Computer Vision

CVI620

Session 20
03/2025

What is Left?

9 sessions

1. Optimization and Loss Function
2. Code + Logistic Regression
3. ML and Images
4. Perceptron and Neural Networks
5. Deep Neural Networks
6. Convolution Neural Networks (CNN)
7. Advanced CNNs
8. Project
9. Segmentation
10. Introduction to object detection and image generation methods with AI
11. Project

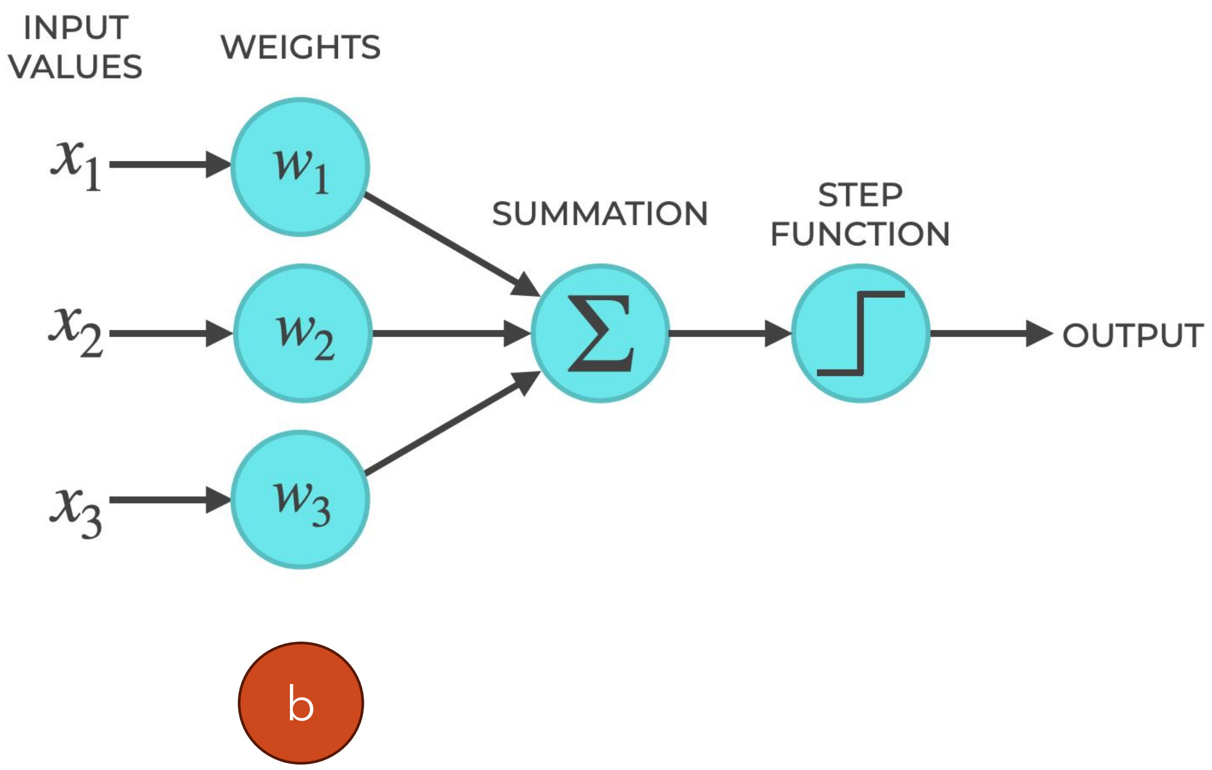
Agenda

01 Perceptron FW and BW

02 Softmax and Cross Entropy Loss

02 Implementation

Perceptron Algorithm

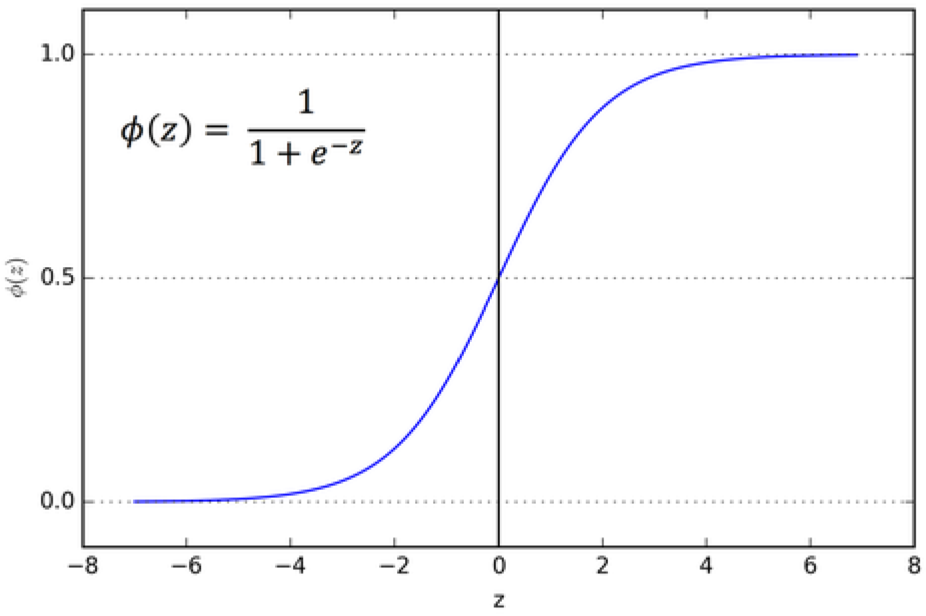


$f(x_0w_0 + x_1w_1 + x_2w_2 + x_3w_3)$

$f(\sum w_i\, x_i)$

$f(net)$

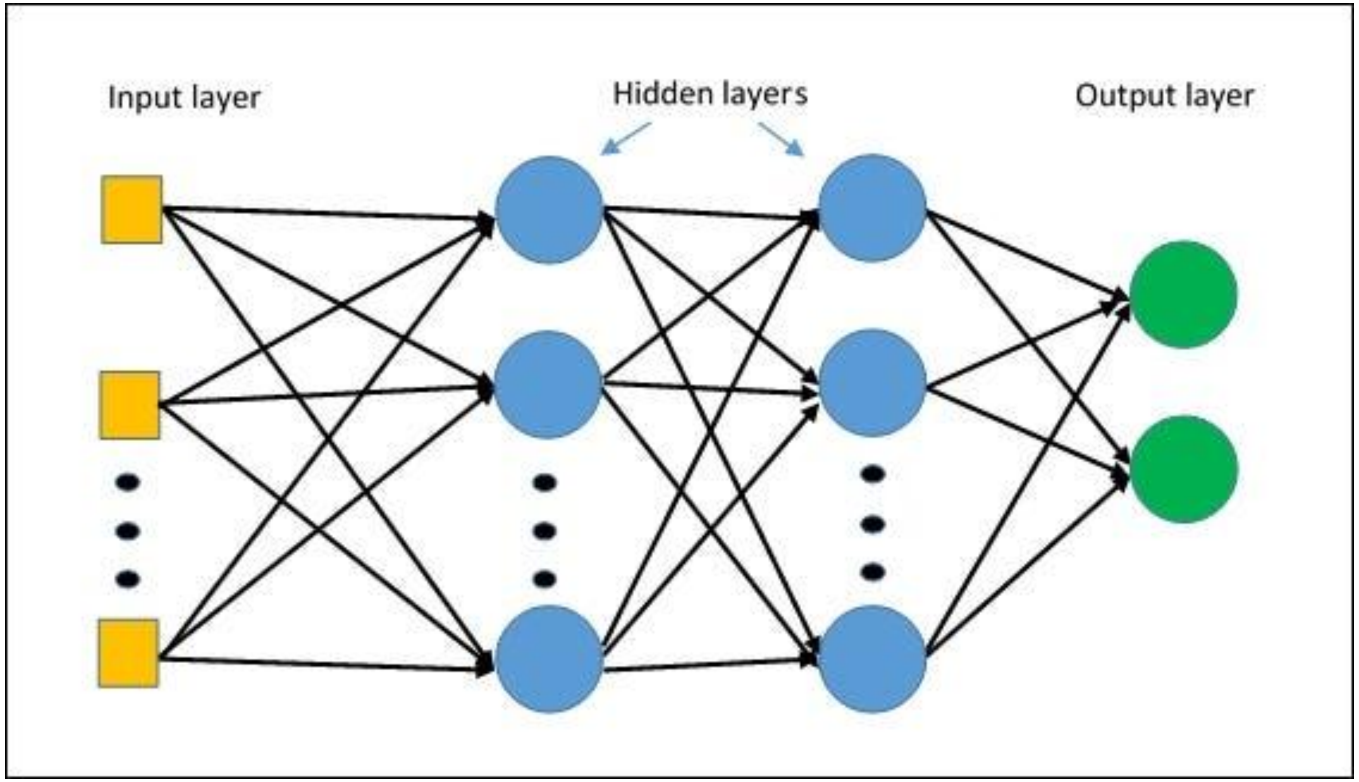
Sigmoid



Pros: derivative at all points

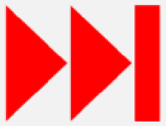
Cons: small derivatives at end points

