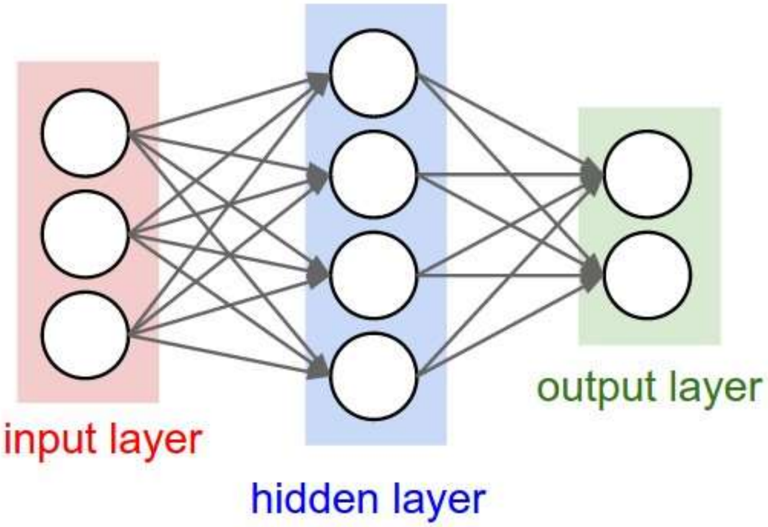
deeplearning.ai

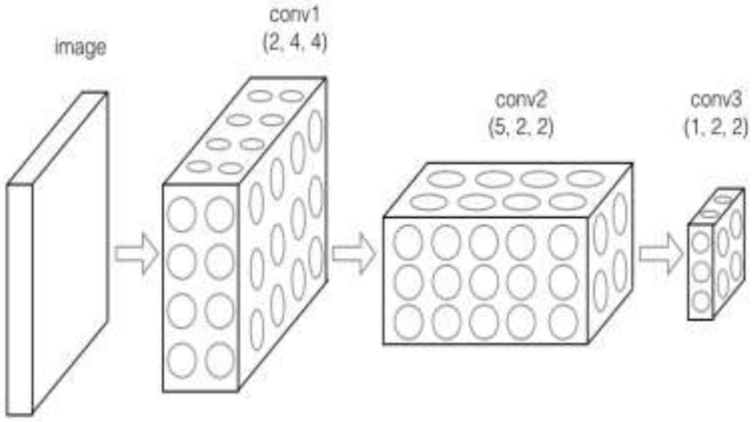
Introduction to Deep Learning

Supervised Learning with Neural Networks

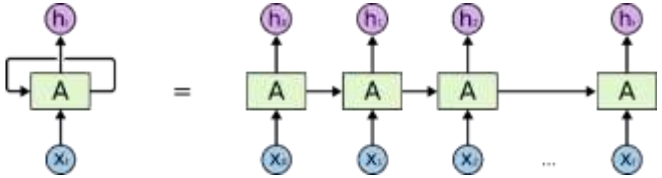
Neural Network examples



Standard NN



Convolutional NN

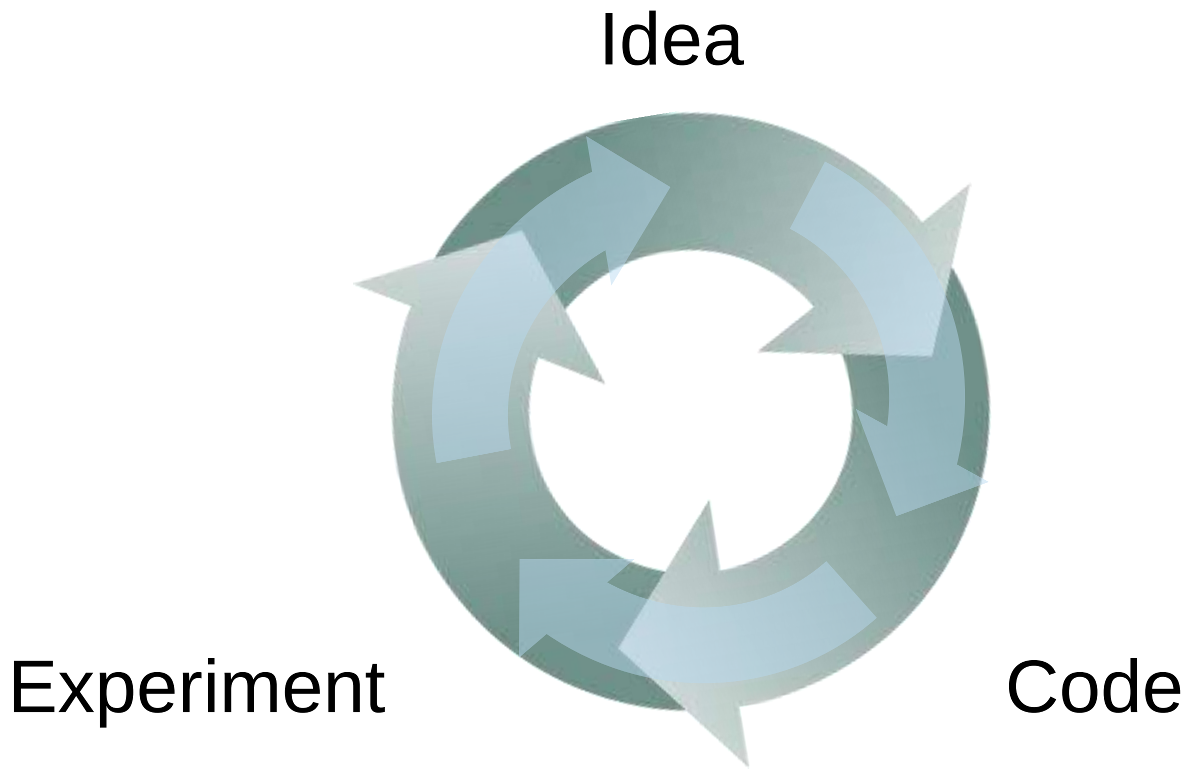


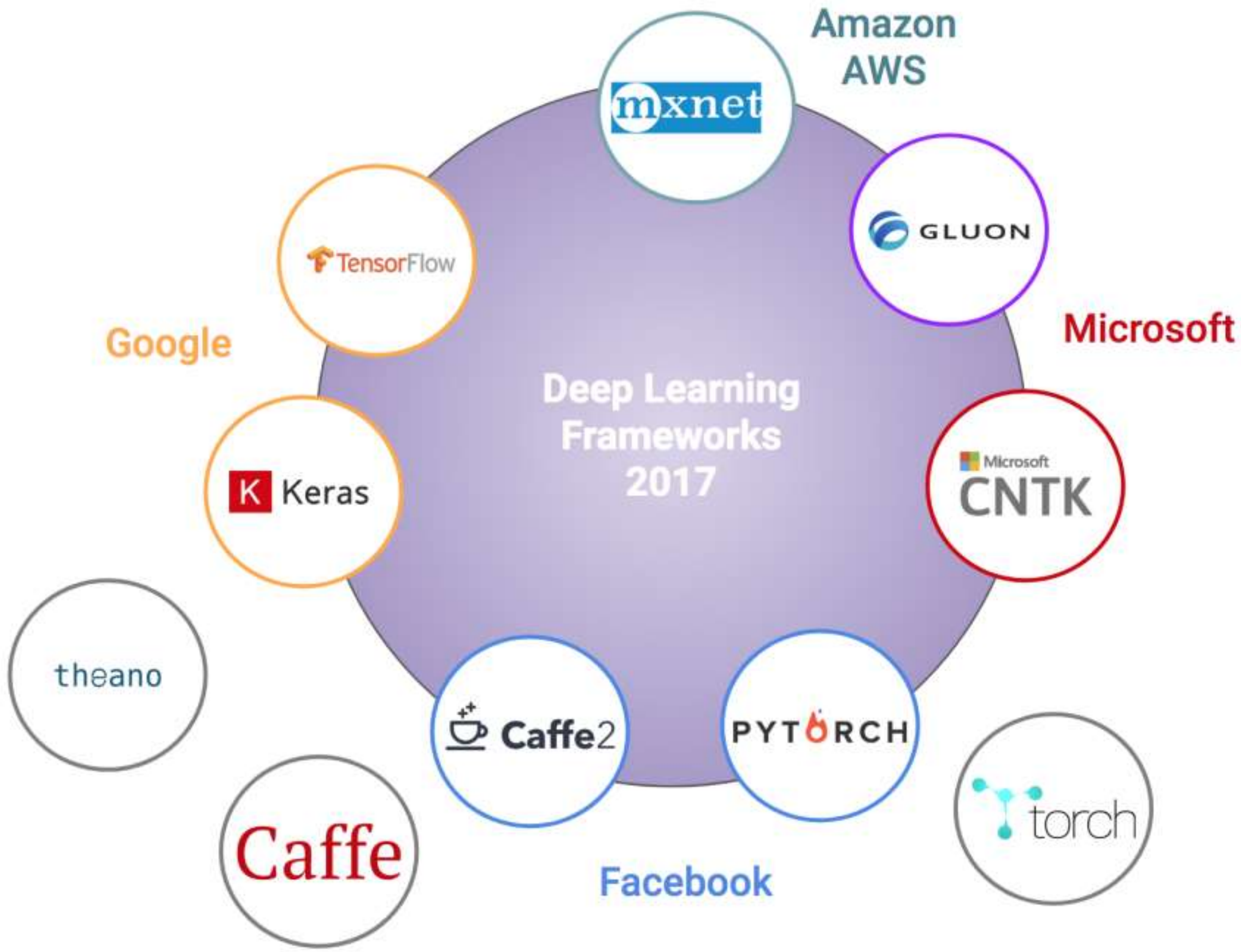
Recurrent NN

Why is Deep Learning taking off?

- More data
- More computational resources
- New algorithms

Deep learning is highly iterative process

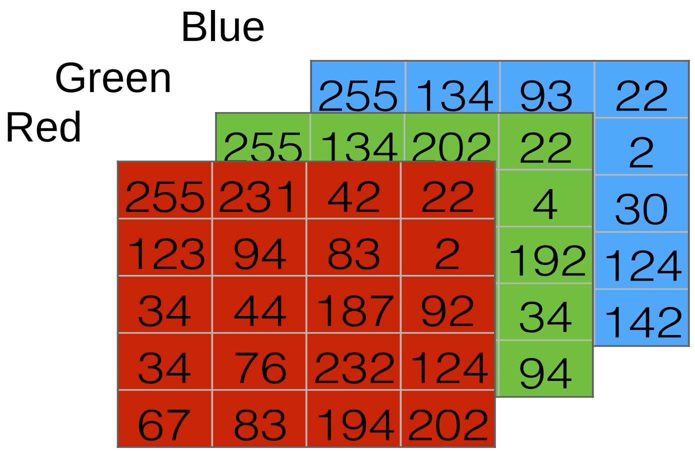




Amazon
AWS
mxnet
GLUON
Microsoft
TensorFlow
Google
Deep Learning
Frameworks
2017
Keras
Microsoft
CNTK
theano
Caffe2
PYTORCH
torch
Caffe
Facebook

Binary Classification

1 (cat) vs 0 (non cat)



Notation

Logistic Regression

http://vision.stanford.edu/teaching/cs231n-demos/linear-classify/

Logistic Regression cost function

$\hat{y} = \sigma(w^T x + b)$, where $\sigma(z) = \frac{1}{1+e^{-z}}$

Given $\left\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\right\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

Loss (error) function:

