



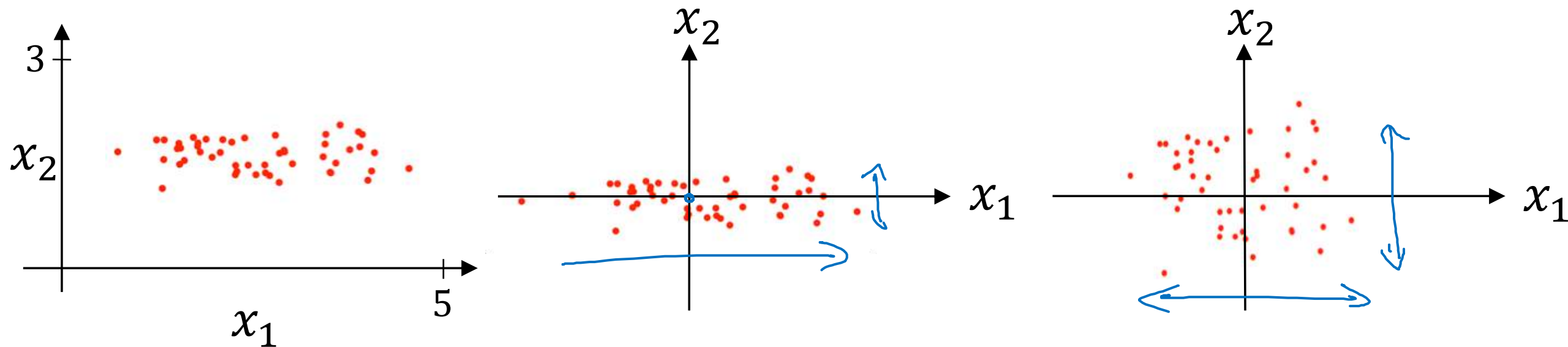
deeplearning.ai

Setting up your
optimization problem

Normalizing inputs

Normalizing training sets

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



Subtract mean:

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

$$x := x - \mu$$

Normalize variance

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n x^{(i)} * x^{(i)T}$$

← element-wise

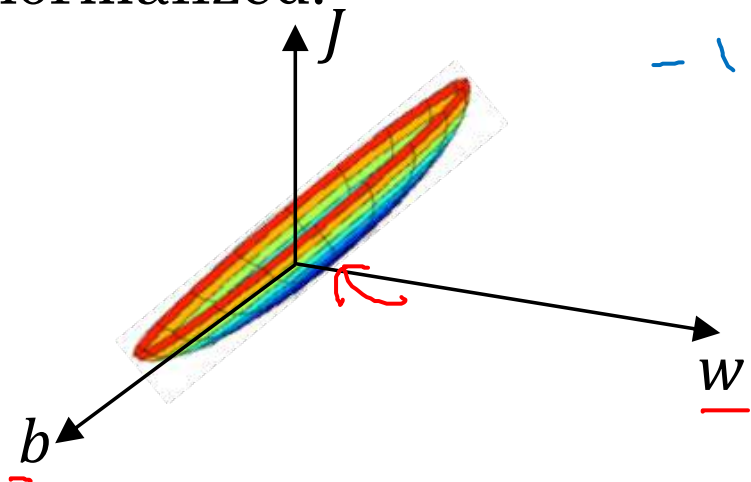
$$x /= \sigma$$

Use same μ σ^2 to normalize test set.

Why normalize inputs?