Multi Layer Perceptron



2 main steps

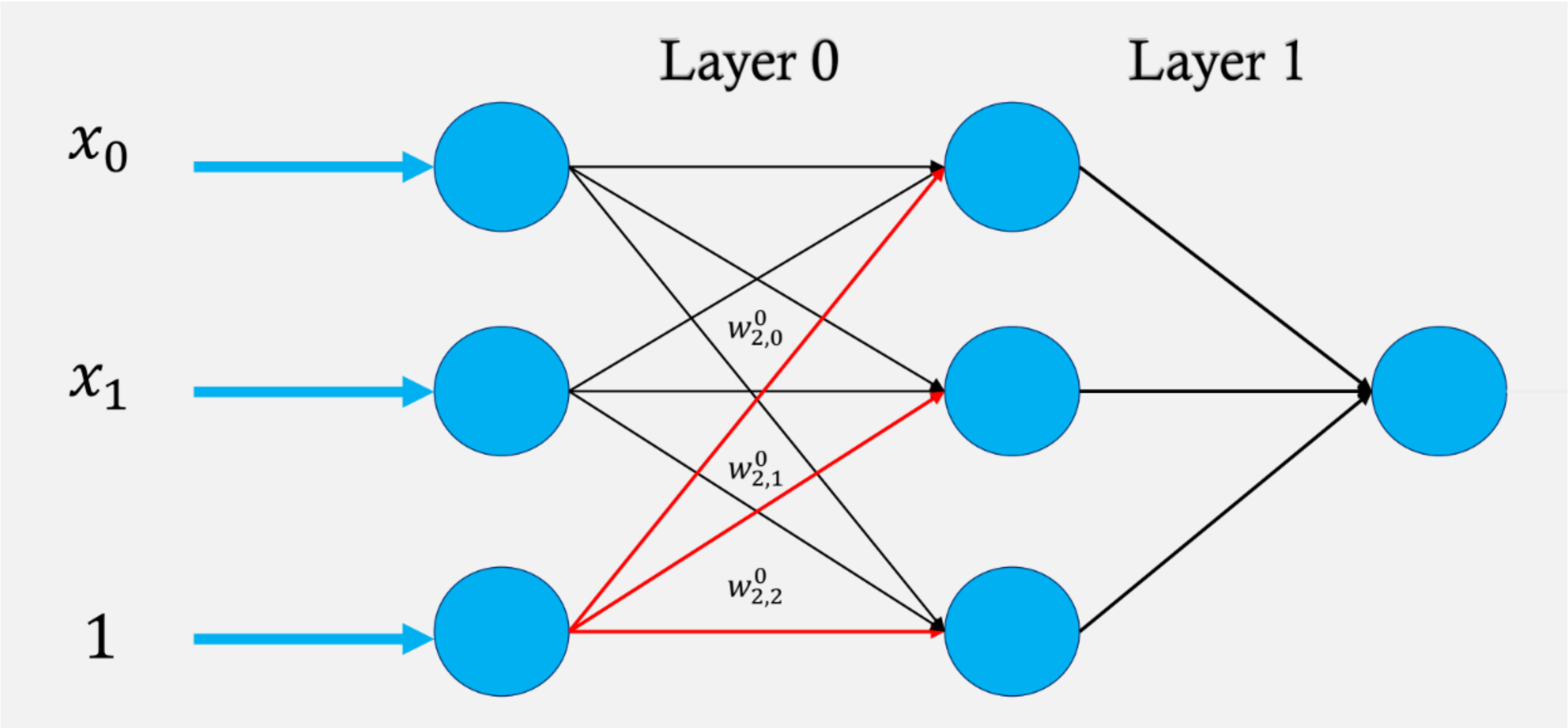
Forward Pass



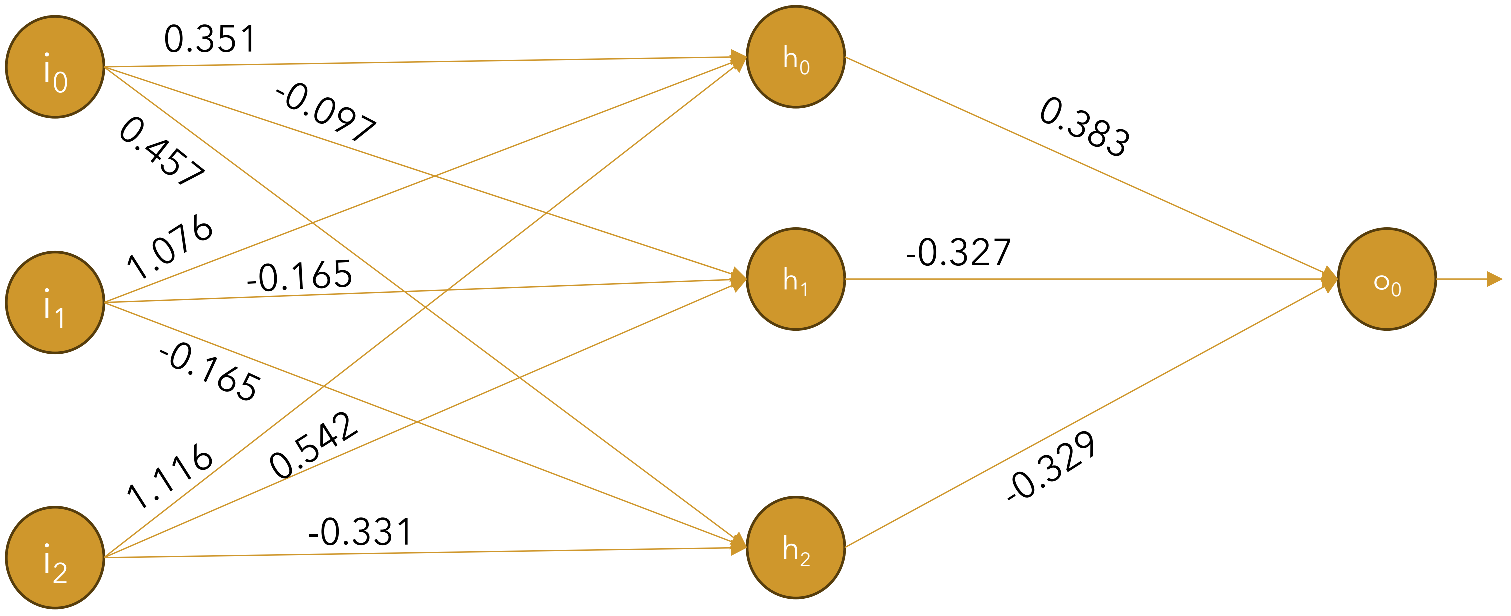
Backward Pass

Forward Pass

Forward Pass

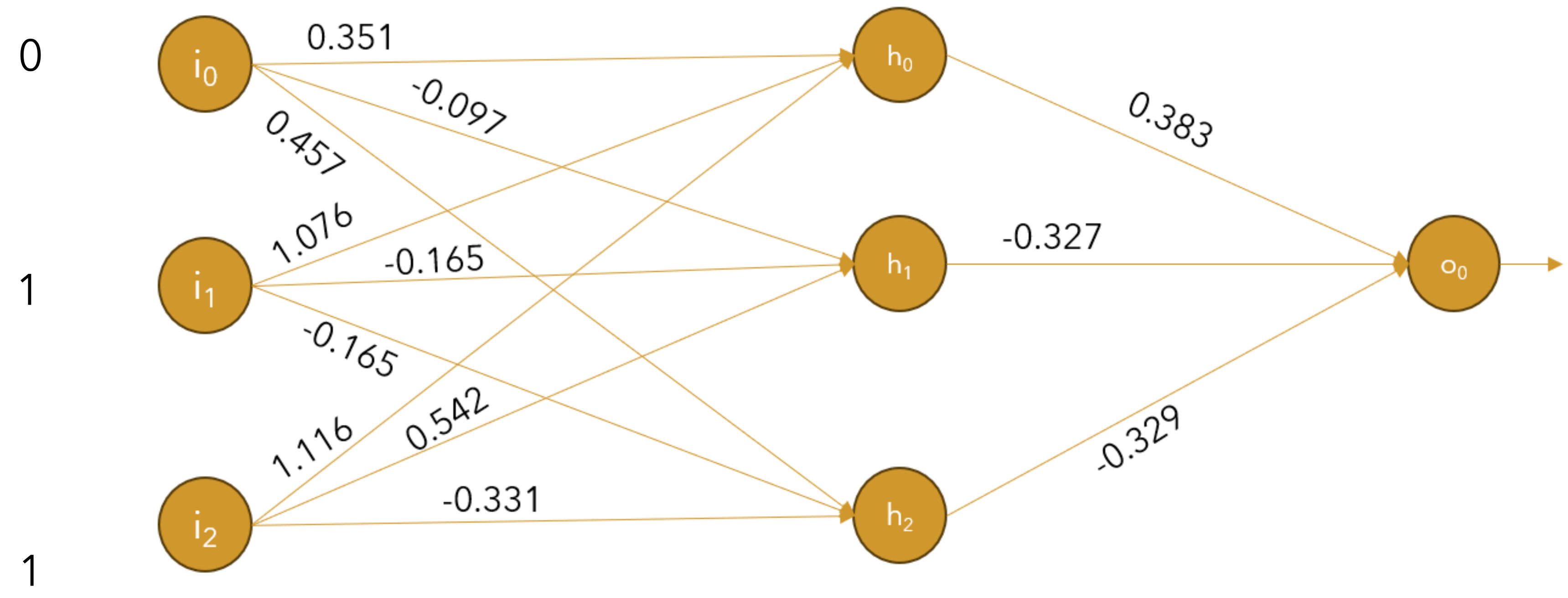


Example



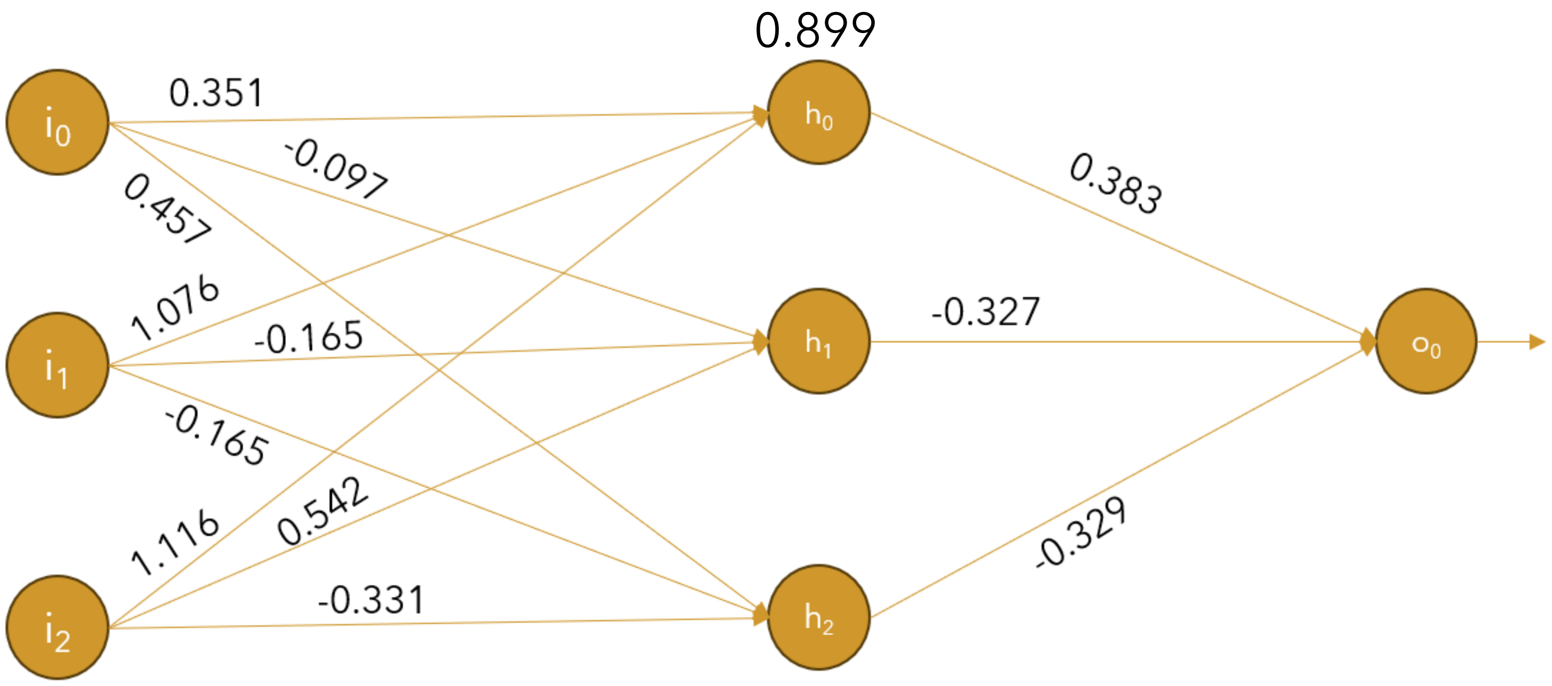
i0
i1
i2
h0
h1
h2
o0
0.351
-0.097
0.457
1.076
-0.165
-0.165
1.116
0.542
-0.331
0.383
-0.327
-0.329