Logistic regression cost function

$$\left.\begin{array}{ll} \boxed{\text{If } \quad y = 1: \qquad p(y|x) = \hat{y}} \\ \boxed{\text{If } \quad y = 0: \qquad p(y|x) = 1 - \hat{y}} \end{array}\right\} \; p(y|x)$$

$$\boxed{p(y|x) = \underbrace{\hat{y}^{y}}\ \underbrace{(1-\hat{y})^{(1-y)}}}$$

If $y=1$: $\underline{p(y|x) = \hat{y}}\ \underbrace{(1-\hat{y})^{0}}_{=1}$

$\underline{\text{If } y=0}$: $\underline{p(y|x)} = \underbrace{\hat{y}^{0}}_{=1}\ (1-\hat{y})^{(1-y)} = 1 \times (1-\hat{y}) = \underline{1-\hat{y}}$

$$\log \underbrace{p(y|x)} = \log \hat{y}^{y}(1-\hat{y})^{(1-y)} = y\log\hat{y} + (1-y)\log(1-\hat{y})$$

$$= -\mathcal{L}(\hat{y}, y)$$

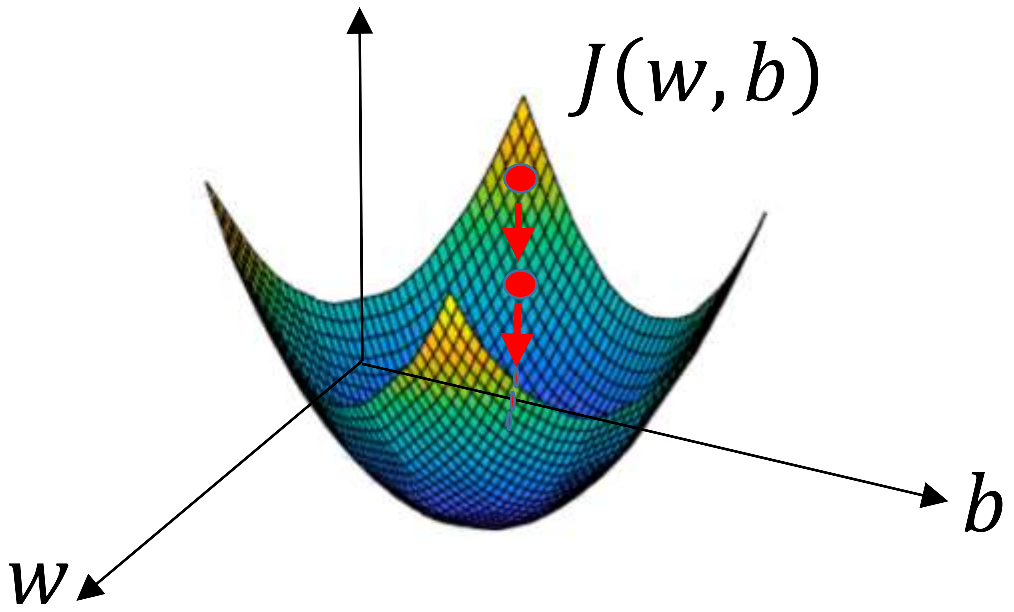
Andrew Ng

Gradient Descent

Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$

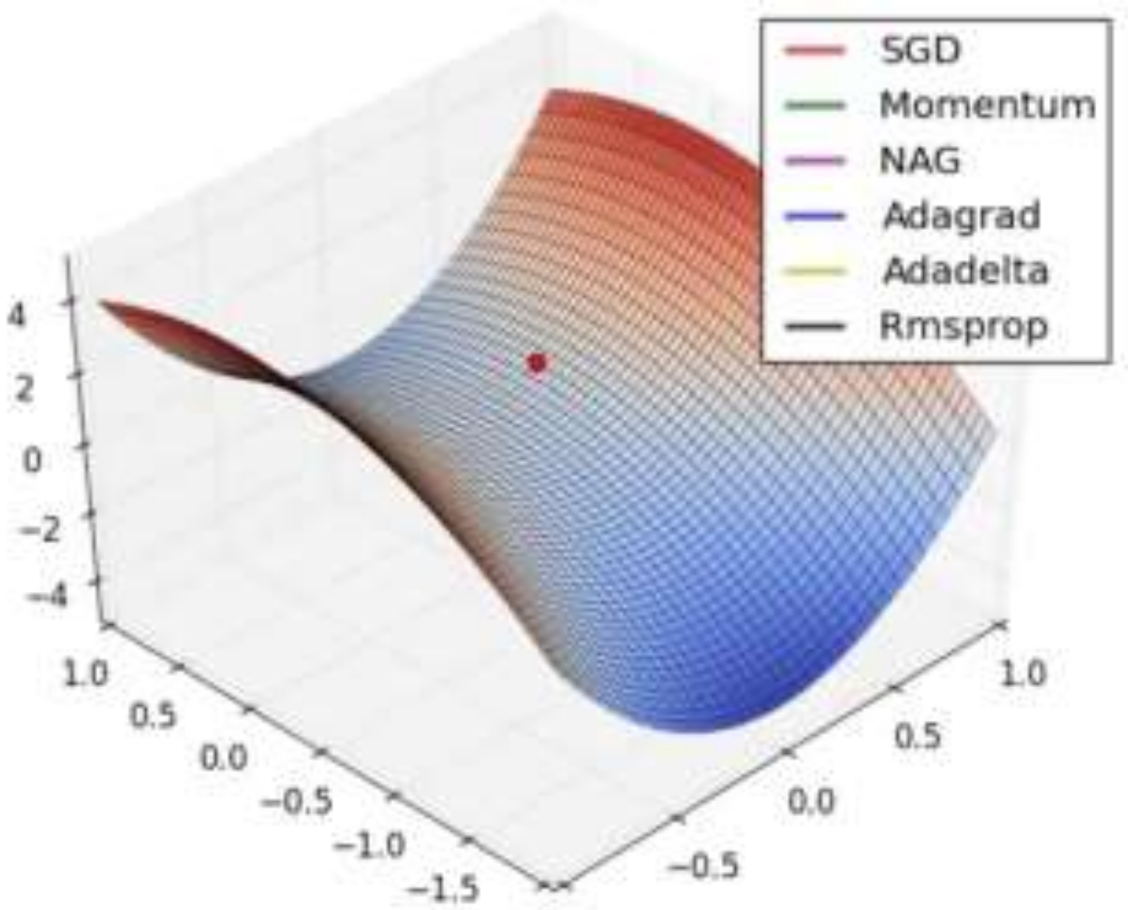
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})$$

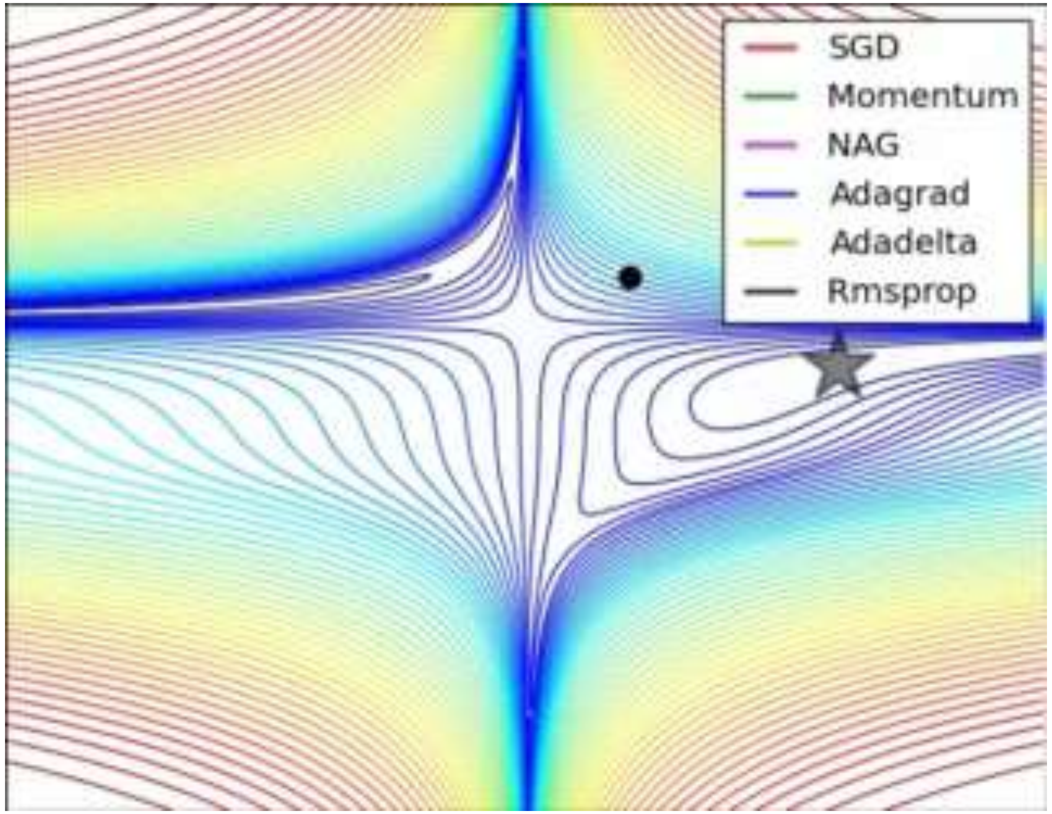
Want to find w, b that minimize $J(w,b)$



Gradient Descent

Optimization methods





deeplearning.ai

Basics of Neural Network Programming

Computation Graph

Derivatives

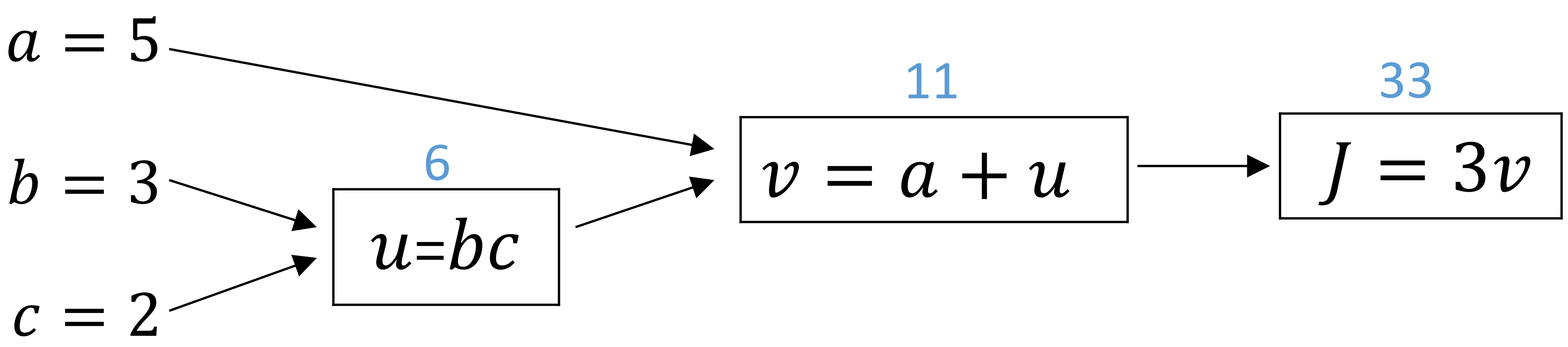
$$f(x,y) = x + y \rightarrow \frac{\partial f}{\partial x} = 1 \quad \frac{\partial f}{\partial y} = 1$$

$$f(x,y) = x \cdot y \rightarrow \frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x$$

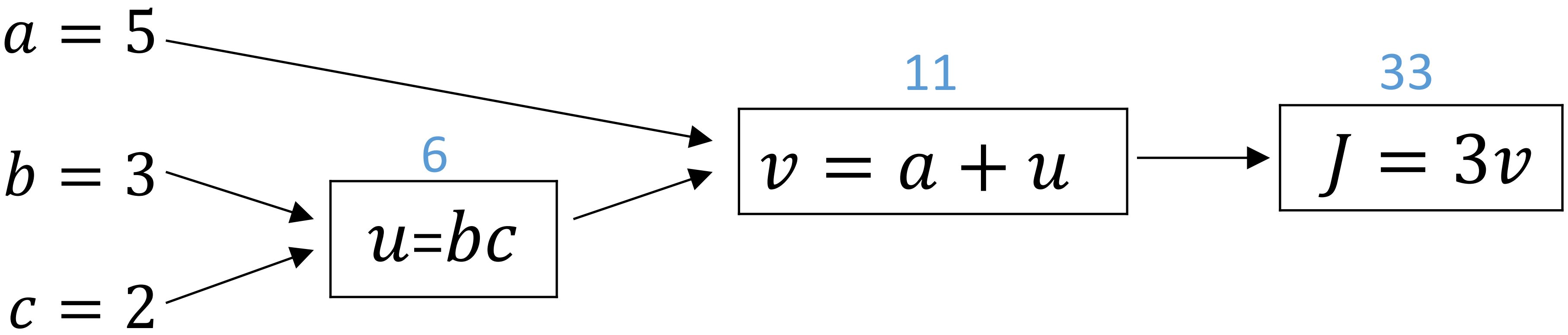
$$f(x) = y(z(x)) \rightarrow \frac{df}{dx} = \frac{df}{dy} \cdot \frac{dy}{dz} \cdot \frac{dz}{dx}$$

