[Week12]_문가을

딥러닝 2단계 : 심층 신경망 성능 향상시키기

7. 다중 클래스 분류

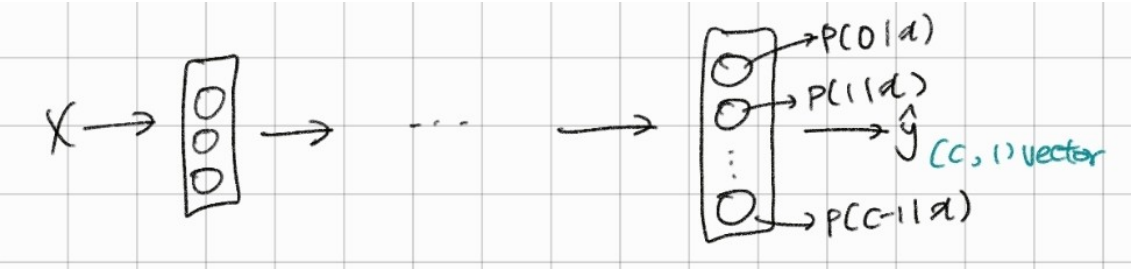
softmax Regression

logistic regression을 일반화한 softmax regression

C = # of classes

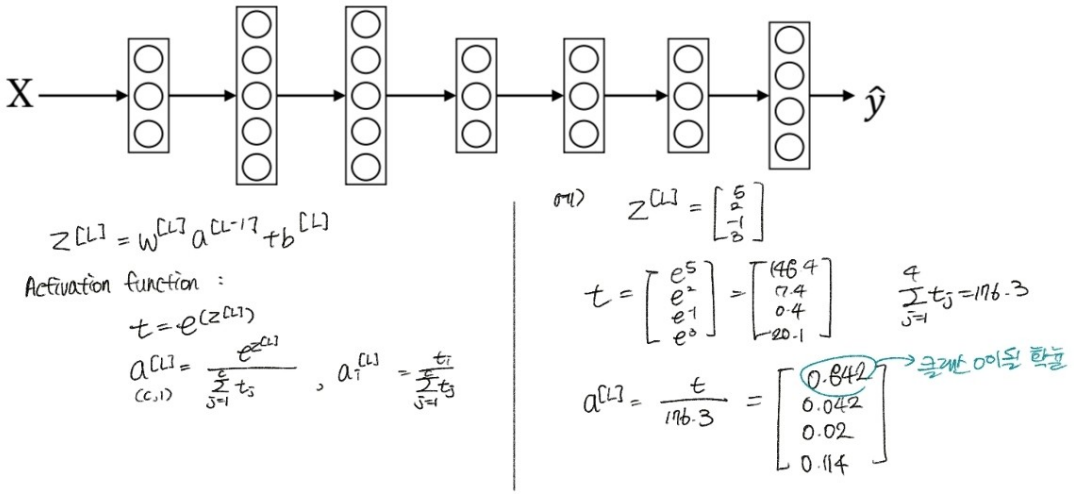
→ 0에서 c-1까지 클래스 부여

출력층 L의 유닛 개수 : n^[L] = C



y^의 각 값들의 합은 1이되어야 함.