Example



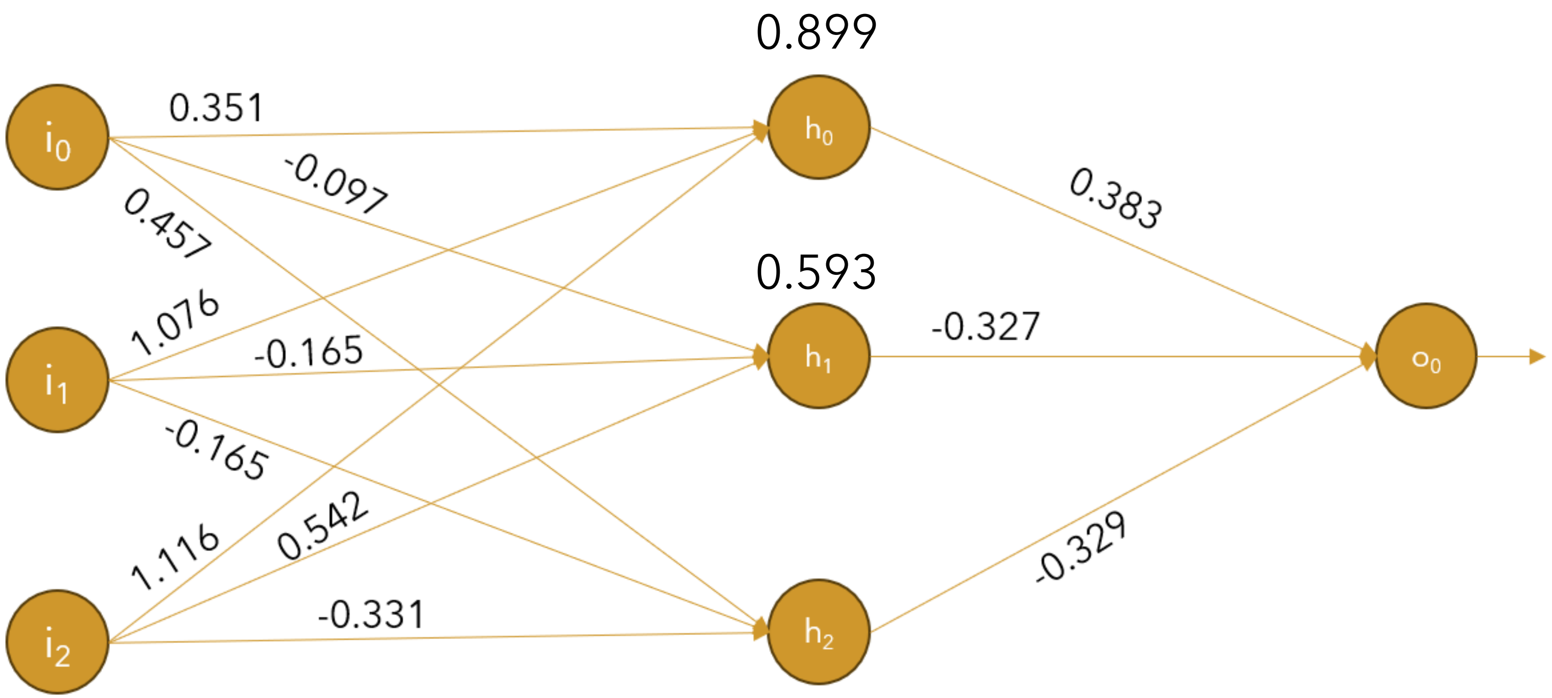
$$h_0 = 0(0.351) + 1(1.076) + 1(1.116) = 2.192$$

Example



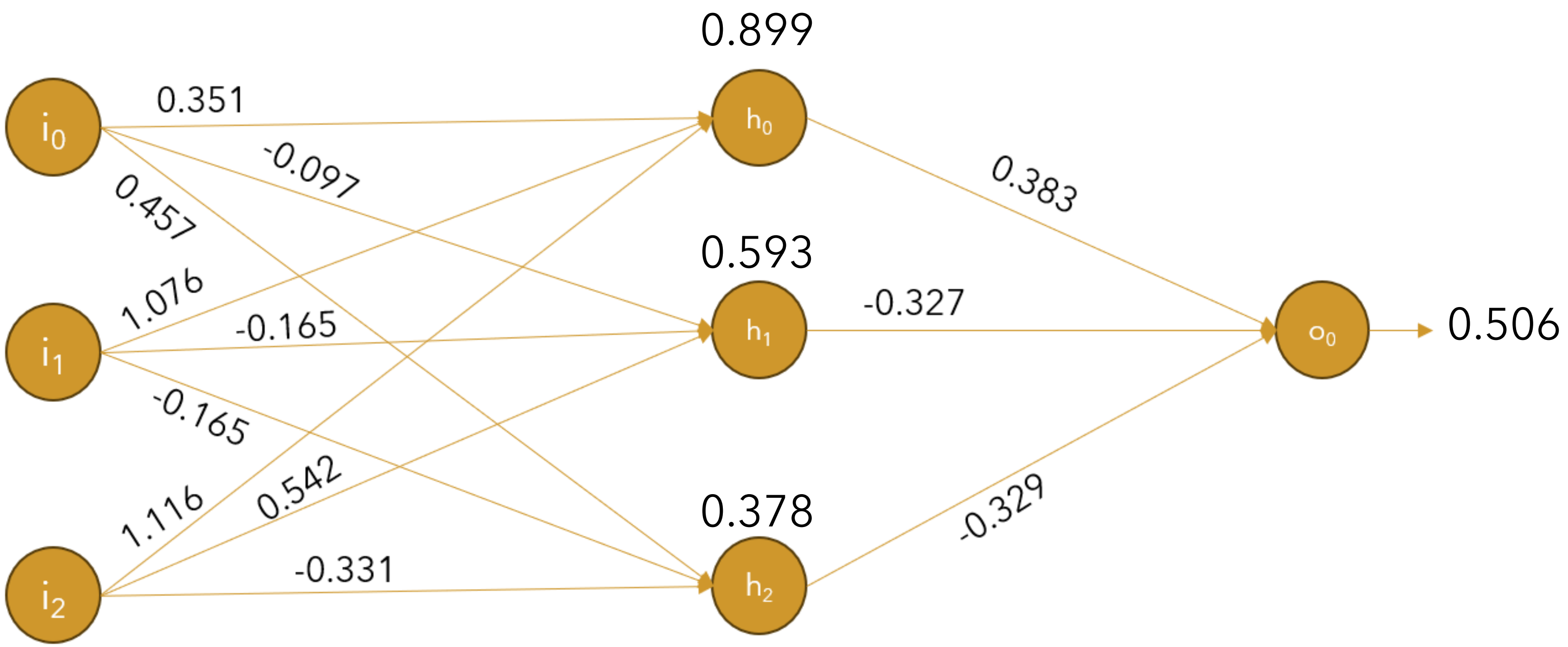
$$\frac{1}{1 + e^{-2.192}} = 0.899$$

Exmaple



0.899
0.593
0.351
-0.097
0.457
1.076
-0.165
-0.165
1.116
0.542
-0.331
0.383
-0.327
-0.329
i_0
i_1
i_2
h_0
h_1
h_2
o_0

