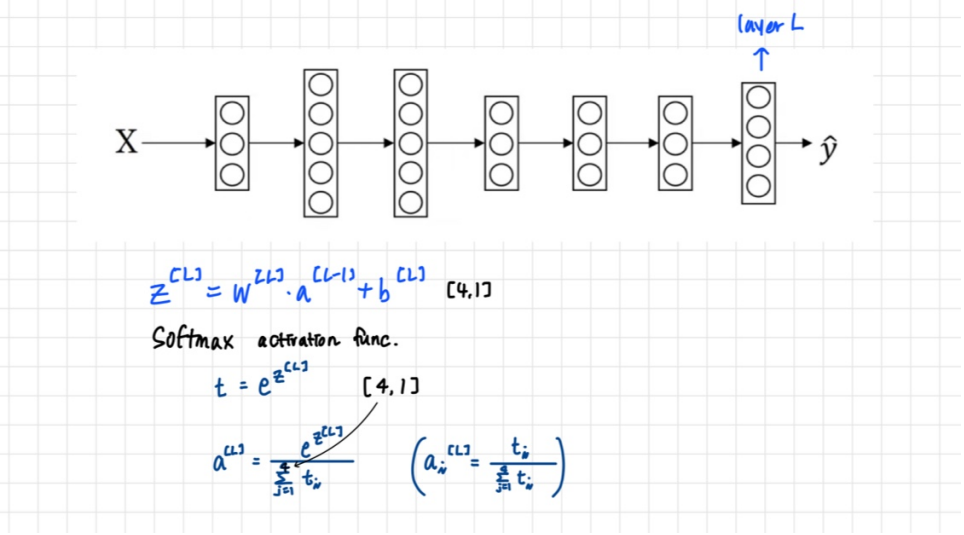
Softmax Regression

여러 클래스 중 하나를 인식할 때 예측에 사용한다.

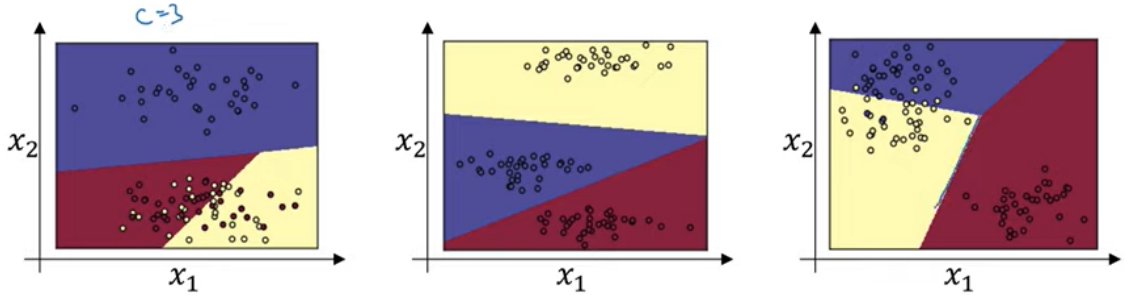
강아지(class 1), 고양이(class 2), 병아리(class 4), 그외(class 0) 로 분류한다고 해보자.

출력층의 유닛의 개수는 4개가 될 것이고 출력값으로 4개의 확률값이 주어지므로 y^의 차원은 [4,1]이다.

우리는 softmax를 통해 출력값을 확률값으로 만든다.



t는 임시변수. a^[L]은 합이 1이 되도록 정규화 한다. 이 활성화 함수에서 특이한 점은 활성화 함수를 적용하였을 때 차원이 바뀌지 않는다는 것이다. 이전의 활성함수에서는 하나의 실수값을 받았으나, softmax는 입력값과 출력값 모두 같은 차원의 벡터이다.