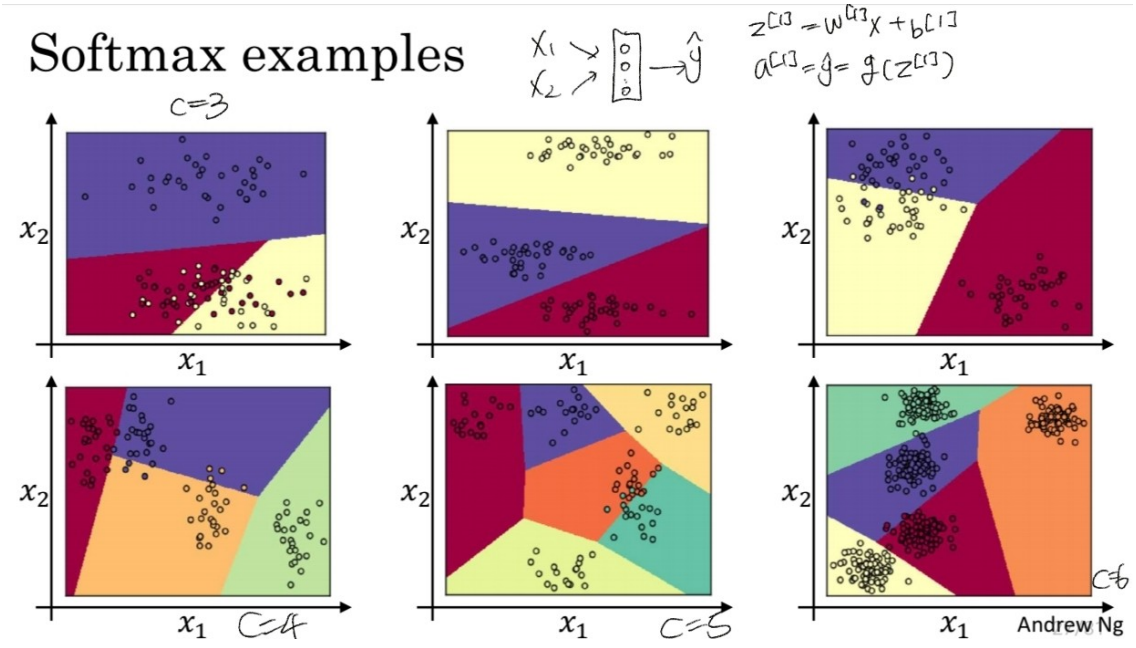
<softmax layer>



softmax activation function에서 특이한 점은 입력값과 출력값이 모두 벡터임.

<softmax examples>

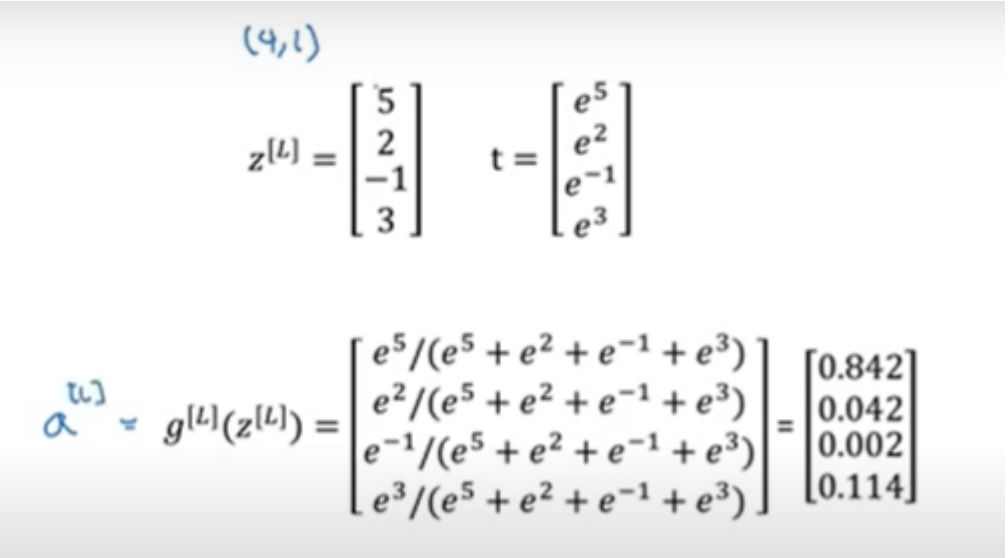
1 layer



두 클래스 사이의 경계가 선형임.

은닉 유닛이 여러 개인, 더 깊은 신경망에서는 여러 클래스를 분류하기 위해 더 복잡하고 비선형의 경계도 볼 수 있음.