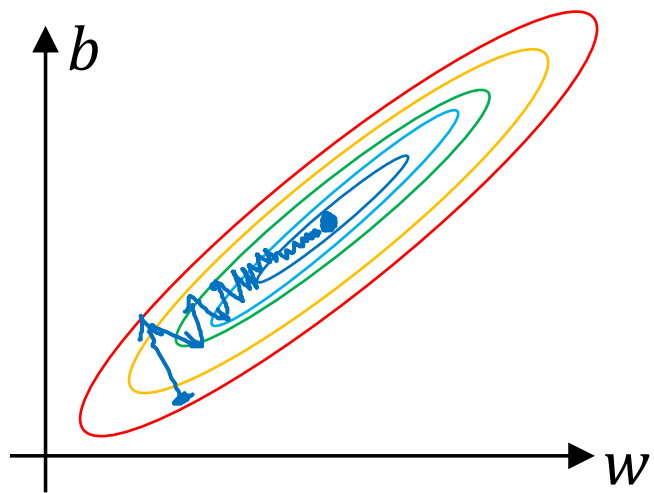
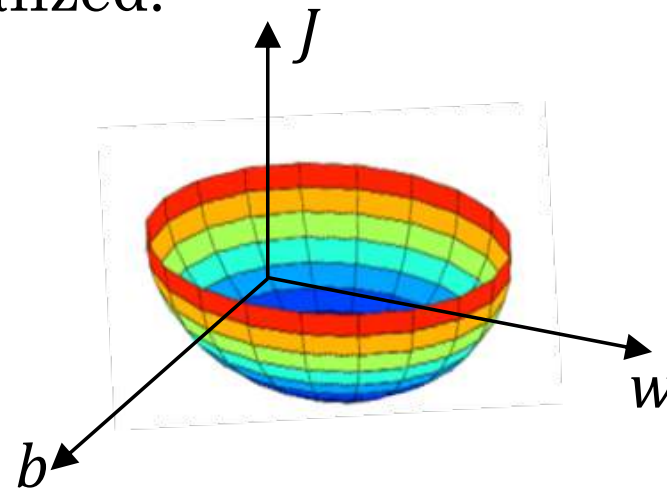
$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Unnormalized:

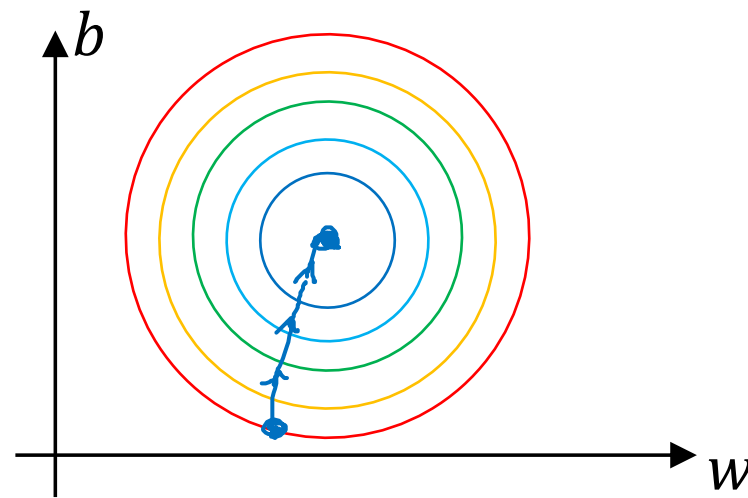


$w_1: x_1: \underline{1 \dots 1000} \leftarrow$
 $w_2: x_2: \underline{0 \dots 1} \leftarrow$
 $\quad \quad \quad -1 \dots 1$

Normalized:



$x_1: 0 \dots 1$
 $x_2: -1 \dots 1$
 $x_3: 1 \dots 2$



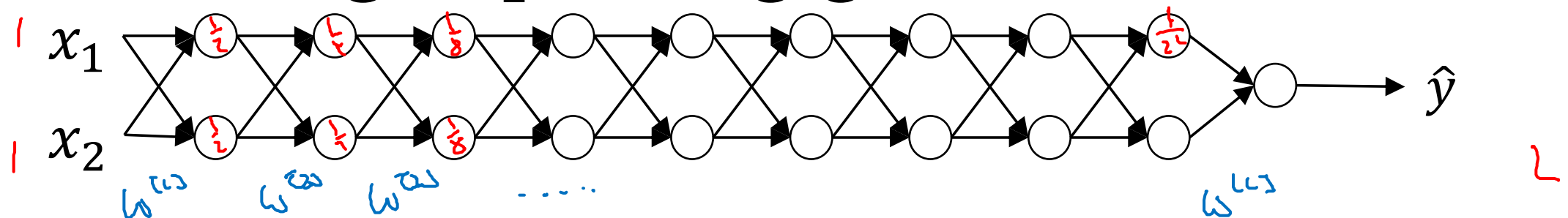


deeplearning.ai

Setting up your
optimization problem

Vanishing/exploding
gradients

Vanishing/exploding gradients



$g(z) = z$ $b^{(2)} = 0$

$\hat{y} = w^{(L,2)} \underbrace{w^{(L-1,2)} w^{(L-2,2)} \dots w^{(2,2)} w^{(1,2)}}_{a^{(1,2)}} x$

1.5^L
 0.5^L

$w^{(1,2)} > I$

$w^{(2,2)} < I$ $\begin{bmatrix} 0.9 & \\ & 0.9 \end{bmatrix}$

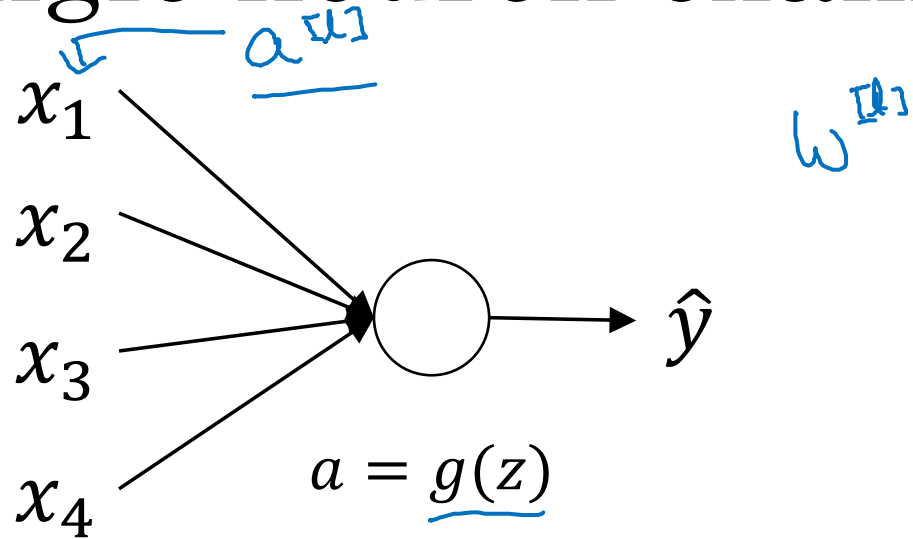
$w^{(2,2)} = \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}$

$z^{(1,2)} = w^{(1,2)} x$
 $a^{(1,2)} = g(z^{(1,2)}) = z^{(1,2)}$
 $a^{(2,2)} = g(z^{(2,2)}) = g(w^{(2,2)} a^{(1,2)})$

$\hat{y} = w^{(L,2)} \begin{bmatrix} 1.5 & 0 \\ 0 & 1.5 \end{bmatrix}^{L-1} x$

$1.5^{L-1} x$
 $0.5^{L-1} x$

Single neuron example



$$z = w_1 x_1 + w_2 x_2 + \dots + w_n x_n \quad \text{to}$$

large $n \rightarrow$ Smaller w_i

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

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

ReLU $g^{[2]}(z) = \text{ReLU}(z)$

Other variants:

tanh

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

Xavier initialization ↑

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

↑



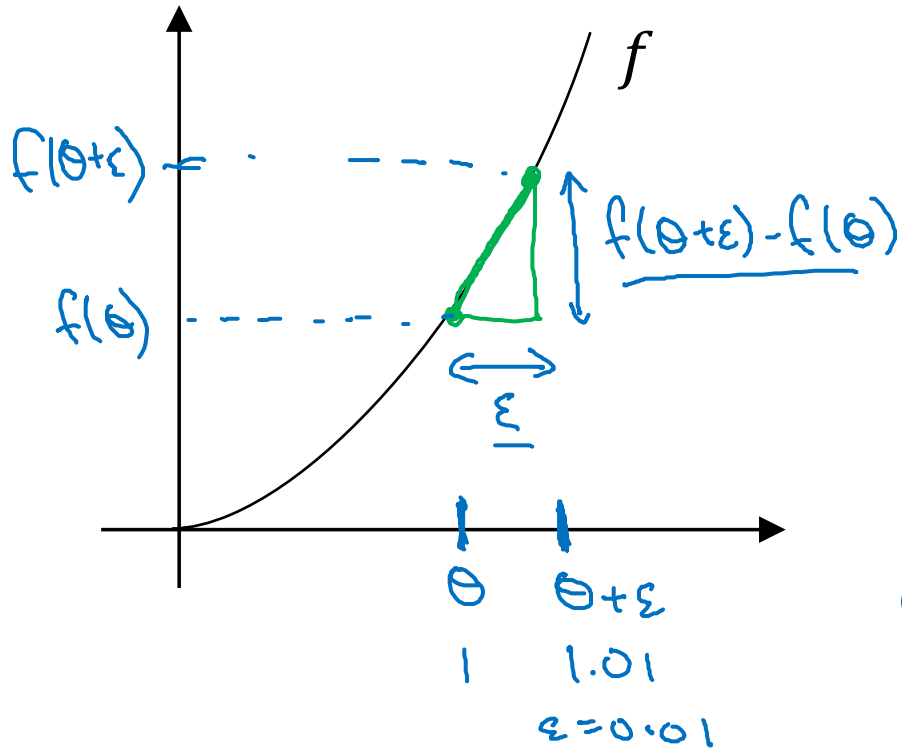
deeplearning.ai

Setting up your optimization problem

Numerical approximation of gradients

Checking your derivative computation

I $f(\theta) = \theta^3$
 $\theta \in \mathbb{R}.$



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

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

$\frac{dw}{db}$

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

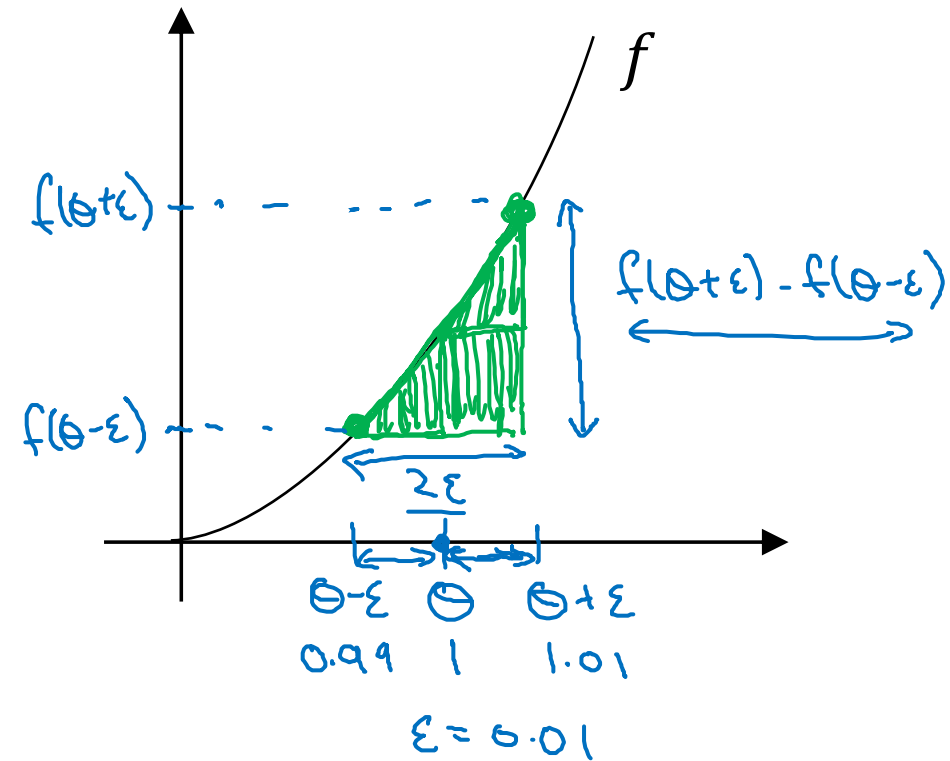
$$\frac{f(\theta + \epsilon) - f(\theta)}{\epsilon} \approx g(\theta)$$

