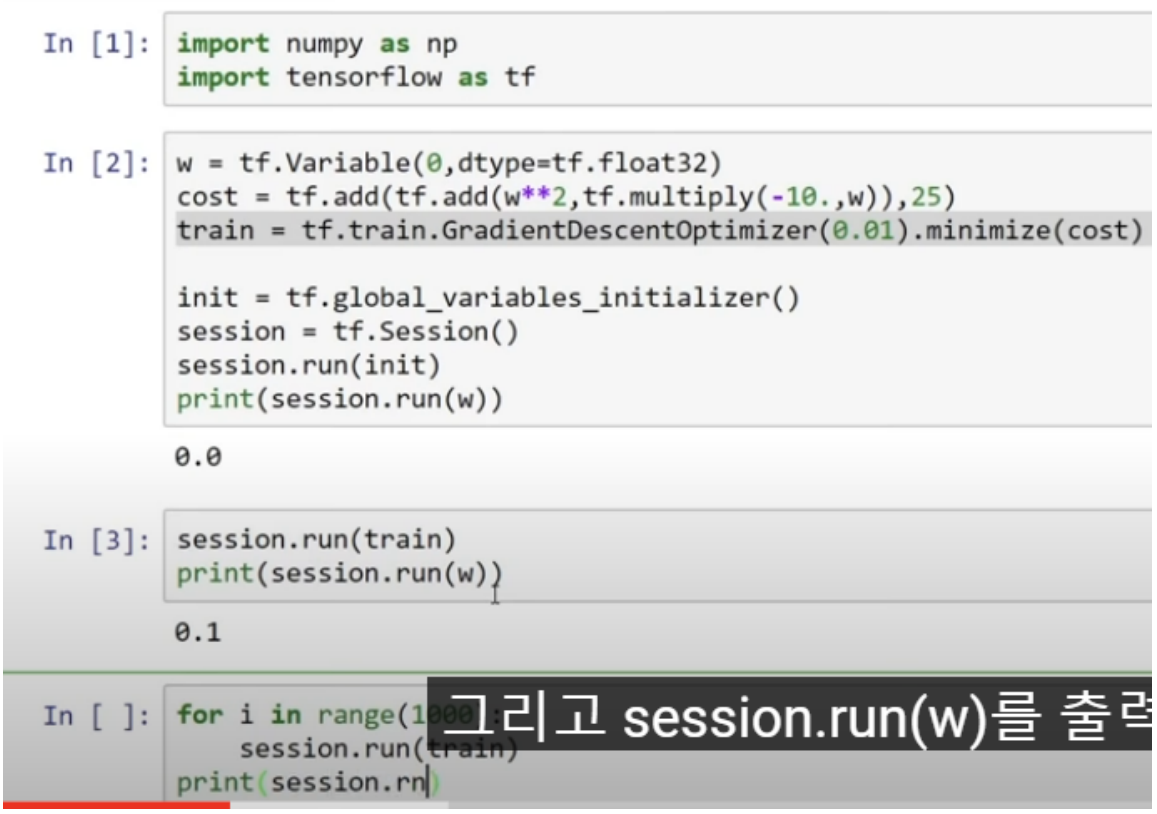
- 개념
    - 딥러닝 프레임워크 중 하나

```
import tensorflow as tf
```

- 장점
    -





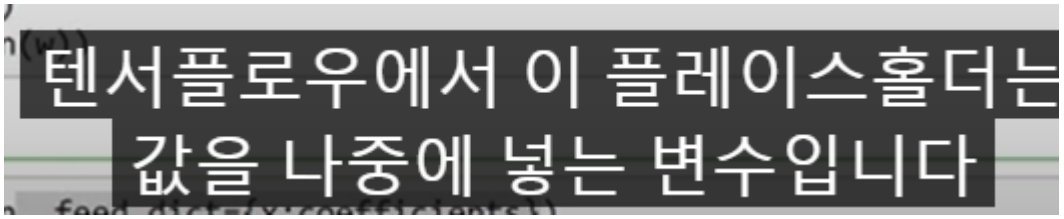
- 


```
cost = w**2 - 10*w + 25
```

- 데이터를 어떻게 사용하느냐

```
x = tf.placeholder(tf.float32, [3,1])
```

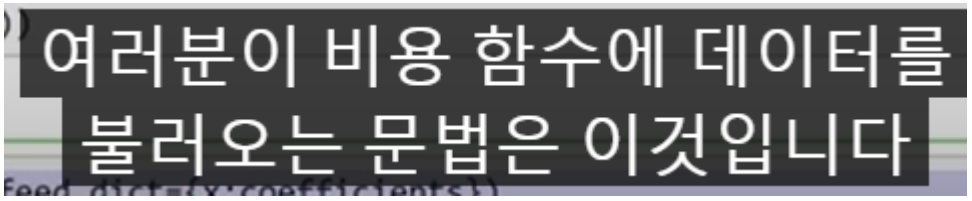
```
cost = x[0][0]*w**2 + x[1][0]*w + x[2][0]
```



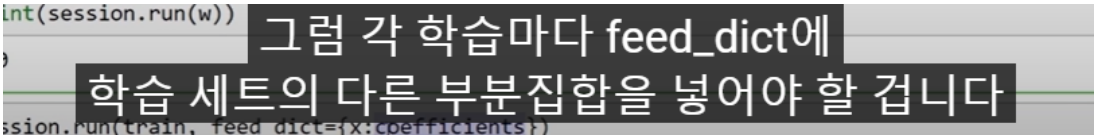
- co~ → x 에 넣는 방법

```
session.run(train, feed_dict={x:coefficients})
```

  - 학습 중에 x가 뭔지 알려줌



- 미니배치 쓰면



- 관용적

```
session = tf.Session()
session.run(init)
print(session.run(w))
```

```
with tf.Session() as session:
    session.run(init)
    print(session.run(w))
```