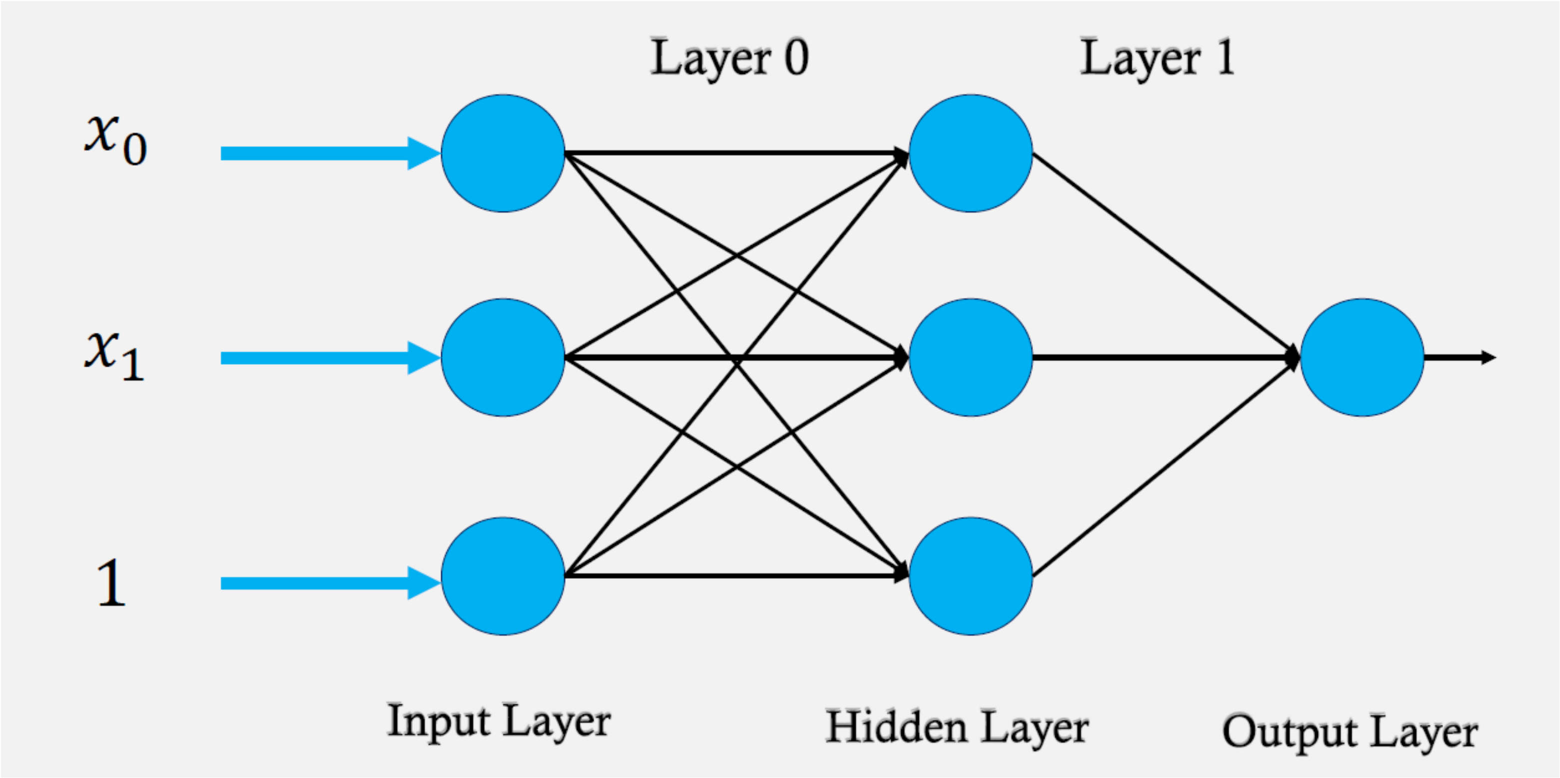
Exmaple



0.899
0.593
0.378
i_0
i_1
i_2
h_0
h_1
h_2
o_0
0.351
-0.097
0.457
1.076
-0.165
-0.165
1.116
0.542
-0.331
0.383
-0.327
-0.329
0.506

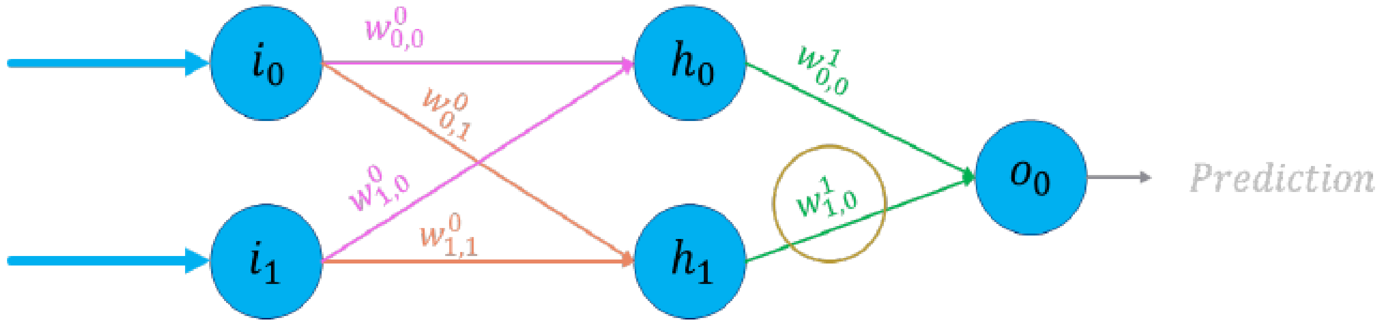
Backward Pass

Backpropagation

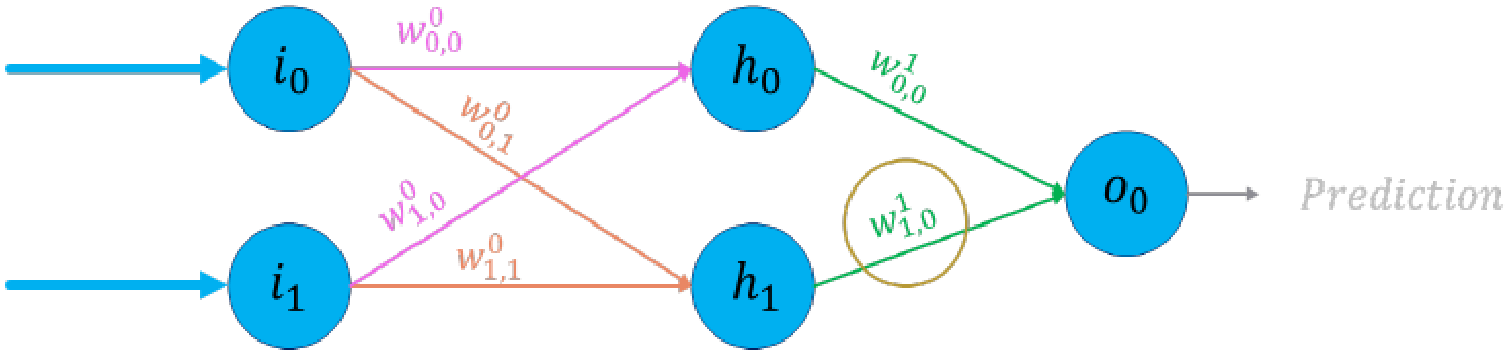


Simpler Backpropagation

For simplicity let's only consider summation and multiplications (no activation or bias)