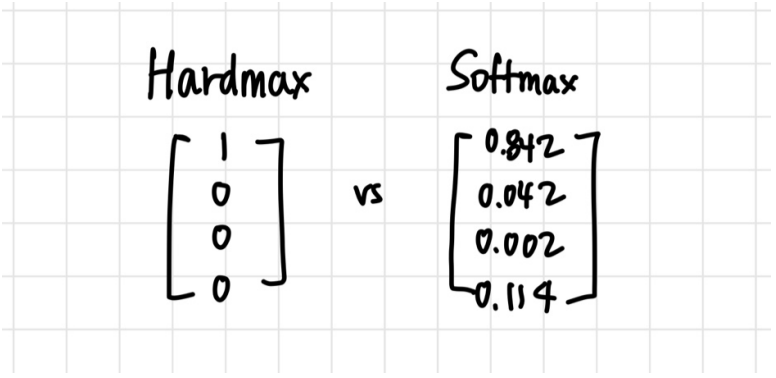
은닉층이 없는 경우 softmax를 사용하면 선형 결정 경계가 나타남을 알 수 있다. 만약 은닉층이 많은 깊은 신경망을 다룬다면, 선형이 아닌 비선형의 더 복잡한 경계도 볼 수 있을 것이다.

$$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \qquad t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix}$$

$$g^{[L]}(z^{[L]}) = \begin{bmatrix} e^5/(e^5+e^2+e^{-1}+e^3) \\ e^2/(e^5+e^2+e^{-1}+e^3) \\ e^{-1}/(e^5+e^2+e^{-1}+e^3) \\ e^3/(e^5+e^2+e^{-1}+e^3) \end{bmatrix} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$

z^[L]의 가장 큰 원소가 5였고, a^[L]에서도 보면 원소가 5였던 부분의 확률값이 0.842로 제일 큼을 알수 있다.

+Hardmax는 가장 큰값이 있는 곳에 1을, 나머지에는 0을 갖는 벡터로 대응시킨다.
Softmax가 더 부드럽게 z값들을 이런 확률들로 대응시킨다.

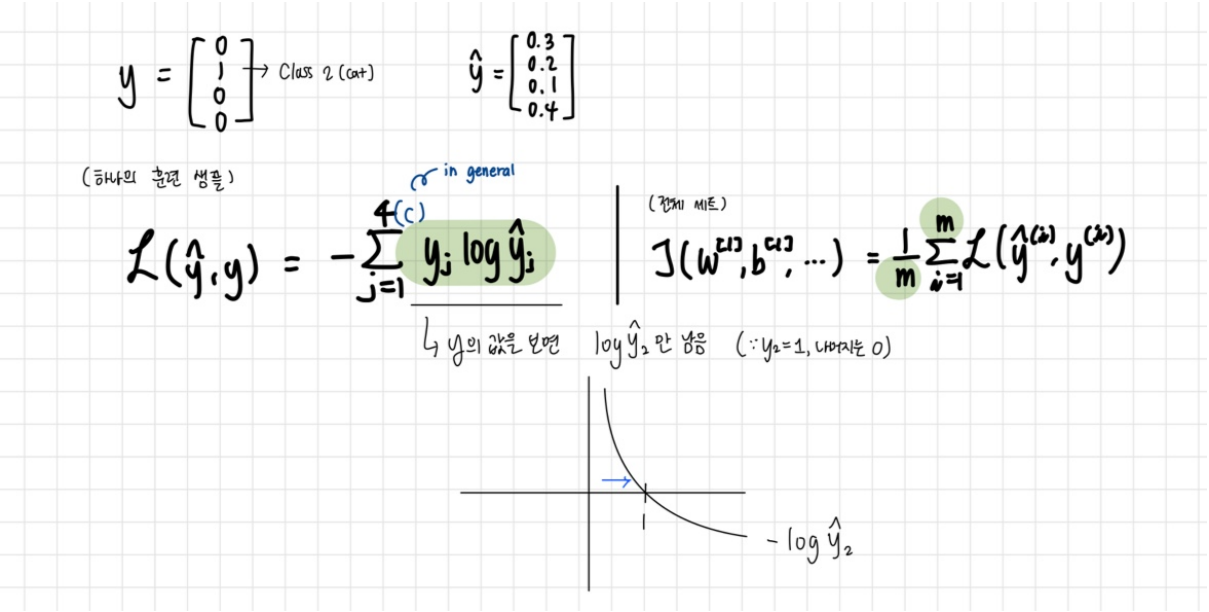


Softmax는 두 클래스만 다루는 logistic regression을 C개의 클래스로 일반화 한 것이다.