$$\frac{(1.01)^3 - 1^3}{0.01} = \frac{1.030301 - 1}{0.01} = \frac{0.0301}{0.01} = 3.0301 \approx 3$$

Annotations: $\theta = 1$, $\theta + \epsilon = 1.01$, $\epsilon = 0.01$, 3.1 , 3.2

Checking your derivative computation

$$\underline{f(\theta) = \theta^3}$$



$$\left[\frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2\epsilon} \approx \underline{g(\theta)} \right]$$

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

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

approx error: 0.0001

(prev slide: 3.0301. error: 0.03)

$$\left\{ \begin{array}{l} f'(\theta) = \lim_{\epsilon \rightarrow 0} \frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2\epsilon} \quad \begin{array}{l} O(\epsilon^2) \\ 0.01 \\ \underline{0.0001} \end{array} \quad \left| \quad \begin{array}{l} \frac{f(\theta + \epsilon) - f(\theta)}{\epsilon} \quad \text{error: } O(\epsilon) \\ \quad \quad \quad 0.01 \end{array} \end{array} \right.$$



deeplearning.ai

Setting up your
optimization problem

Gradient Checking

Gradient check for a neural network

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

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

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

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

Gradient checking (Grad check)

$$J(\theta) = J(\theta_1, \theta_2, \theta_3, \dots)$$

for each i :

$$\rightarrow \underline{d\theta_{\text{approx}}[i]} = \frac{J(\theta_1, \theta_2, \dots, \overset{\downarrow}{\theta_i + \epsilon}, \dots) - J(\theta_1, \theta_2, \dots, \overset{\downarrow}{\theta_i - \epsilon}, \dots)}{2\epsilon}$$

$$\approx \underline{d\theta[i]} = \frac{\partial J}{\partial \theta_i} \quad | \quad d\theta_{\text{approx}} \approx d\theta$$

Checks

$$\rightarrow \frac{\|d\theta_{\text{approx}} - d\theta\|_2}{\|d\theta_{\text{approx}}\|_2 + \|d\theta\|_2}$$
$$\underline{\epsilon = 10^{-7}}$$

$$\approx \frac{10^{-7}}{10^{-5}} - \text{great!} \leftarrow$$
$$\rightarrow 10^{-3} - \text{worry.} \leftarrow$$



deeplearning.ai

Setting up your
optimization problem

Gradient Checking
implementation notes

Gradient checking implementation notes

- Don't use in training – only to debug

$$\frac{d\theta_{\text{approx}}[\vec{i}]}{\uparrow \uparrow} \longleftrightarrow \frac{d\theta[\vec{i}]}{\uparrow}$$

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

$$\frac{db^{[L]}}{\uparrow} \quad \frac{dW^{[L]}}{\uparrow}$$

- Remember regularization.

$$\underline{J(\theta)} = \frac{1}{n} \sum_i \ell(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2n} \sum_l \|W^{[l]}\|_F^2$$

$d\theta = \text{gradient of } J \text{ wrt. } \theta$

- Doesn't work with dropout.

J

$$\underline{\text{keep-prob} = 1.0}$$

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

$$\underline{W, b \approx 0}$$