Backpropagation



$$\text{Prediction} = (x_0 w^0_{0,0} + x_0 w^0_{1,0})\, w^1_{0,0} + (x_0 w^0_{0,1} + x_0 w^0_{1,1})\, w^1_{1,0}$$

$$\text{Loss} = \frac{(prediction - actual)^2}{2}$$

Goal

Final goal is to achieve to the best value for parameters

What are the parameters?

$$E = \frac{1}{n}\sum_{i=0}^{n}(y_i - (mx_i + c))^2$$

$$D_m = \frac{1}{n}\sum_{i=0}^{n}2(y_i - (mx_i + c))(-x_i)$$

$$D_m = \frac{-2}{n}\sum_{i=0}^{n}x_i(y_i - \bar{y}_i)$$

$$m = m - \alpha \times D_m$$

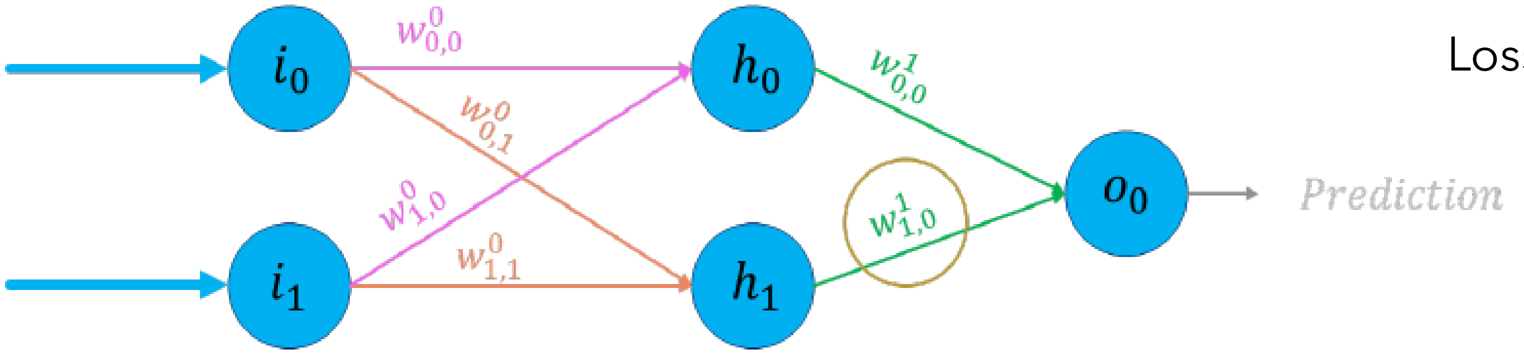
$$D_c = \frac{-2}{n}\sum_{i=0}^{n}(y_i - \bar{y}_i)$$

$$c = c - \alpha \times D_c$$

Backpropagation

$$\text{Prediction} = (x_0 w^0_{0,0} + x_0 w^0_{1,0})\, w^1_{0,0} + (x_0 w^0_{0,1} + x_0 w^0_{1,1})\, w^1_{1,0}$$

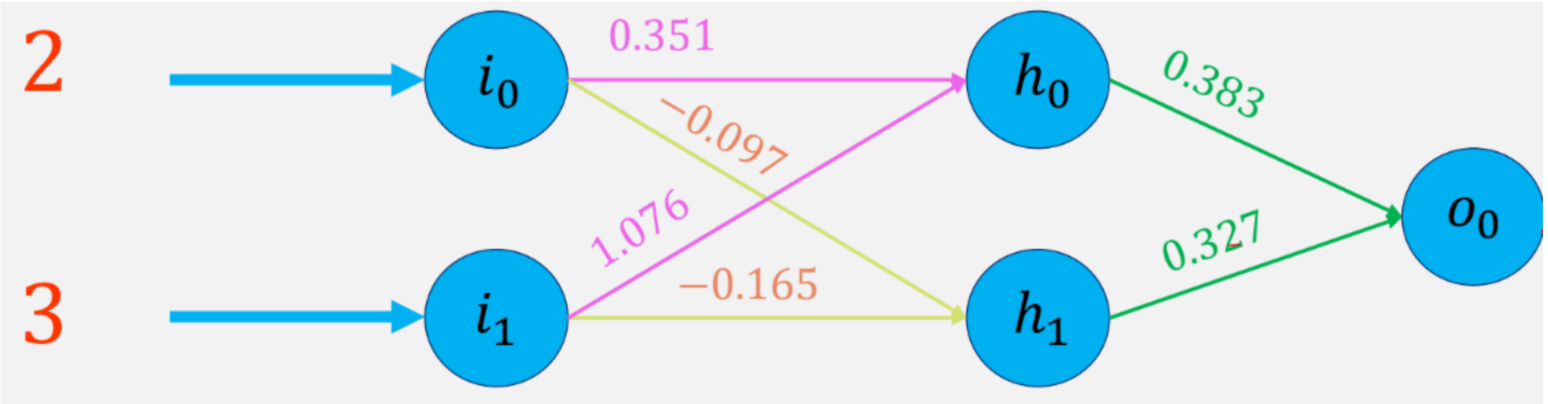
$$\text{Loss} = \frac{(prediction - actual)^2}{2}$$



$$\frac{\partial loss}{\partial w^1_{1,0}} = \frac{\partial loss}{\partial Prediction} \times \frac{\partial Prediction}{\partial w^1_{1,0}}$$

Example

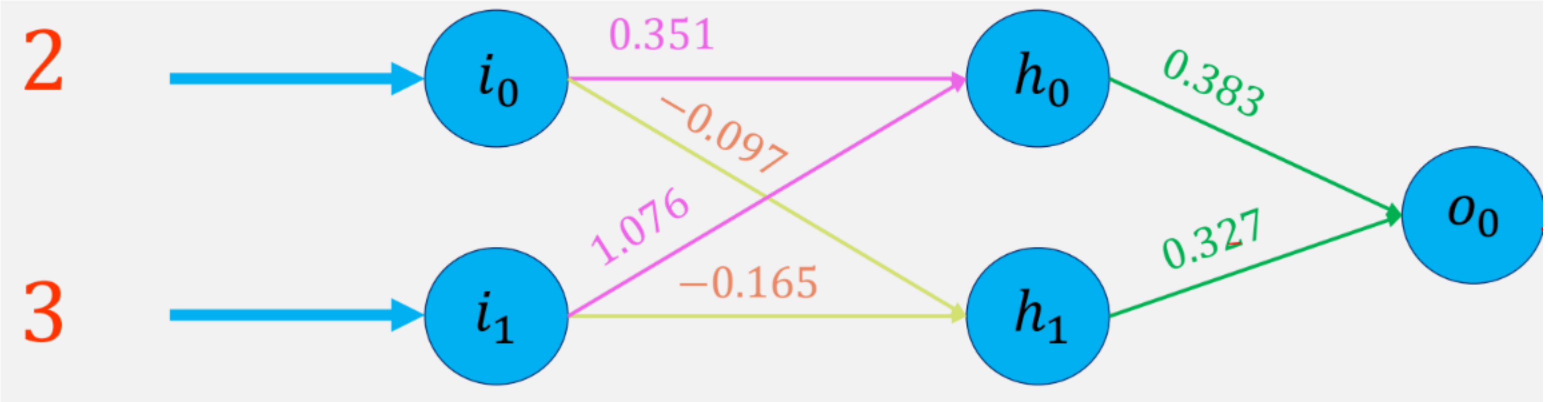
$$\frac{\partial loss}{\partial w_{1,0}^{1}} = \frac{\partial loss}{\partial Prediction} \times \frac{\partial Prediction}{\partial w_{1,0}^{1}}$$