Computation Graph

Computing derivatives

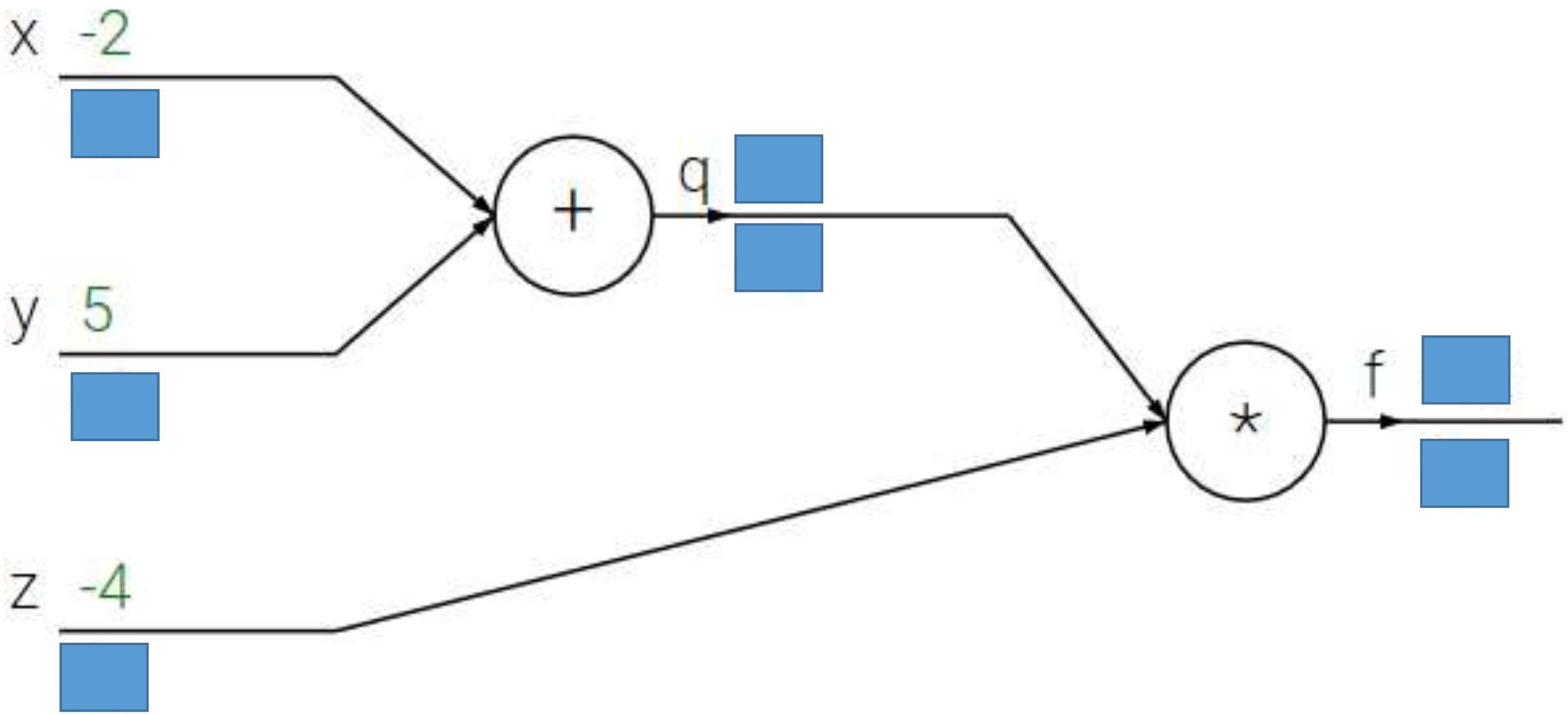


Computing derivatives



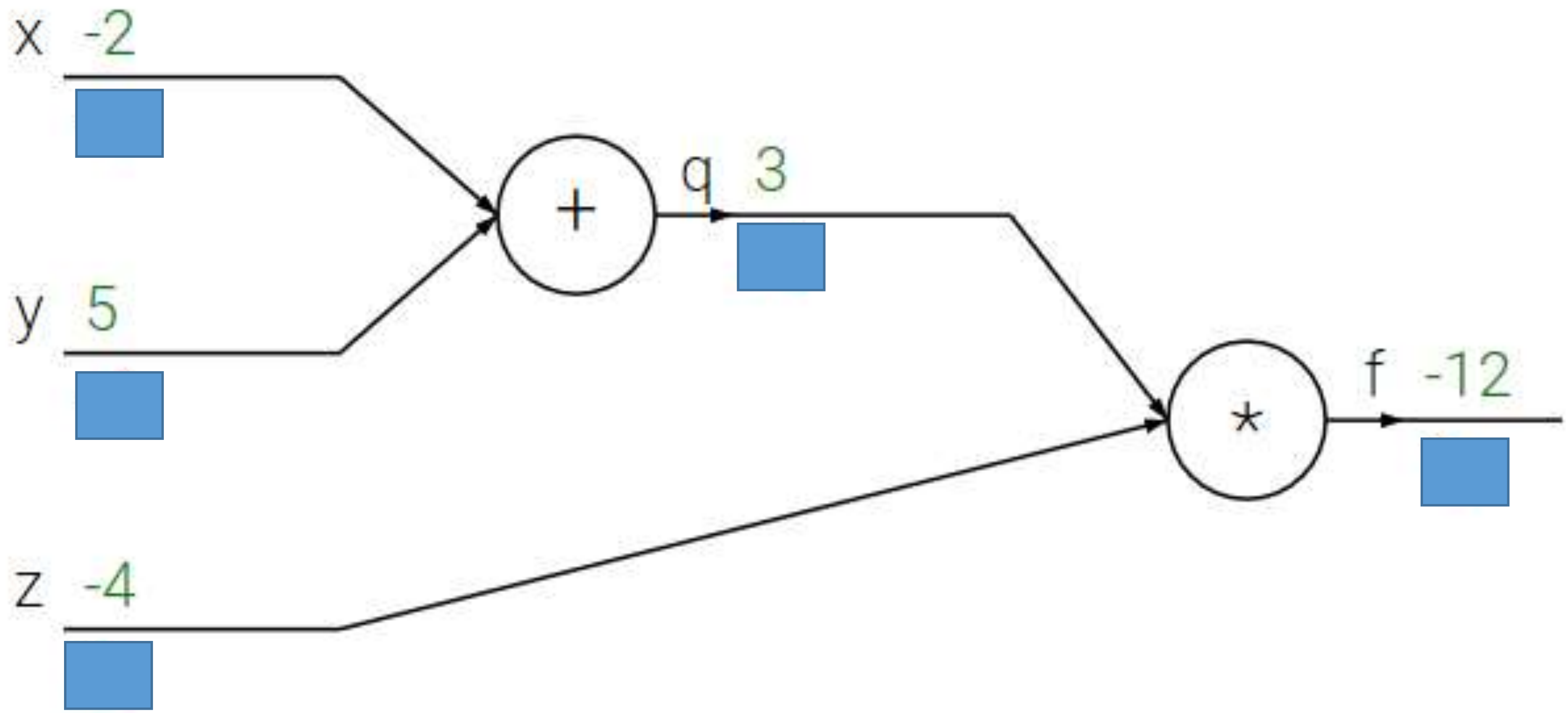
Computational graph

f(x, y, z) = (x + y) * z



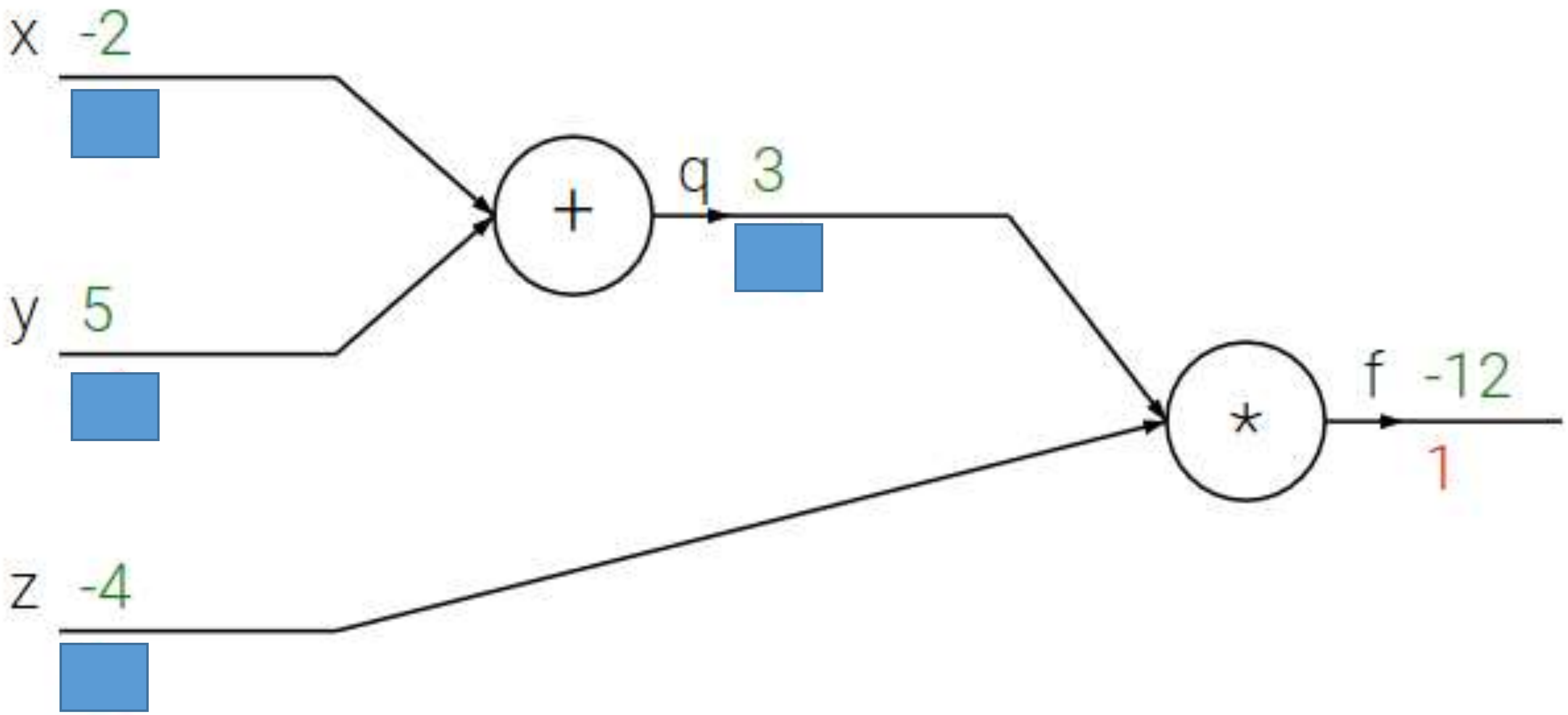
Computational graph

$$f(x, y, z) = (x + y) * z$$



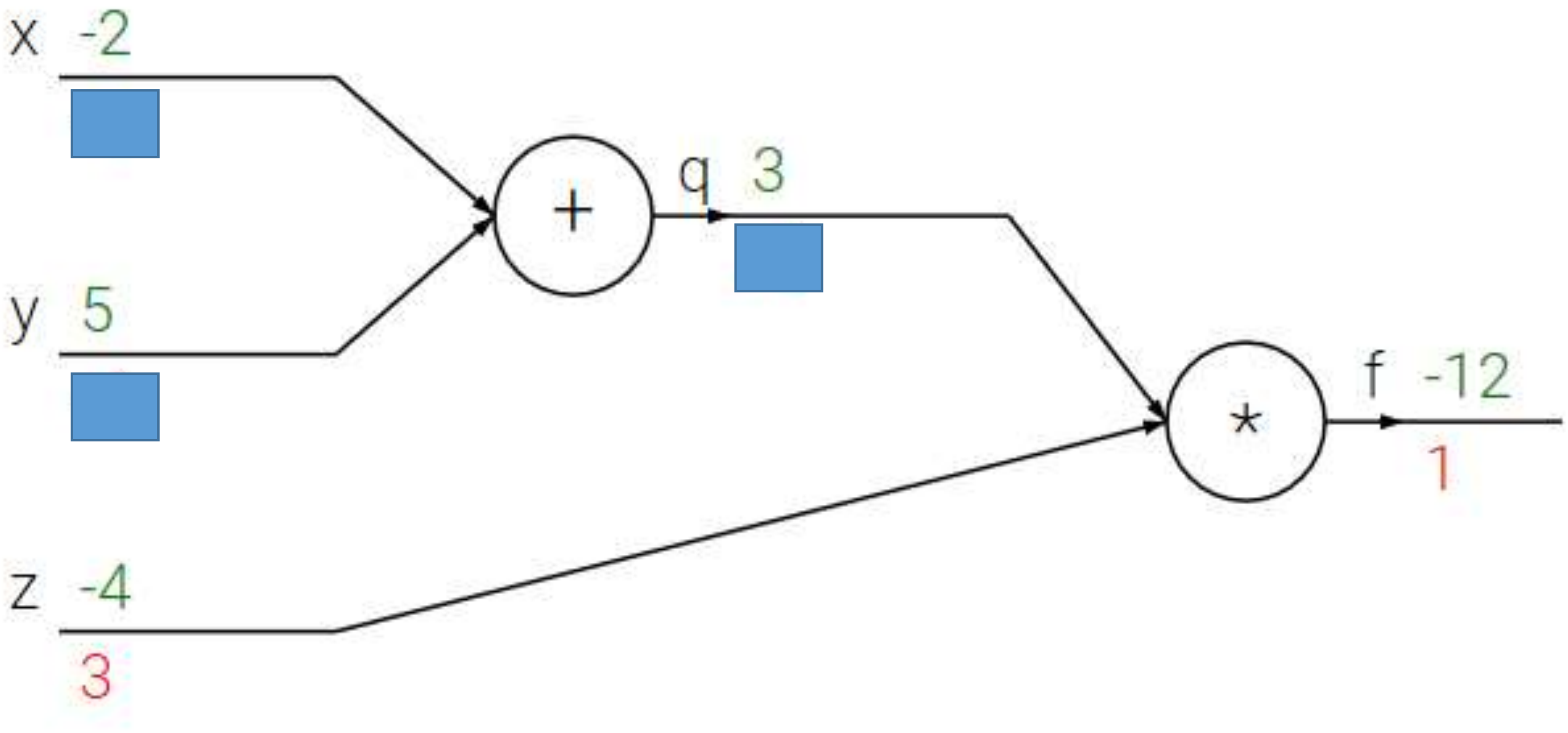
Computational graph

$$f(x, y, z) = (x + y) * z$$



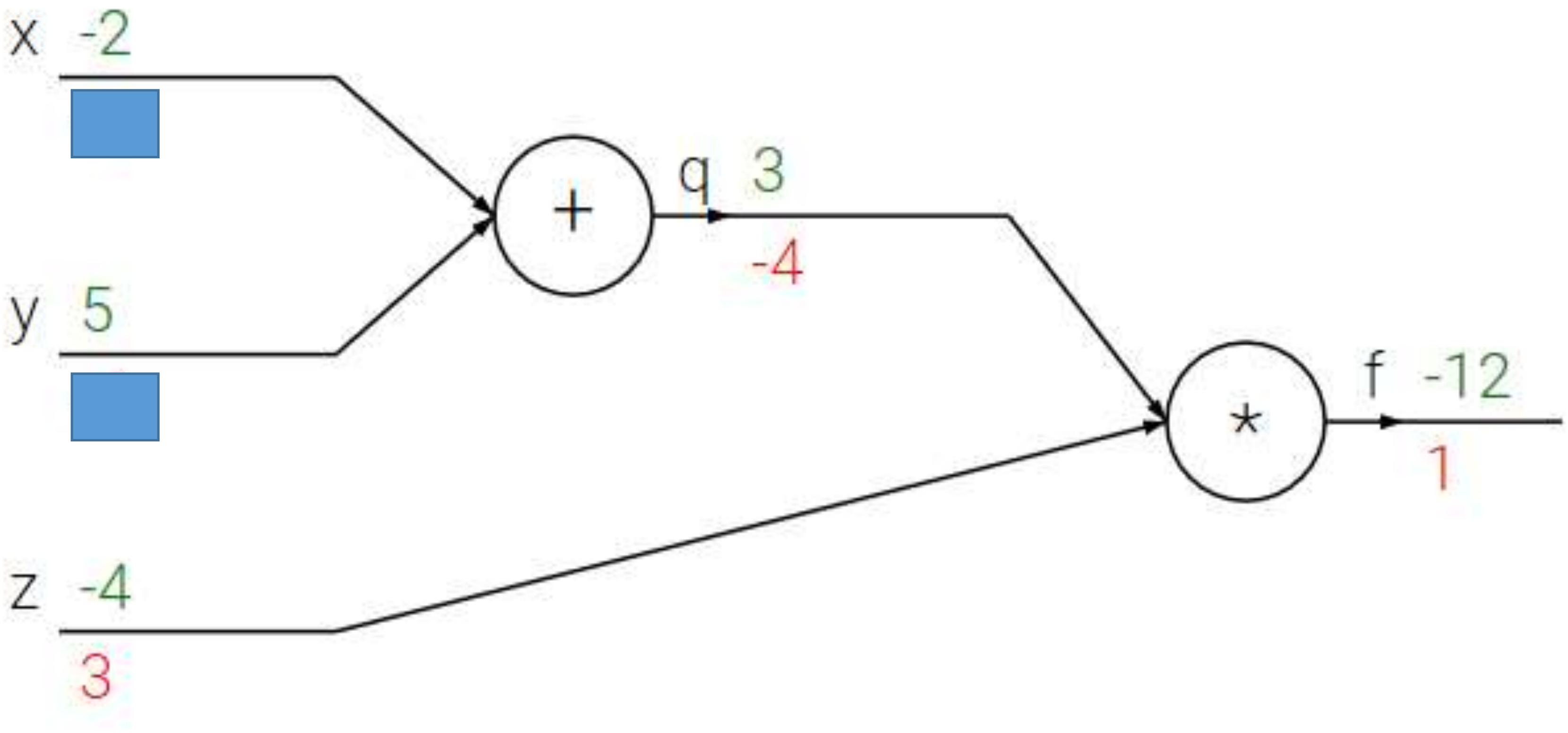
Computational graph

$$f(x, y, z) = (x + y) * z$$



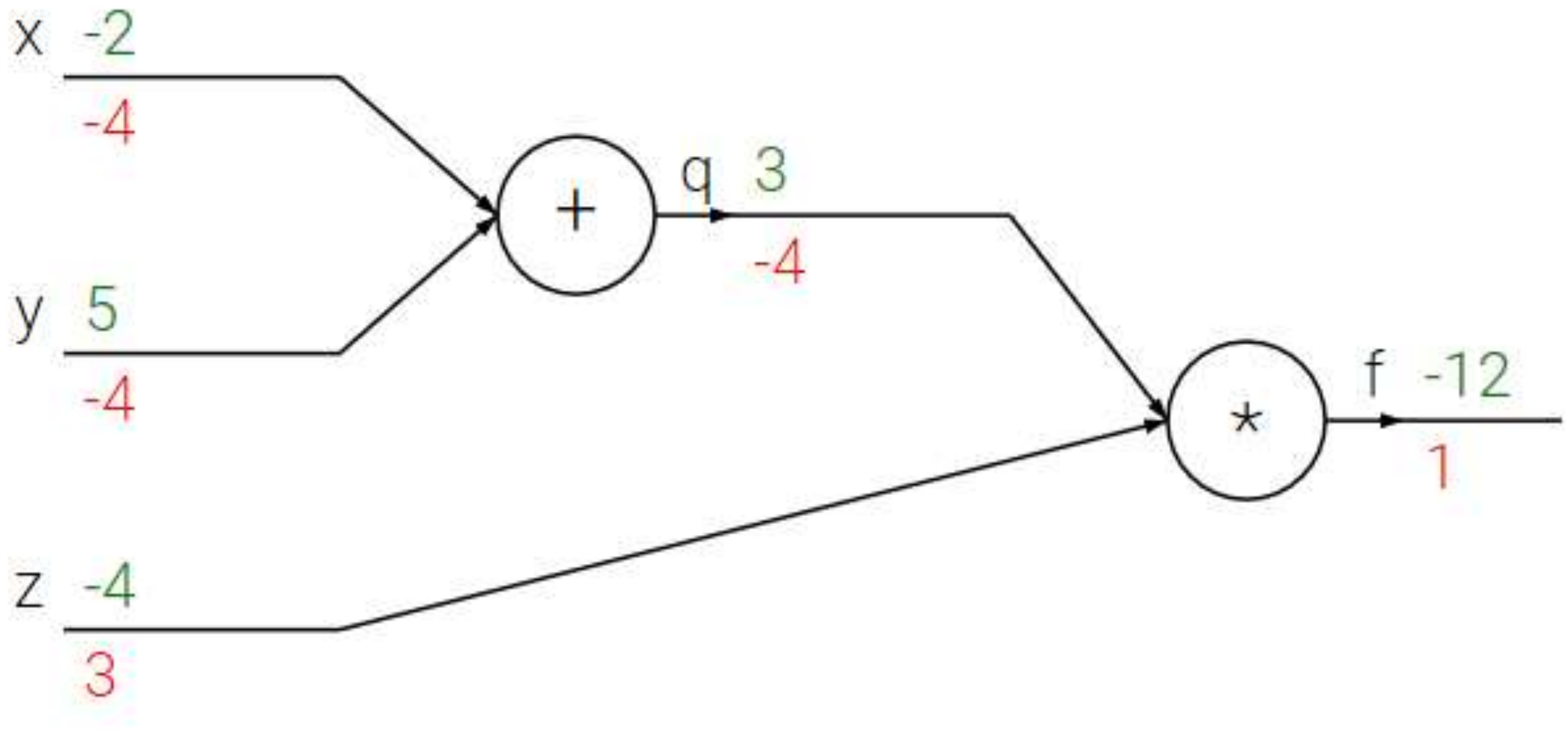
Computational graph

f(x, y, z) = (x + y) * z



Computational graph

$$f(x, y, z) = (x + y) * z$$



Logistic regression

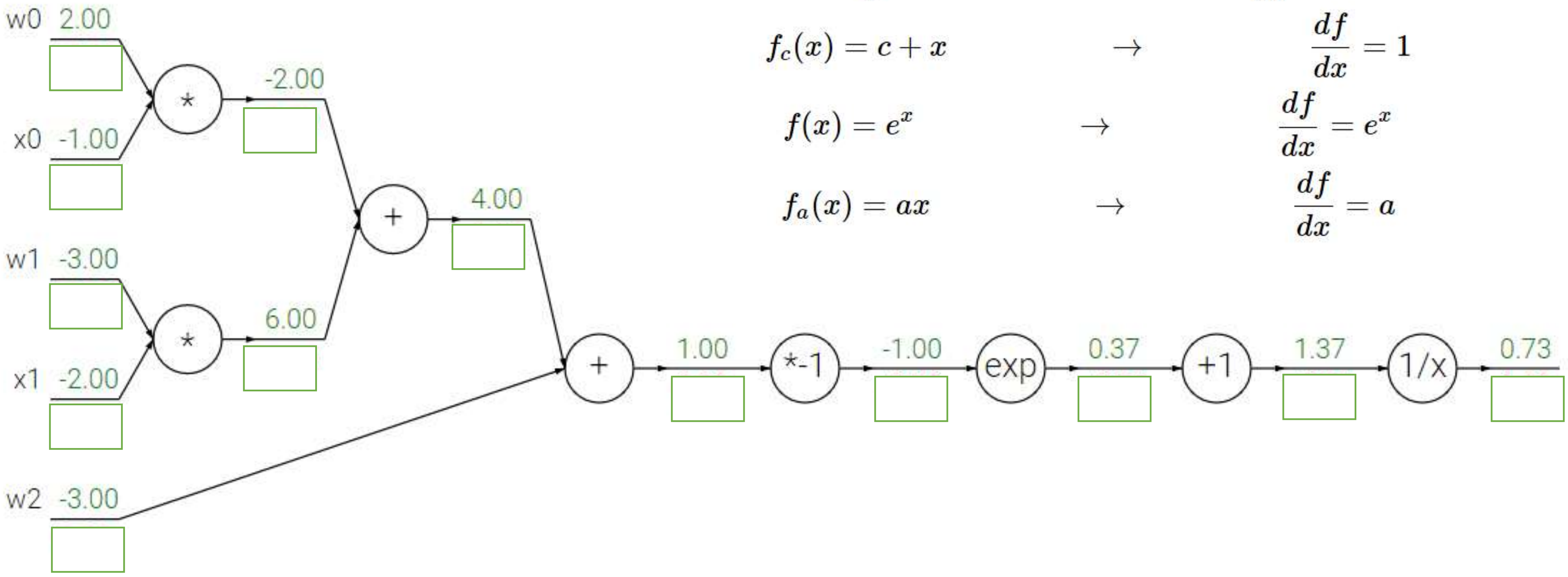
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$



Logistic regression

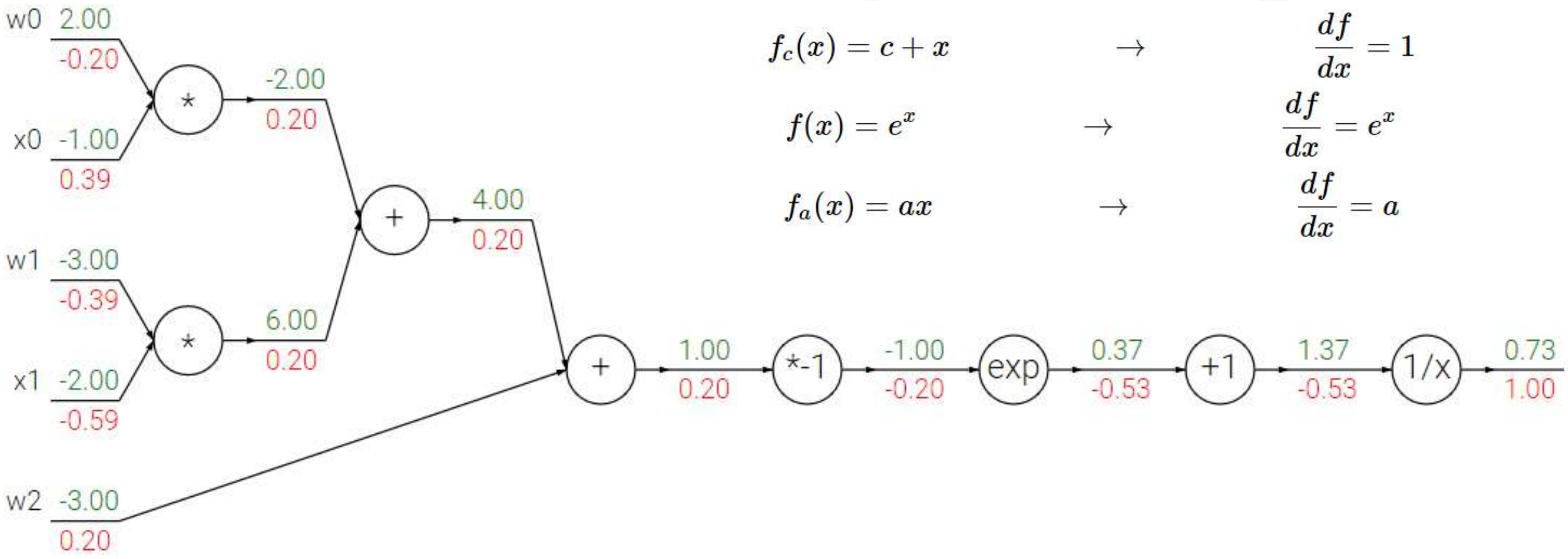
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$



Basics of Neural Network Programming

deeplearning.ai

Logistic Regression Gradient descent

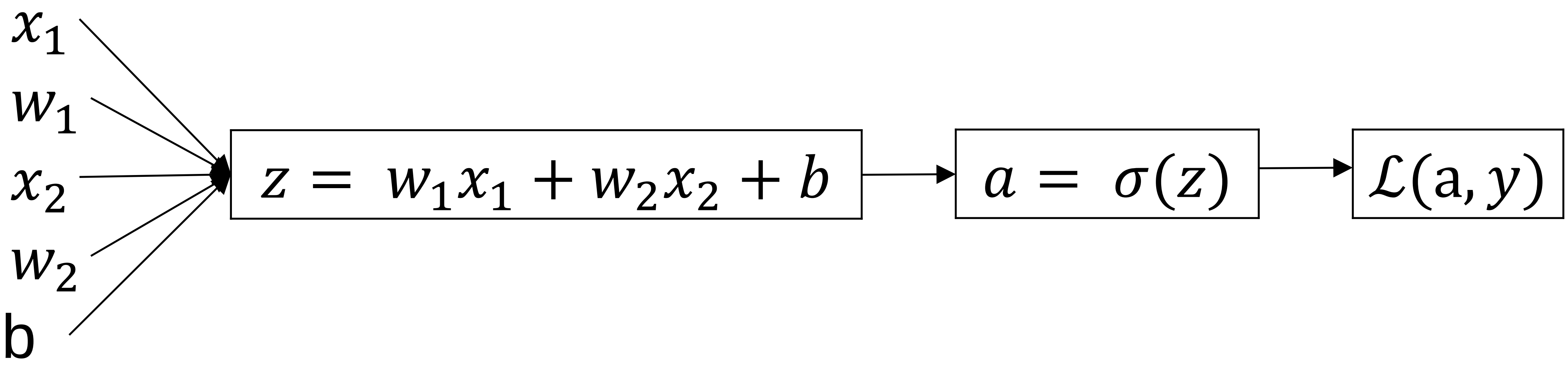
Logistic regression recap

$$z = w^T x + b$$

$$\hat{y} = a = \sigma(z)$$

$$\mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$

Logistic regression derivatives



Logistic regression on m examples

Logistic regression on *m* examples

Basics of Neural Network Programming

Vectorization

deeplearning.ai

What is vectorization?