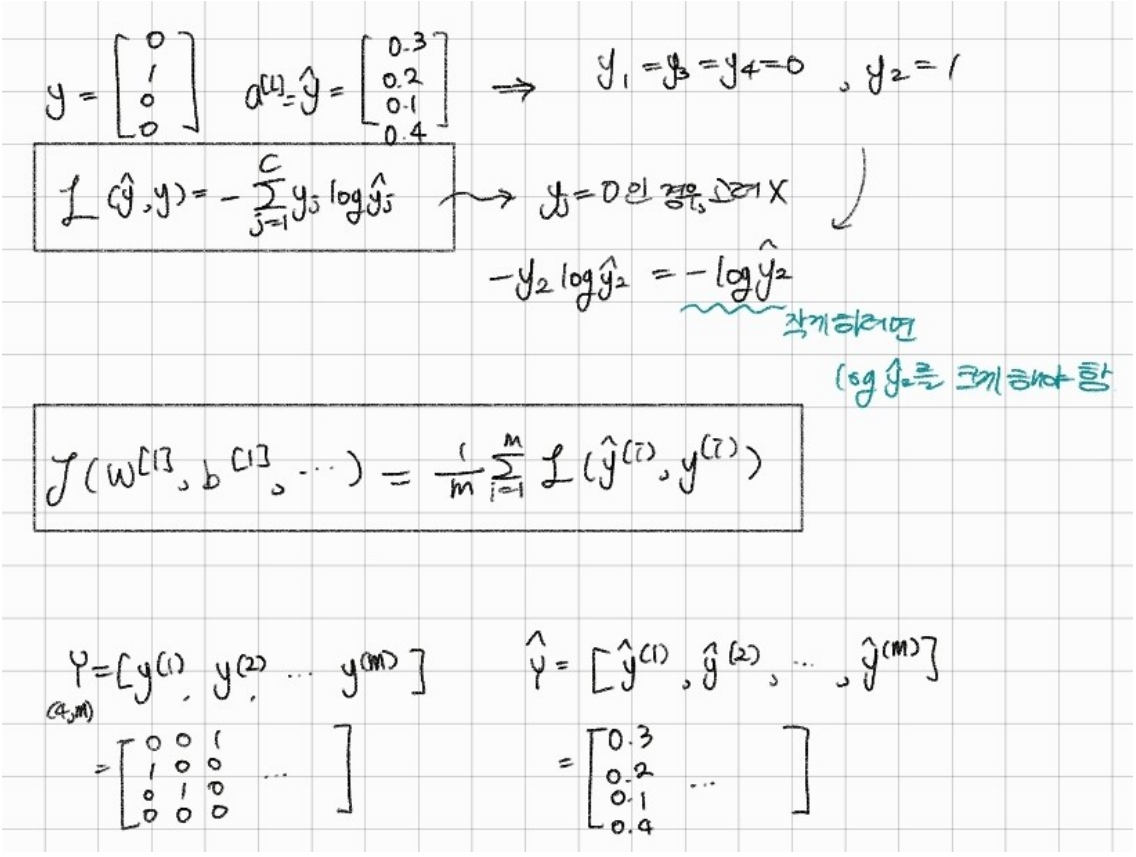
Softmax 분류기 훈련시키기



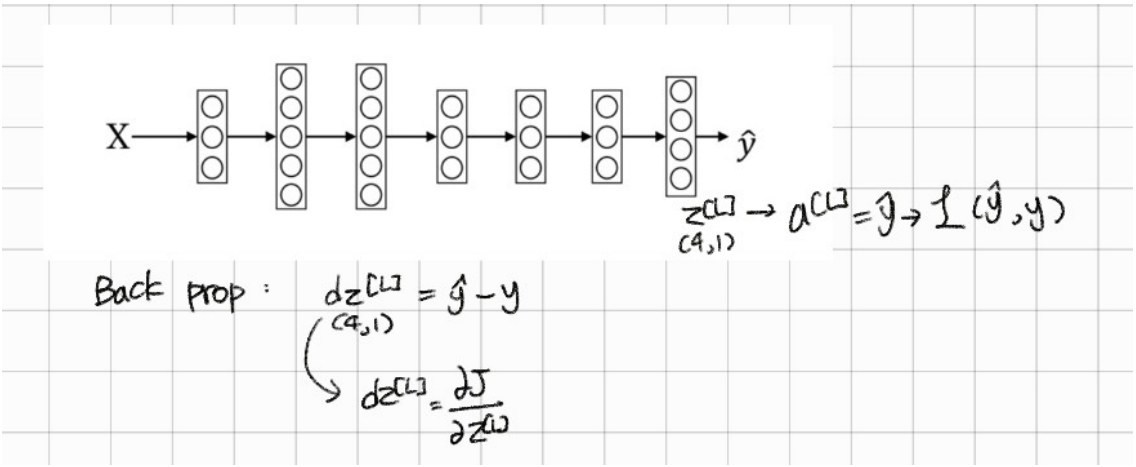
hard max → [1, 0, 0, 0]’

: z 값을 보고 가장 큰 값이 있는 곳에 1을 나머지에는 0을 갖는 벡터로 대응 시킴.

