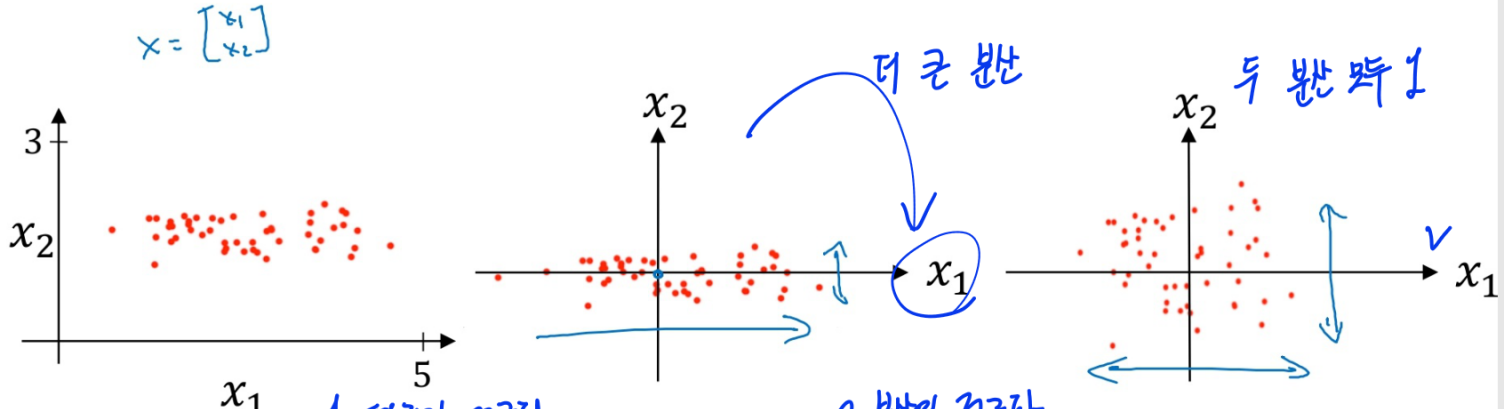
deeplearning.ai

Setting up your optimization problem

Normalizing inputs

입력의 정규화.

Normalizing training sets



1. 평균의 정규화

Subtract mean:

$\mu = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$

$x := x - \mu$

2. 분산의 정규화

Normalize variance

$\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} x^{(i)} **2$ ← element-wise

$x /= \sigma^2$

Use same μ σ^2 to normalize test set.

Andrew Ng

Why normalize inputs?

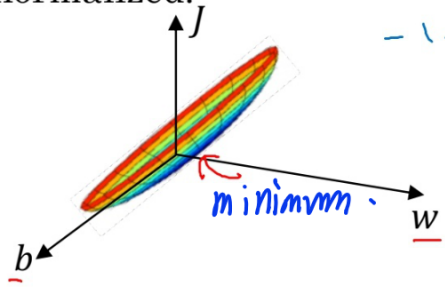
$$J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

w_1 x_1: 1....1000 ←

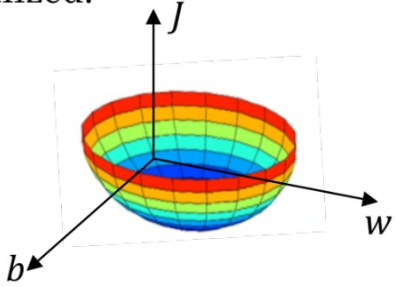
w_2 x_2: 0...1 ← range 매우 다름!

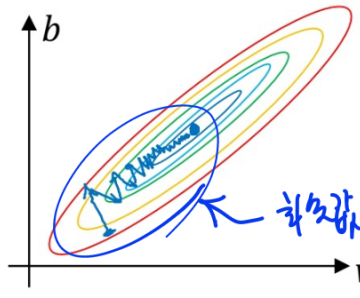
-1...1

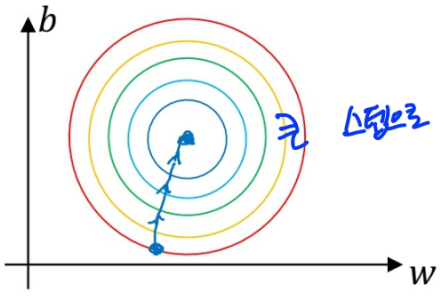
Unnormalized:



Normalized:







Andrew Ng

deeplearning.ai

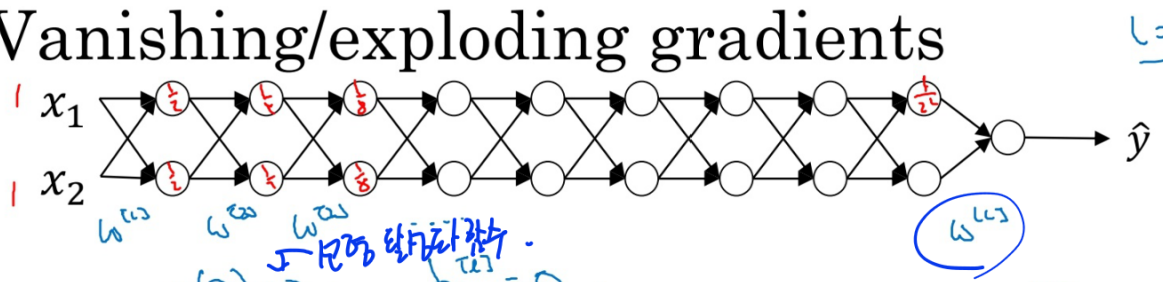
Setting up your optimization problem

Vanishing/exploding gradients

매우 깊은 신경망을 훈련시킬 때 미분값 · 기울기가 아주 작 or 커질 수 있다. → 무작위 가중치 초기화

L = 150 ··· 832.

Vanishing/exploding gradients



L=150

L

$g(z) = z$ 모두 항등함수라 가정. $b^{[l]} = 0$.

$\hat{y} = w^{[L]} w^{[L-1]} w^{[L-2]} \cdots a^{[3]} = w^{[3]} w^{[2]} w^{[1]} x$

1.5^L

0.5^L

$w^{[l]} > I$

$w^{[l]} < I$ [0.9 0.9]

$z^{[1]} = w^{[1]} x$

$a^{[1]} = g(z^{[1]}) = z^{[1]}$

$a^{[2]} = g(z^{[2]}) = g(w^{[2]} a^{[1]})$

$w^{[l]} = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$

$\hat{y} = w^{[L]} \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}^{L-1} x$

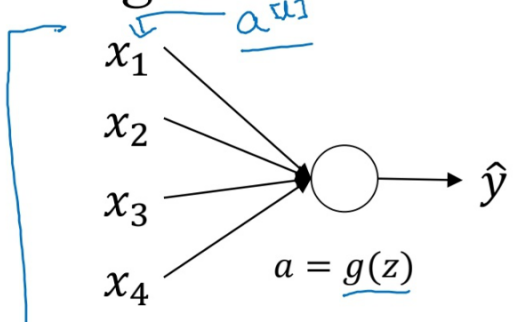
$1.5^{L-1} x$ ← 활성값 폭발

$0.5^{L-1} x$

Andrew Ng

Weight initialization for deep network.

Single neuron example



$w^{[l]}$

$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$

large n → smaller w_i

$Var(w_i) = \frac{1}{n}$ $\frac{2}{n}$ (ReLU에서)

$w^{[l]} = np.random.randn(shape) * np.sqrt\left(\frac{2}{n^{[l-1]}}\right)$

ReLU $g^{[l]}(z) = ReLU(z)$

O the vanishs:

tanh $\frac{1}{n^{[l-1]}}$

Xavier initialization 제이비에 초기화.

$\frac{2}{n^{[l-1]} + n^{[l]}}$

Andrew Ng

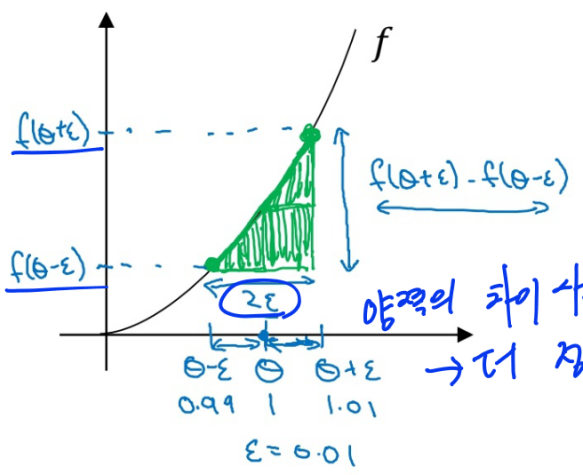
deeplearning.ai

Setting up your optimization problem

Numerical approximation of gradients

Checking your derivative computation

$f(\theta) = \theta^3$



양쪽의 차이 사용.
→더 정확하다!