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$$

```
u = np.zeros((n,1))
for i in range(n):
    u[i]=math.exp(v[i])
```

Logistic regression derivatives

```
J = 0, dw1 = 0, dw2 = 0, db = 0
for i = 1 to n:
```

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J \mathrel{+}= -\left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})\right]$

$\mathrm{d}z^{(i)} = a^{(i)}(1 - a^{(i)})$

$\mathrm{d}w_1 \mathrel{+}= x_1^{(i)} \mathrm{d}z^{(i)}$

$\mathrm{d}w_2 \mathrel{+}= x_2^{(i)} \mathrm{d}z^{(i)}$

$\mathrm{db} \mathrel{+}= \mathrm{d}z^{(i)}$

$\mathrm{J} = \mathrm{J}/\mathrm{m},\ \mathrm{d}w_1 = \mathrm{d}w_1/\mathrm{m},\ \mathrm{d}w_2 = \mathrm{d}w_2/\mathrm{m},\ \mathrm{db} = \mathrm{db}/\mathrm{m}$

Basics of Neural Network Programming

Vectorizing Logistic Regression

deeplearning.ai

Vectorizing Logistic Regression

$z^{(1)} = w^T x^{(1)} + b$ $\quad$ $z^{(2)} = w^T x^{(2)} + b$ $\quad$ $z^{(3)} = w^T x^{(3)} + b$

$a^{(1)} = \sigma(z^{(1)})$ $\quad$ $a^{(2)} = \sigma(z^{(2)})$ $\quad$ $a^{(3)} = \sigma(z^{(3)})$

Vectorizing Logistic Regression

Implementing Logistic Regression

J = 0, dw_1 = 0, dw_2 = 0, db = 0

for i = 1 to m:

$$z^{(i)} = w^T x^{(i)} + b$$

$$a^{(i)} = \sigma(z^{(i)})$$

$$J \mathrel{+}= -\left[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})\right]$$