Training a softmax classifier

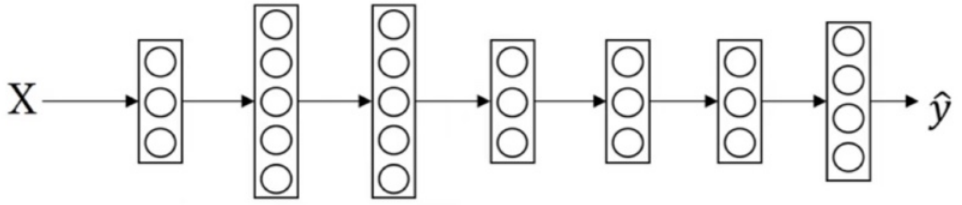
softmax를 사용하여 모델을 학습시키는 방법을 배워보자.

먼저 신경망을 학습하기 위해 사용했던 손실함수에 대해 정의해보자.



학습과정에서 우리는 loss function의 값을 줄이는게 목표이고, 이는 결국 -logy^_2을 작게 만드는 것이다. 이는 y^_2의 값을 가능한 크게 만들어야 한다는 것이다. 즉, y^_2의 확률값이 커야 된다. 다른 종류의 클래스의 확률값에 상관없이 실제값의 클래스에 해당하는 확률을 높여야 되는 것.

Gradient descent with softmax

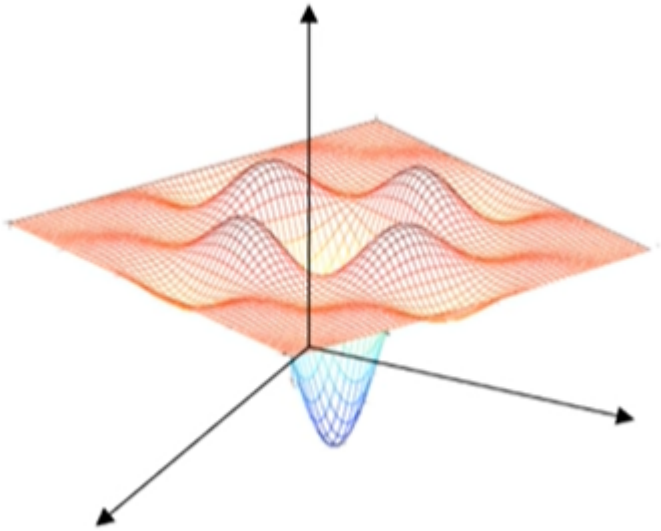


$z^{[1]} \rightarrow a^{[1]} \rightarrow \hat{y} \rightarrow \text{loss } \ell(\hat{y}, y)$