$\frac{f(\theta+\varepsilon) - f(\theta-\varepsilon)}{2\varepsilon} \approx g(\theta)$

$\frac{(1.01)^3 - (0.99)^3}{2(0.01)} = 3.0001 \approx 3$

$g(\theta) = 3\theta^2 = 3$

approx error: 0.0001

(prev slide: 3.0301. error: 0.03)

$f'(\theta) = \lim_{\varepsilon \to 0} \frac{f(\theta+\varepsilon) - f(\theta-\varepsilon)}{2\varepsilon}$ $O(\varepsilon^2)$ 0.01 0.0001

$\frac{f(\theta+\varepsilon) - f(\theta)}{\varepsilon}$ error: $O(\varepsilon)$ 0.01

Andrew Ng

Checking your derivative computation

$f(\theta) = \theta^3$

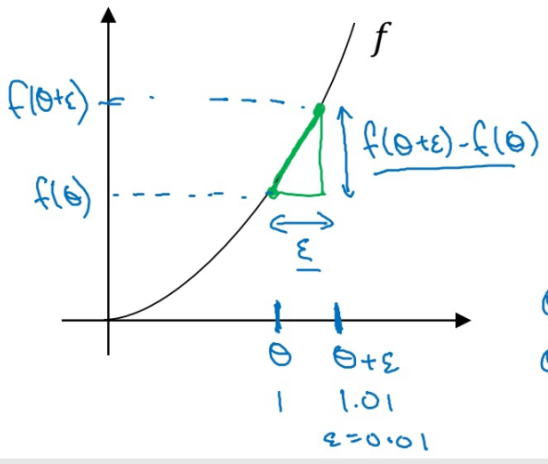
$\theta \in \mathbb{R}$.

$g(\theta) = \frac{d}{d\theta} f(\theta) = f'(\theta)$

$g(\theta) = 3\theta^2$.

$g(\theta) = 3 \cdot (1)^2 = 3$ when $\theta = 1$

$\frac{dw}{db}$



$\theta = 1$

$\theta + \varepsilon = 1.01$

$\frac{f(\theta+\varepsilon) - f(\theta)}{\varepsilon} \approx g(\theta)$

$\frac{(1.01)^3 - 1^3}{0.01} = 3.0301 \approx 3$

0.0301

3.1

3.2

Andrew Ng

deeplearning.ai

Setting up your optimization problem

Gradient Checking implementation notes

Gradient check for a neural network

Take $W^{[1]}, b^{[1]}, \dots, W^{[L]}, b^{[L]}$ and reshape into a big vector θ.

concatenate

$J(W^{[1]}, b^{[1]}, \dots, W^{[L]}, b^{[L]}) = J(\theta)$.

Take $dW^{[1]}, db^{[1]}, \dots, dW^{[L]}, db^{[L]}$ and reshape into a big vector $d\theta$.

concatenate

Is $d\theta$ the gradient of $J(\theta)$?

Gradient checking (Grad check)

for each i:

$$d\theta_{approx}[i] = \frac{J(\theta_1, \theta_2, \dots, \theta_i + \varepsilon, \dots) - J(\theta_1, \theta_2, \dots, \theta_i - \varepsilon, \dots)}{2\varepsilon}$$

$$\approx d\theta[i] = \frac{\partial J}{\partial \theta_i}$$

$d\theta_{approx} \overset{?}{\approx} d\theta$

Check

$$\frac{\|d\theta_{approx} - d\theta\|_2}{\|d\theta_{approx}\|_2 + \|d\theta\|_2} \approx \begin{cases} 10^{-7} & \text{great!} \\ 10^{-5} & \\ 10^{-3} & \text{worry.} \end{cases}$$

$\varepsilon = 10^{-7}$

Andrew Ng

deeplearning.ai

Setting up your optimization problem

Gradient Checking

Gradient checking implementation notes

- Don't use in training – only to debug

- If algorithm fails grad check, look at components to try to identify bug.

- Remember regularization.

- Doesn't work with dropout.

- Run at random initialization; perhaps again after some training.

Andrew Ng