$$\mathrm{d}z^{(i)} = a^{(i)} - y^{(i)}$$

$$\mathrm{d}w_1 \mathrel{+}= x_1^{(i)} \mathrm{d}z^{(i)}$$

$$\mathrm{d}w_2 \mathrel{+}= x_2^{(i)} \mathrm{d}z^{(i)}$$

$$\mathrm{d}b \mathrel{+}= \mathrm{d}z^{(i)}$$

J = J/m, dw_1 = dw_1/m, dw_2 = dw_2/m

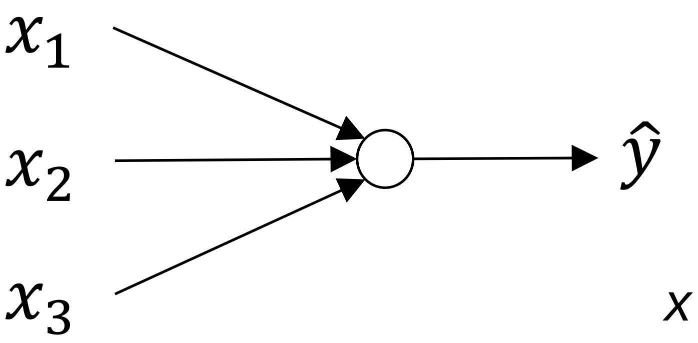
db = db/m

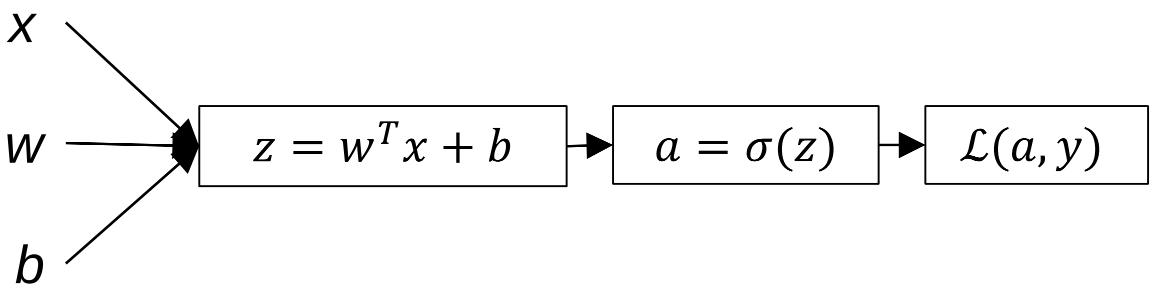
One hidden layer Neural Network

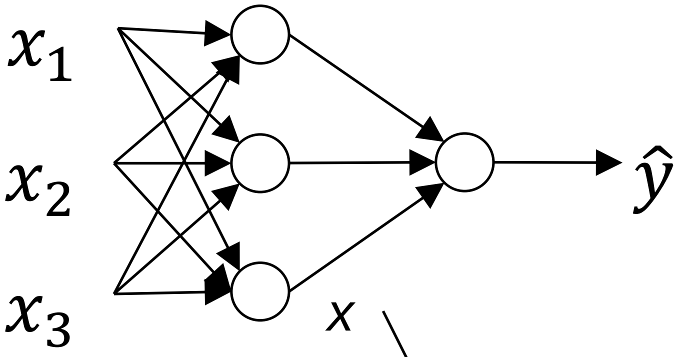
deeplearning.ai

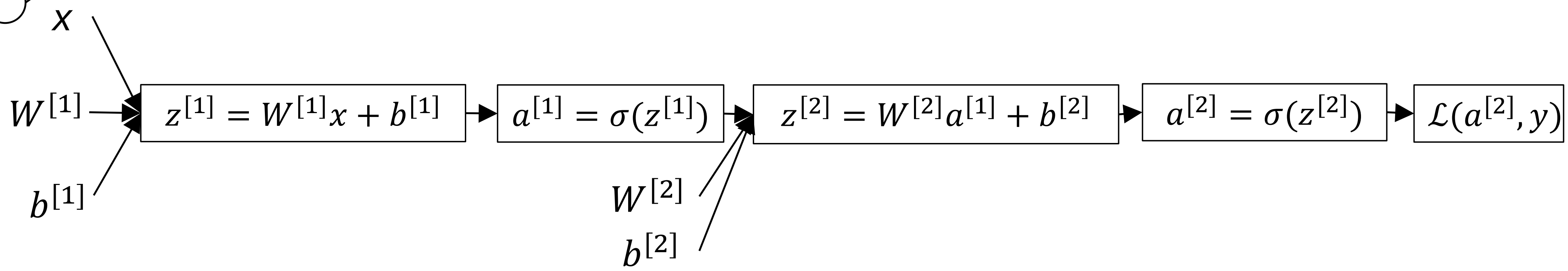
Neural Networks Overview

What is a Neural Network?