● 역전파 : $\frac{dz^{[1]}}{\frac{\partial L}{\partial z^{[1]}}} = \hat{y} - y$

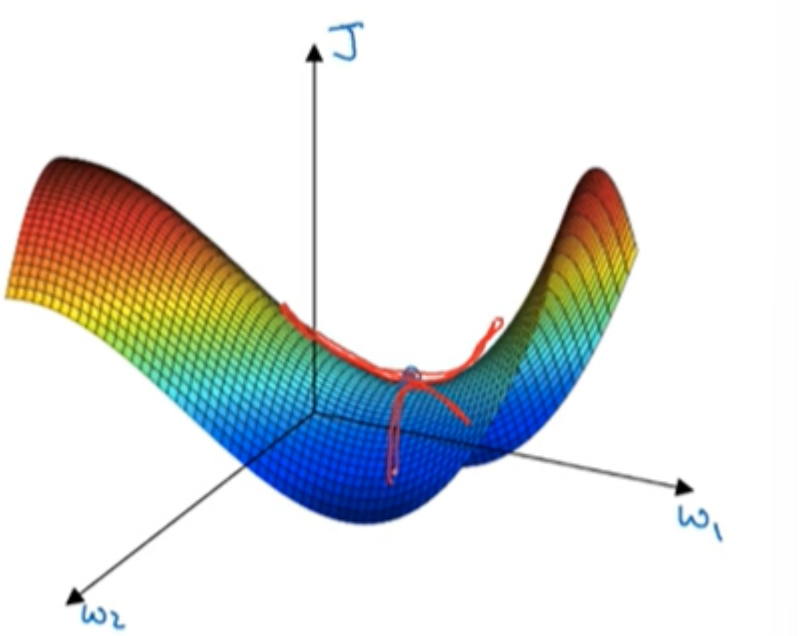
Local Optima problem

아래의 그래프를 보자. 아래의 그래프의 면적의 높이가 비용함수이다. 여기에는 local optima값이 많다. 이는 알고리즘이 global optima에 도달하기 전에 local값에 갇혀버리기 쉽다는 것을 의미한다.



비용함수의 경사가 0인 점은 대부분 지역최적값이 아니라 안장점이다.

안장점은 아래의 그래프를 참고하자. 안장점은 기울기가 0이고, 어떤 점에서 어느 한 축에서는 극대에 이르면서 다른 축에서는 극소에 이르는 점을 의미한다.

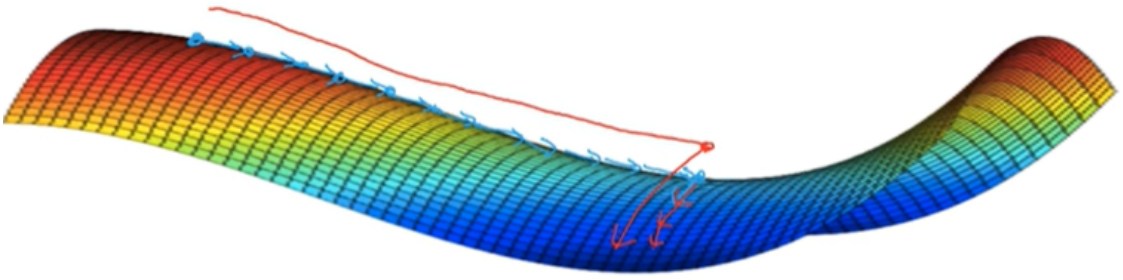


특히 고차원의 함수의 경우 20000개의 차원의 함수라면, 20000개의 차원에서 모두 아래로 볼록해야 한다는 것인데, 그럴 확률은 매우 낮다.

충분히 큰 신경망을 학습시킨다면 오히려 local optima에 갇힐 문제는 사라지지만, 안정지대가 학습을 매우 느리게 만든다는 문제가 생긴다.

Problem of plateaus

안정지대(plateaus)란 미분값이 아주 오랫동안 0에 가깝게 유지되는 지역을 말한다. 안정지대 때문에 학습속도는 매우 느려진다.



이는 momentum, RMSprop, Adam등의 알고리즘으로 개선할 수 있다.

DL framework : Tensorflow

비용함수 J= w^2 -10w+25 를 최소화하는 값을 찾는다고 해보자.

w를 찾고 싶으니 변수로 선언하고, 비용함수를 정의하였다. Tensorflow 에서는 미분을 계산할 줄 알기 때문에 정방향 전파만 구현하면 역전파는 알아서 계산해준다.

X는 어떻게 정의할까? 플레이스 홀더를 통해 정의한다. 플레이스홀더는 값을 나중에 넣는 변수로, 학습데이터를 불러올 수 있다. 미니 배치를 사용한다면, 학습마다 다른 coefficient를 넣어주면된다.

경사하강법을 1000번 실행하면 w가 최적값 5와 가까운 4.9999를 가짐을 알 수 있다.