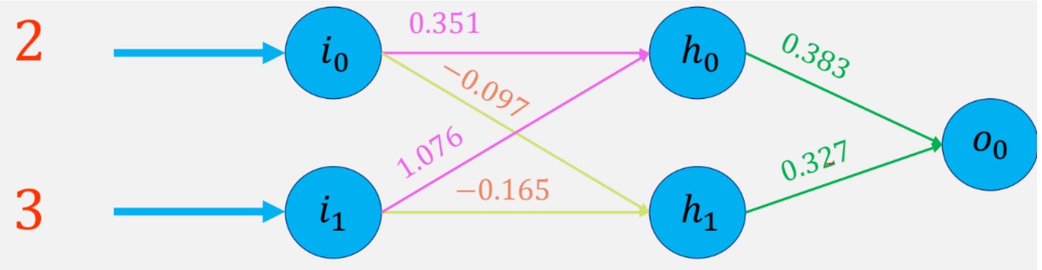
Actual y = 1

Example

$$\frac{\partial loss}{\partial w_{1,0}^{1}} = \frac{\partial loss}{\partial Prediction} \times \frac{\partial Prediction}{\partial w_{1,0}^{1}}$$



$$Prediction = (2 \times 0.351 + 3 \times 1.076)0.383 + (2 \times -0.097 + 3 \times -0.165)0.327 = 1.730$$



$$\frac{\partial loss}{\partial w^1_{1,0}} = \frac{\partial loss}{\partial Prediction} \times \frac{\partial Prediction}{\partial w^1_{1,0}}$$

Prediction = $(x_0 w^0_{0,0} + x_0 w^0_{1,0})\, w^1_{0,0} + (x_0 w^0_{0,1} + x_0 w^0_{1,1})\, w^1_{1,0}$

$$\Delta = Prediction\ - actual = 1.730\ - 1 = 0.730$$

h_1 = -0.097*2 - 3*0.165 = -0.689

$$\frac{\partial Error}{\partial w^1_{1,0}} = \frac{\partial Error}{\partial Prediction} \times \frac{\partial Prediction}{\partial w^1_{1,0}} \quad \Rightarrow \Delta h_1$$

$$\frac{\partial Error}{\partial w^1_{1,0}} = (0.730) \times (-0.689) = \boxed{-0.502}$$

$$\frac{\partial Error}{\partial w_{1,0}^{1}} = \frac{\partial Error}{\partial Prediction} \times \frac{\partial Prediction}{\partial w_{1,0}^{1}} \quad \Rightarrow \Delta h_1$$

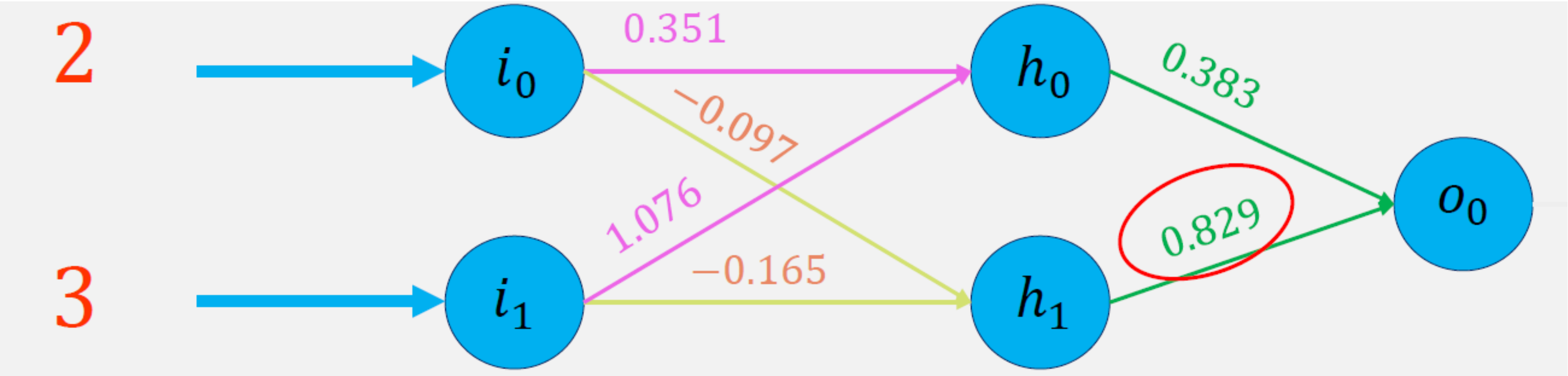
$$\frac{\partial Error}{\partial w_{1,0}^{1}} = (0.730) \times (-0.689) = \boxed{-0.502}$$

$$w^{+} = w^{-} - \alpha \frac{\partial L}{\partial w}$$

$$w_{1,0\,\mathrm{new}}^{1} = 0.327 - (-0.502) = 0.829$$

Updated Weight

$$Prediction = (2 \times 0.351 + 3 \times 1.076)0.383 + (2 \times -0.097 + 3 \times -0.165)0.327 = 1.730$$



$$prediction\ after\ update = (2 * 0.351 + 3 * 1.076)\ 0.383 + (2 * -0.097 + 3 * -0.165)\ 0.829 = 1.255$$