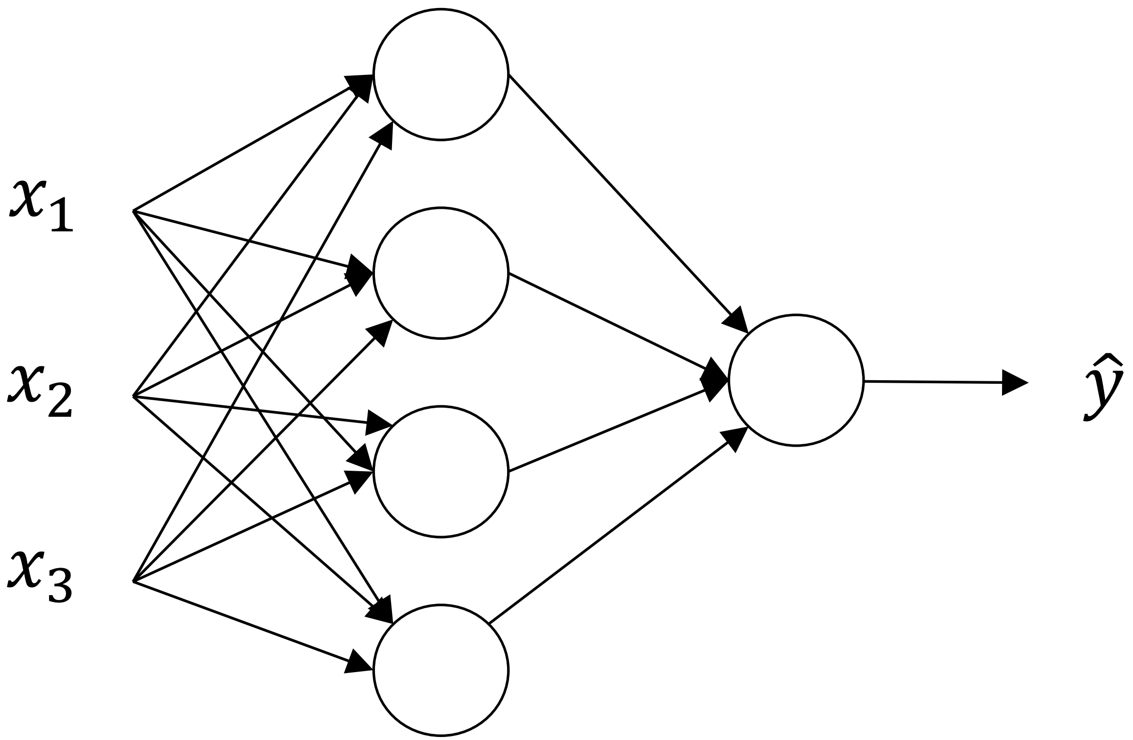




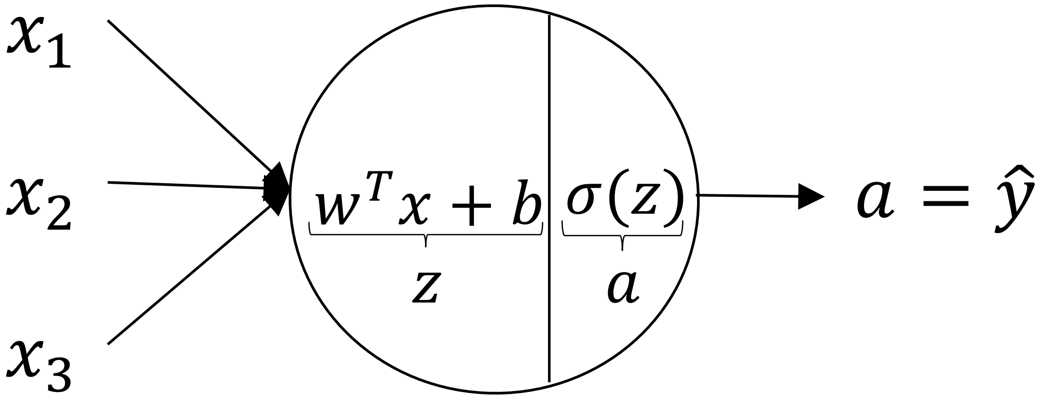


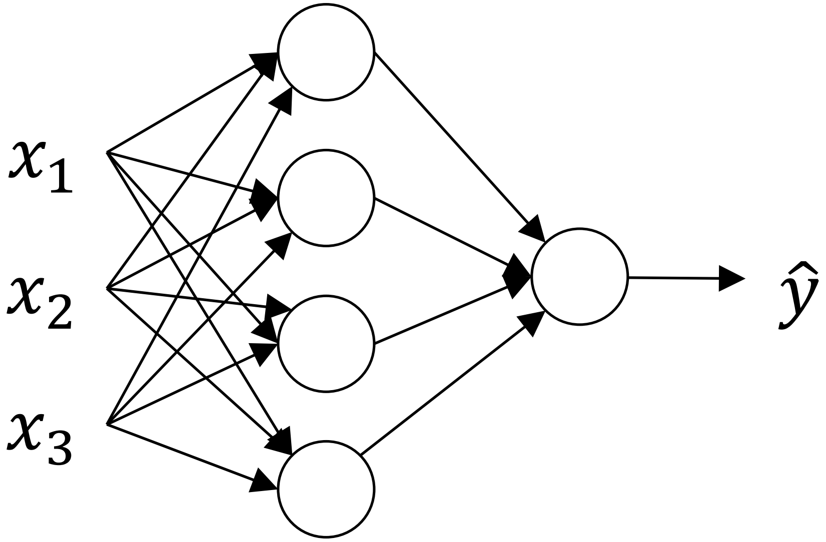


Neural Network Representation



Neural Network Representation

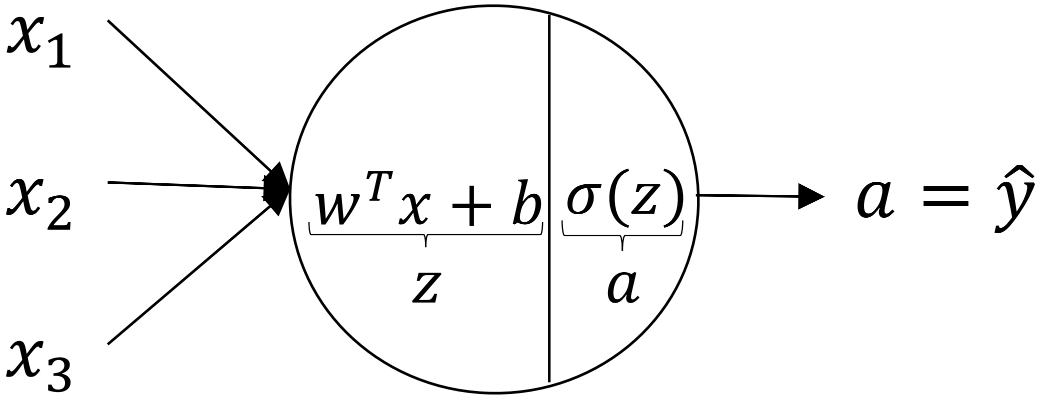


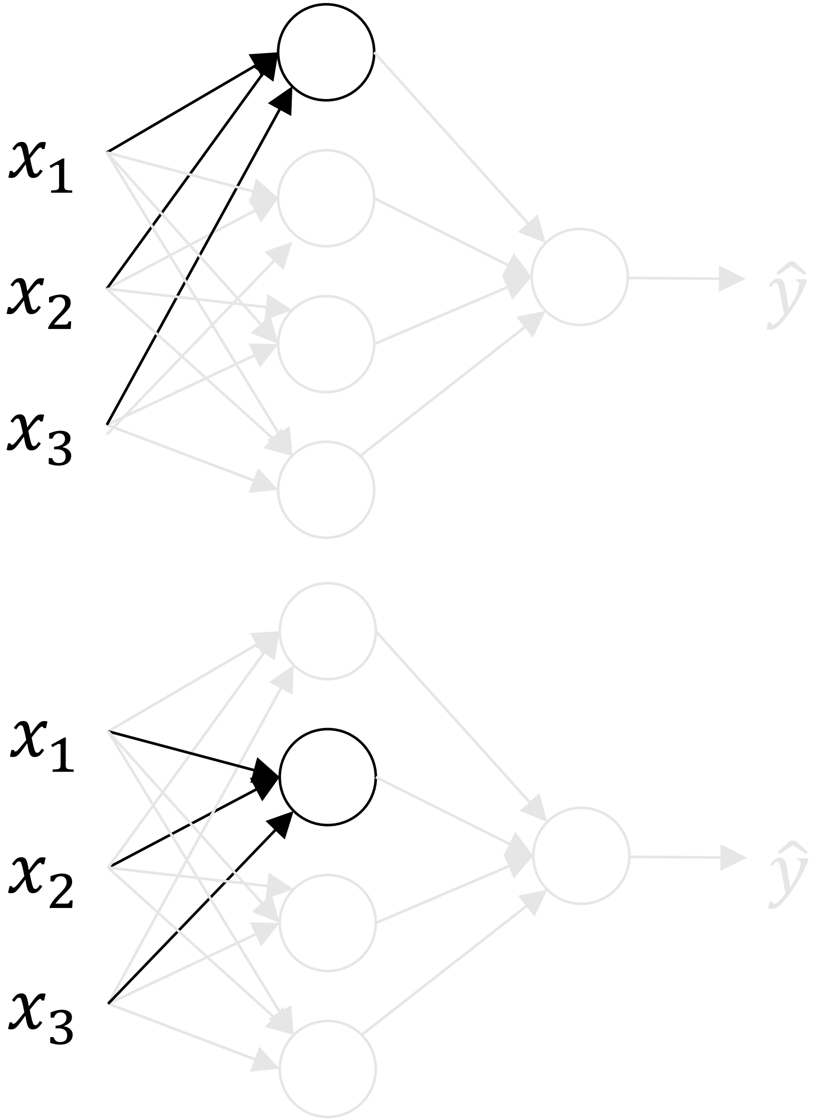


$$z = w^T x + b$$

$$a = \sigma(z)$$

Neural Network Representation

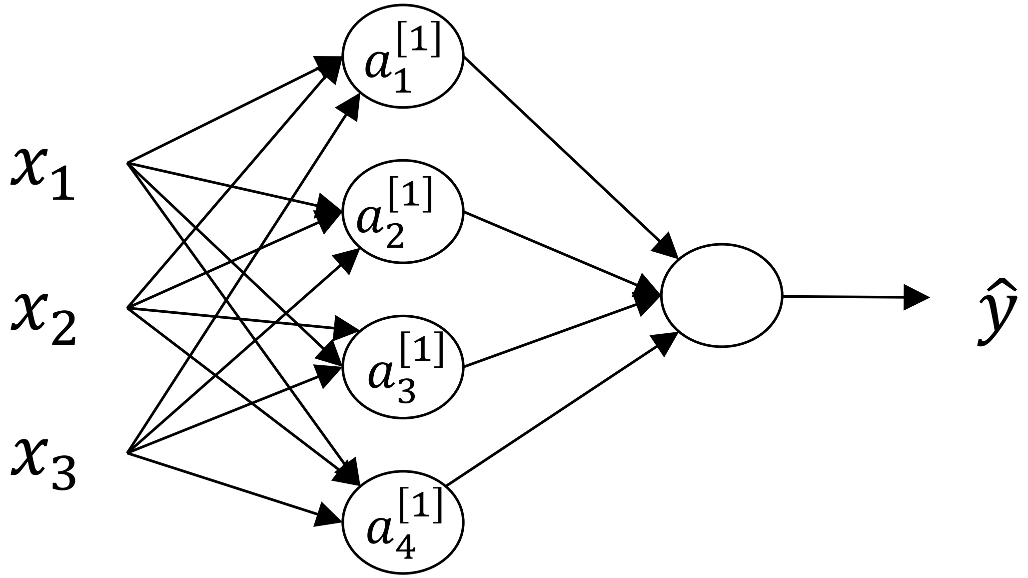




$z = w^T x + b$

$a = \sigma(z)$

Neural Network Representation



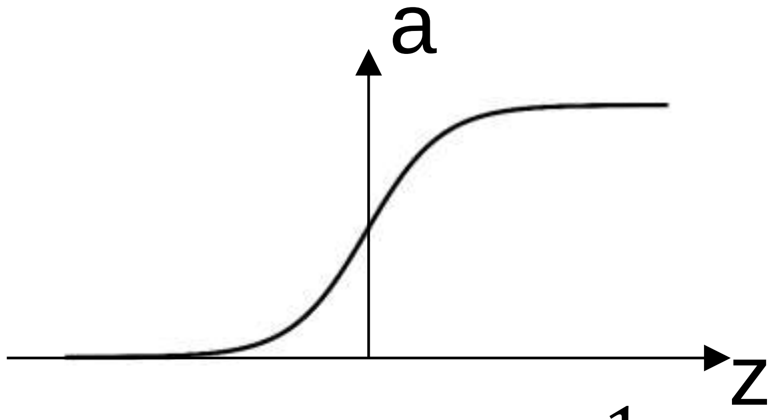
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]},\ a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]},\ a_2^{[1]} = \sigma(z_2^{[1]})$$

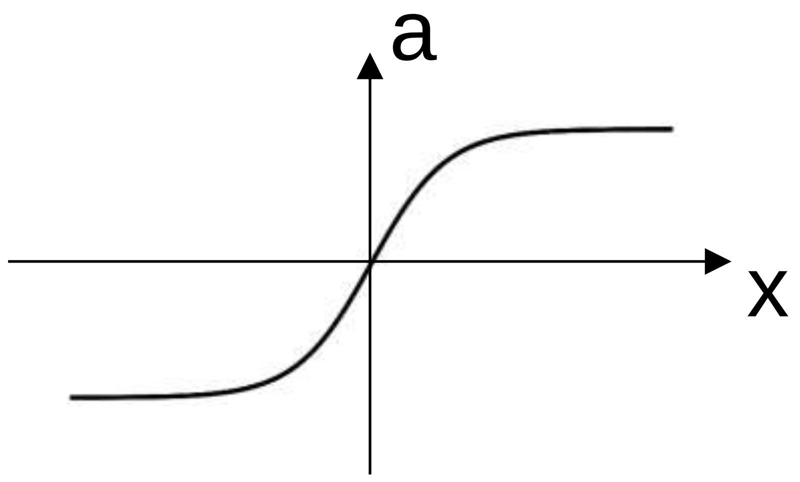
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]},\ a_3^{[1]} = \sigma(z_3^{[1]})$$

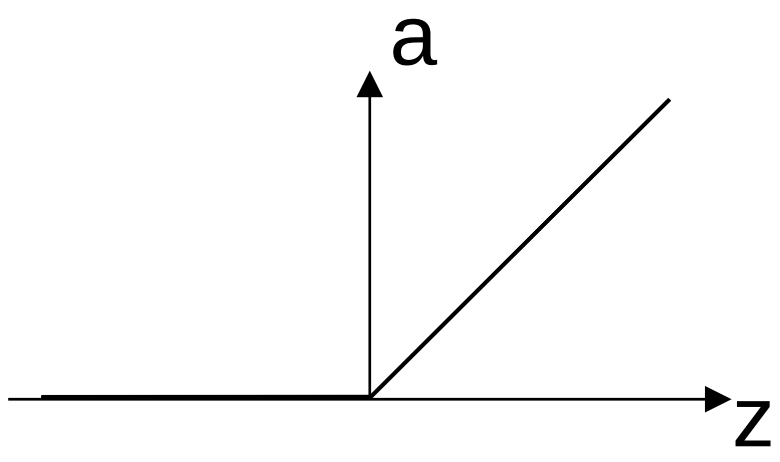
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]},\ a_4^{[1]} = \sigma(z_4^{[1]})$$

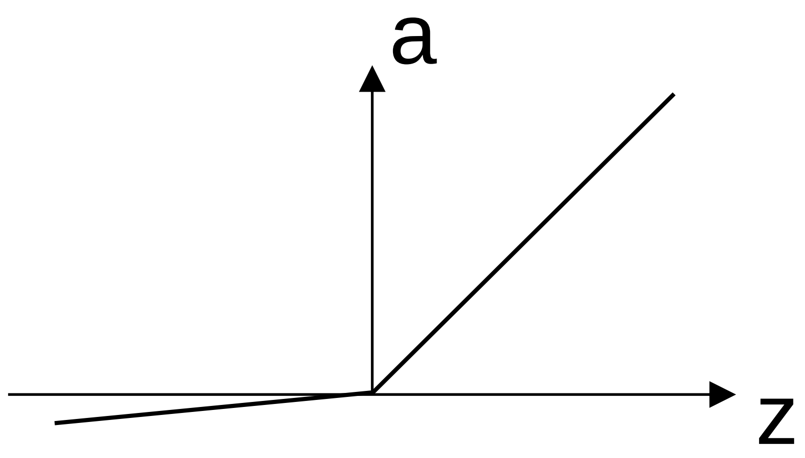