<Loss function(손실 함수)>

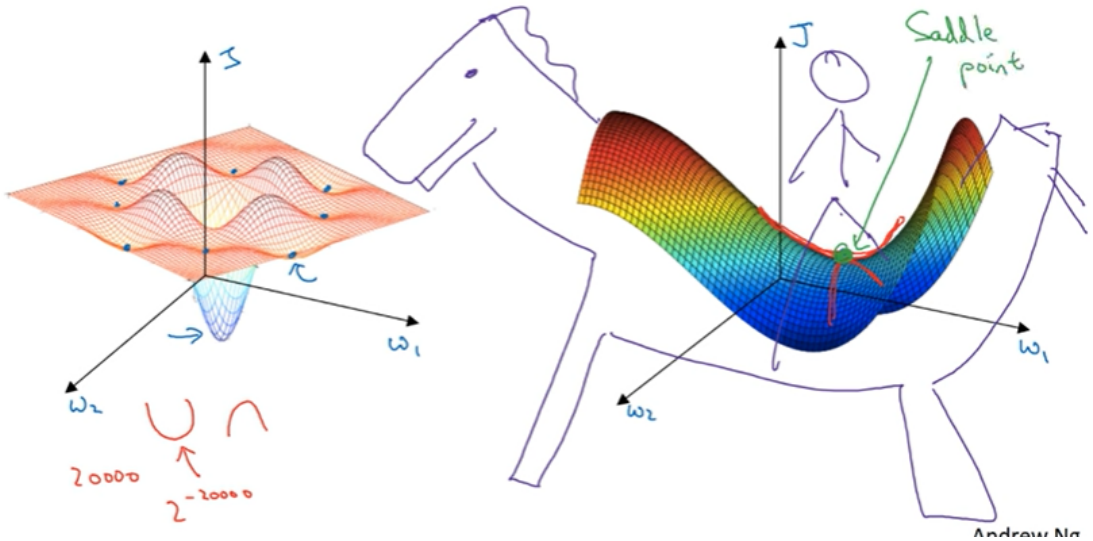


<Gradient descent with softmax>

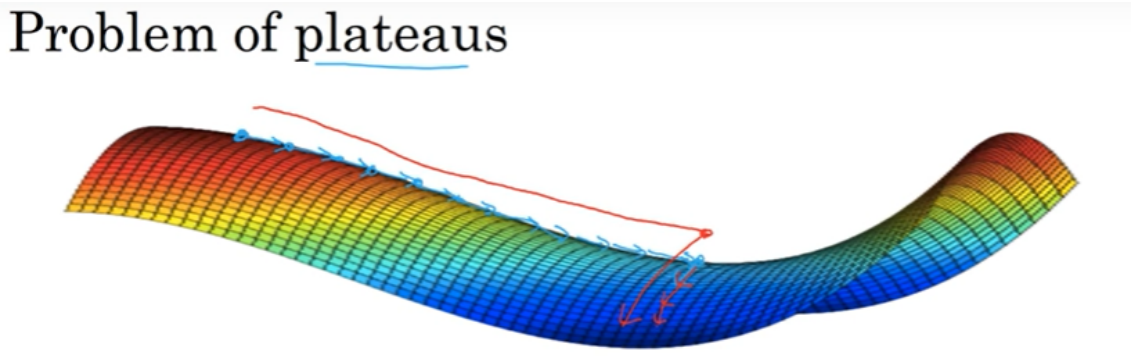


8. 프로그래밍 프레임워크 소개

지역 최적값 문제



고차원 비용함수에서 경사가 0인 경우는 대부분 지역 최적값이 아니라 대개 안장점임.

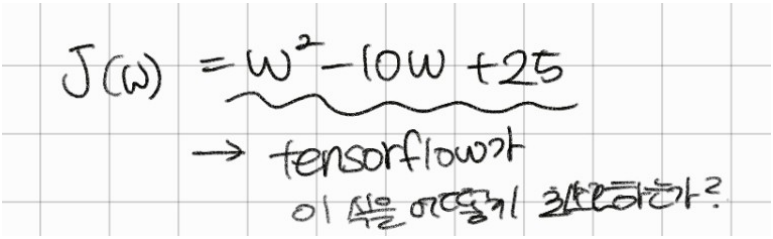


- 안장점으로 향하는 구간인 안정지대는 미분값이 아주 오랫동안 0에 가깝게 유지되는 지역 .
- 대개 충분히 큰 Network 학습시 지역 최적값에 갇이는 일은 거의 없다.
- 안정지대의 문제점은 경사가 거의 0에 가깝기 때문에 학습속도가 느려짐. 또한, 다른 쪽으로 방향변환이 없다면 안정지대에서 벗어나기 어려움. 이는 Adam, RMSprop 등 알고리즘이 해결해줌.

Tensorflow

딥러닝 프레임워크 중 하나.

<Motivating problem>



tensorflow를 이용하여 매개 변수와 비용함수를 정의하고, 비용함수를 최소화하는 경사하강법을 이용하는 훈련 객체 만듦.

```
w = tf.Variable(0,dtype=tf.float32)
cost = tf.add(tf.add(w**2,tf.multiply(-10.,w)),25)
train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)

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

전방향 경사만 잘 구현해도 tensorflow가 자동으로 역방향 전파를 잘 계산함.

경사 하강법 1000번

```
for i in range(1000) :
    session.run(train)
print(session.run(w))
```

5에 가깝게 나옴.