Activation functions



sigmoid: $a = \dfrac{1}{1 + e^{-z}}$





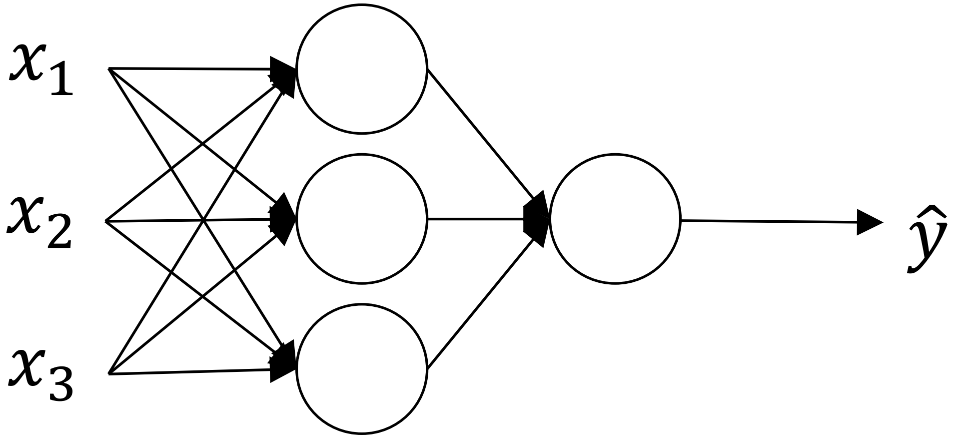


deeplearning.ai

One hidden layer Neural Network

Why do you need non-linear activation functions?

Activation function



Given x:

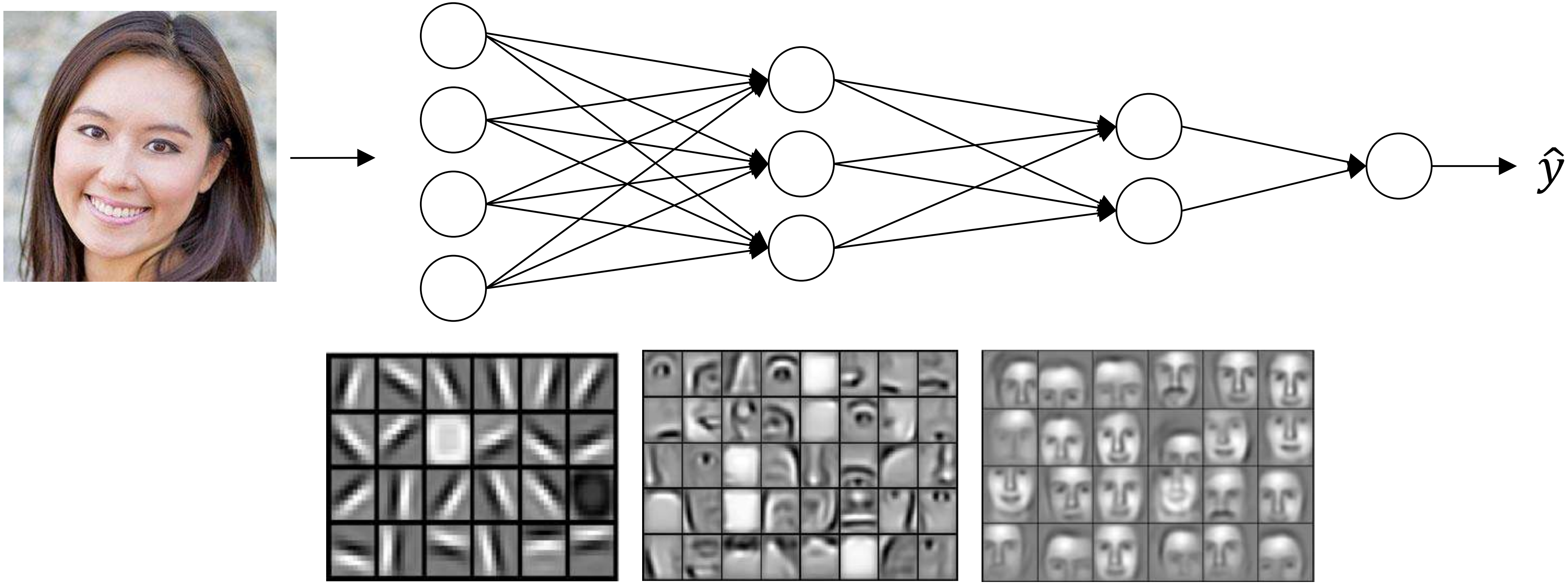
$$z^{[1]} = W^{[1]}x + b^{[1]}$$

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

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

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

Intuition about deep representation

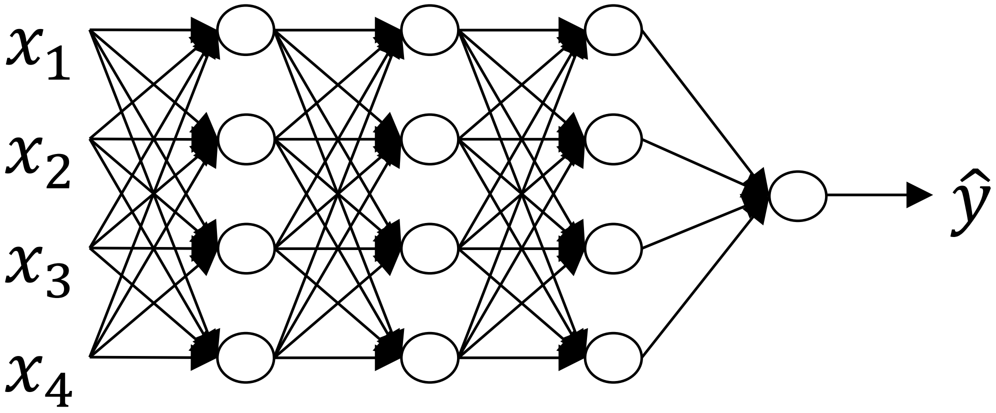


deeplearning.ai

Deep Neural Networks

Building blocks of deep neural networks

Forward and backward functions



Forward and backward functions

Forward propagation for layer *l*

Input $a^{[l-1]}$

Output $a^{[l]}$, cache $(z^{[l]})$

Backward propagation for layer *l*

Input $da^{[l]}$

Output $da^{[l-1]}, dW^{[l]}, db^{[l]}$

Summary

What are hyperparameters?

Parameters: $W^{[1]}$, $b^{[1]}$,$W^{[2]}$,$b^{[2]}$,$W^{[3]}$,$b^{[3]}$ ...

deeplearning.ai

Setting up your ML application

Train/dev/test sets

Train/dev/test sets

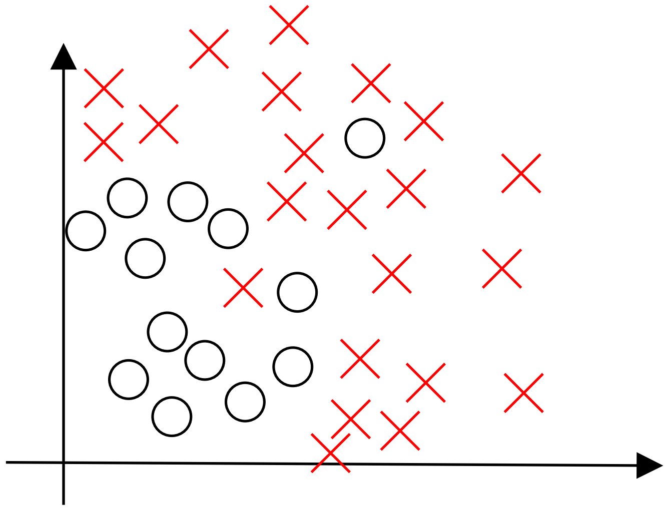
Mismatched train/test distribution

Training set:
Cat pictures from webpages

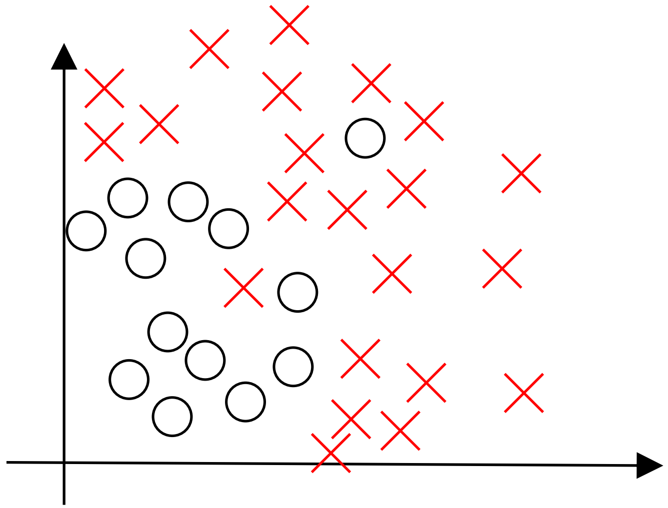
Dev/test sets:
Cat pictures from users using your app

Not having a test set might be okay. (Only dev set.)

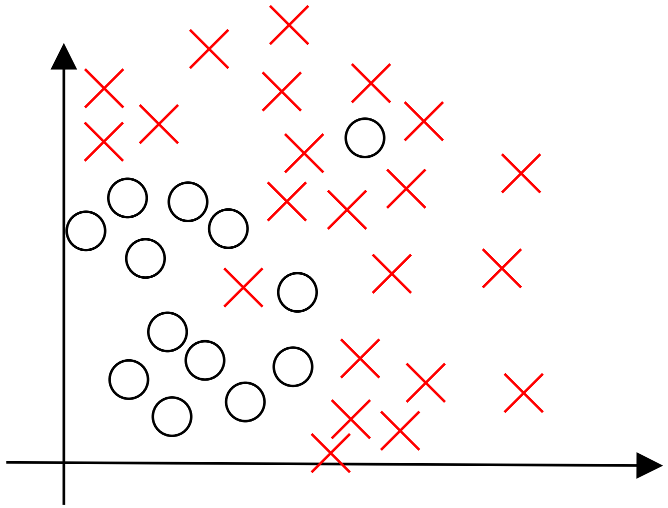
Bias and Variance



high bias



“just right”



high variance

Bias and Variance

Cat classification

Train set error:

Dev set error:

Regularization techniques

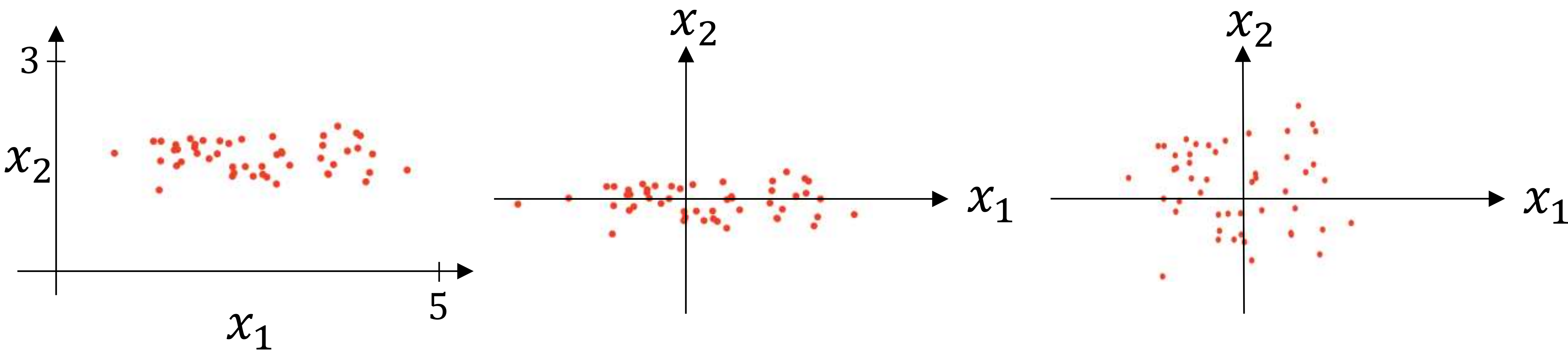
- L1, L2
- Dropout
- Data Augmentation
- Early Stopping

deeplearning.ai

Setting up your optimization problem

Normalizing inputs

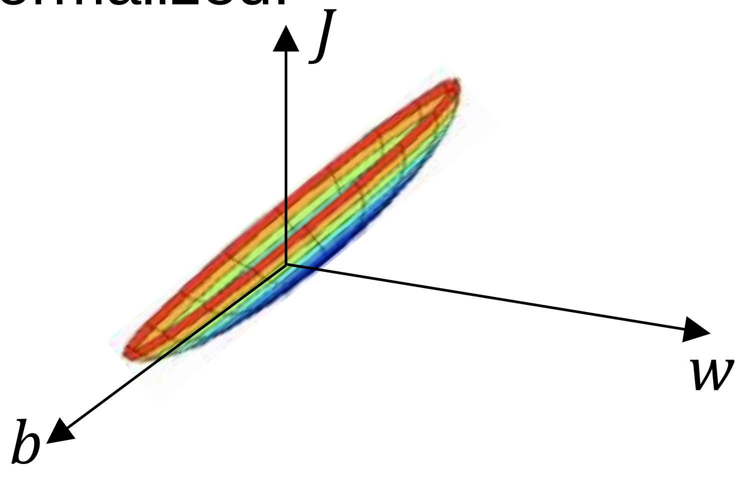
Normalizing training sets



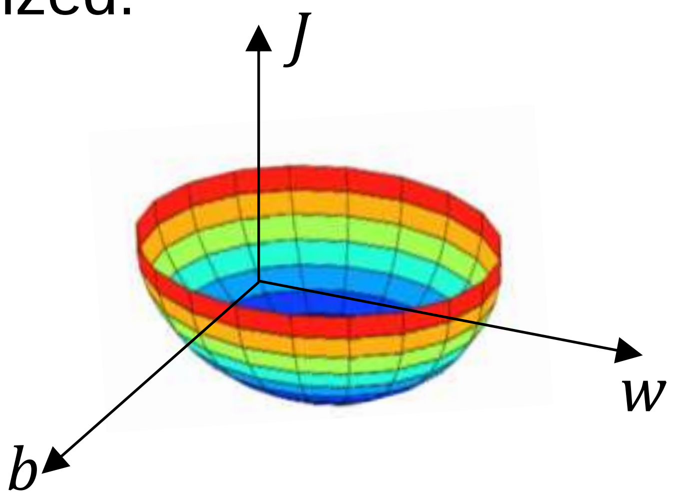
Why normalize inputs?

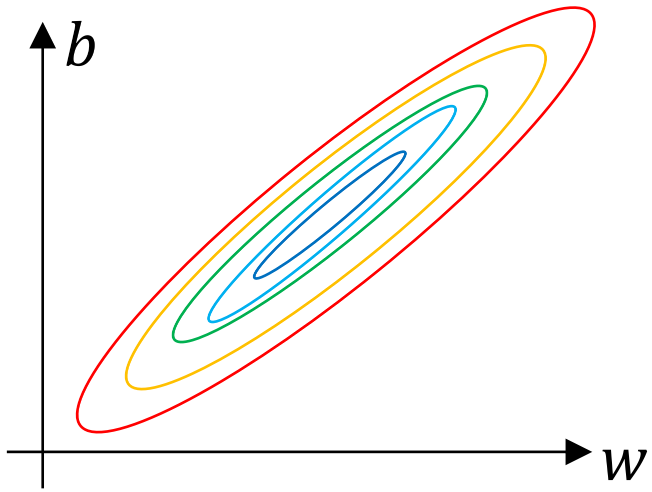
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$

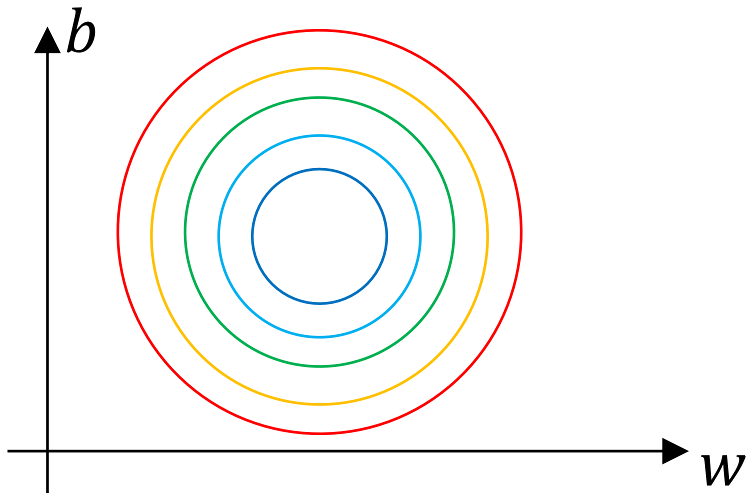
Unnormalized:



Normalized:







Optimization algorithms

- Gradient descent
- Mini-batch gradient descent
- Stochastic gradient descent
- RMSPROP
- ADAGRAD
- ADAM