Frameworks for Neural Networks

Google- 2015

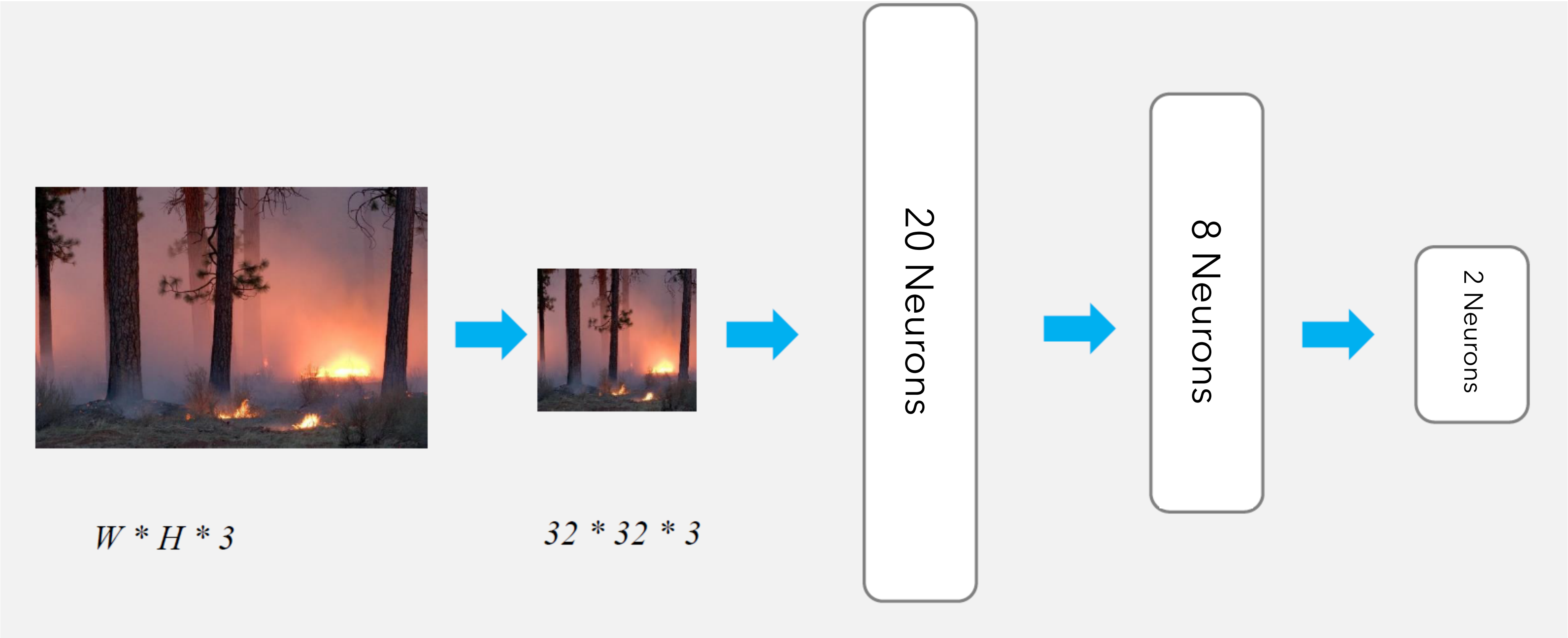
Facebook - 2016

Keras

- By Francois Chollet
- Easier code
- From tf v2

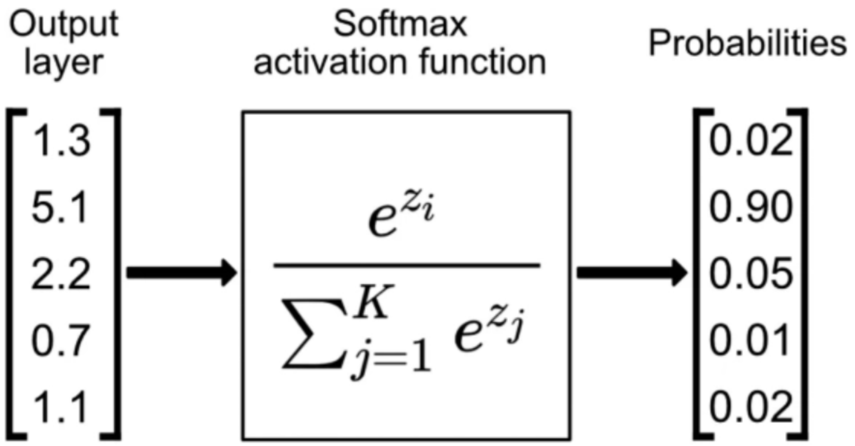
Fire detection with Neural Networks

Architecture

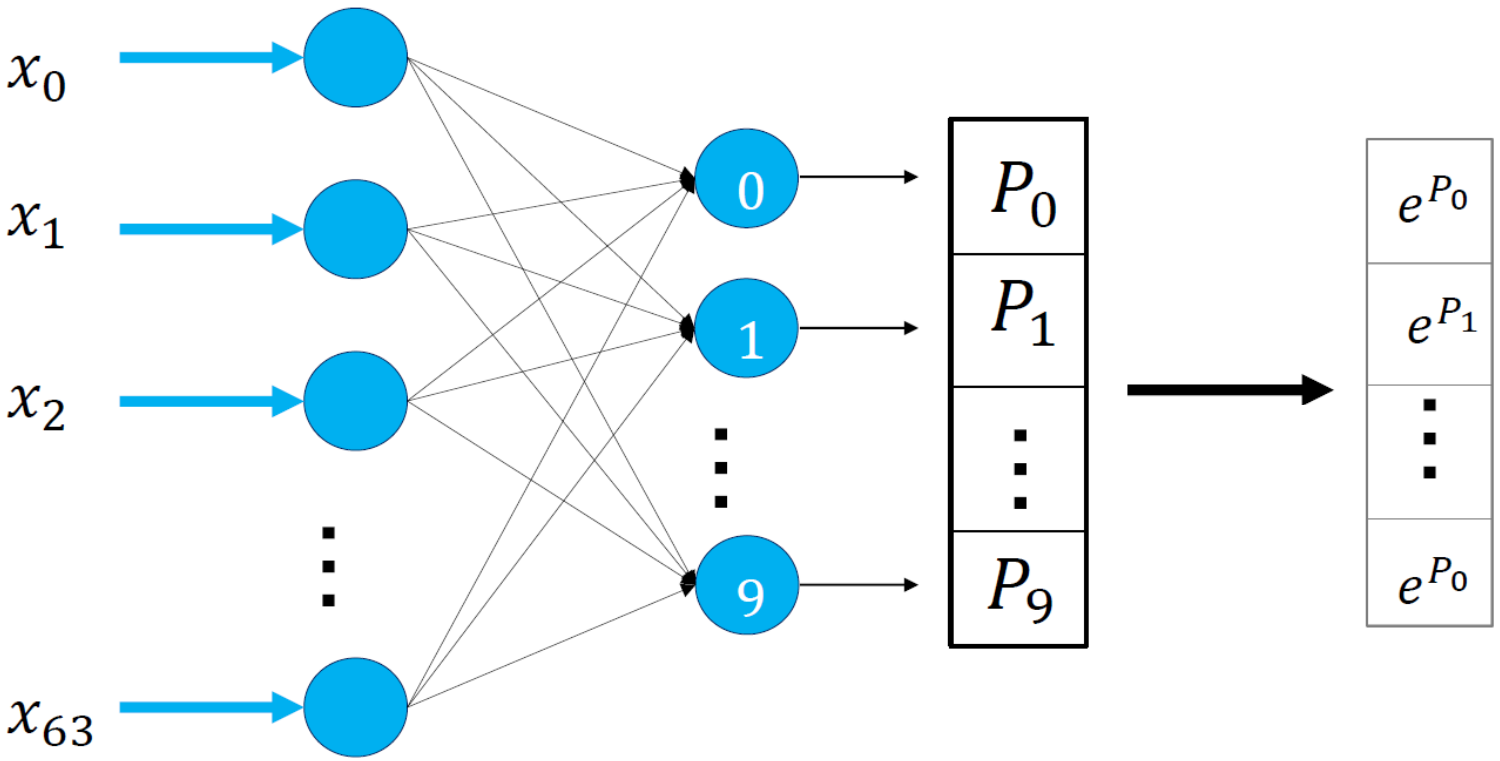


Softmax

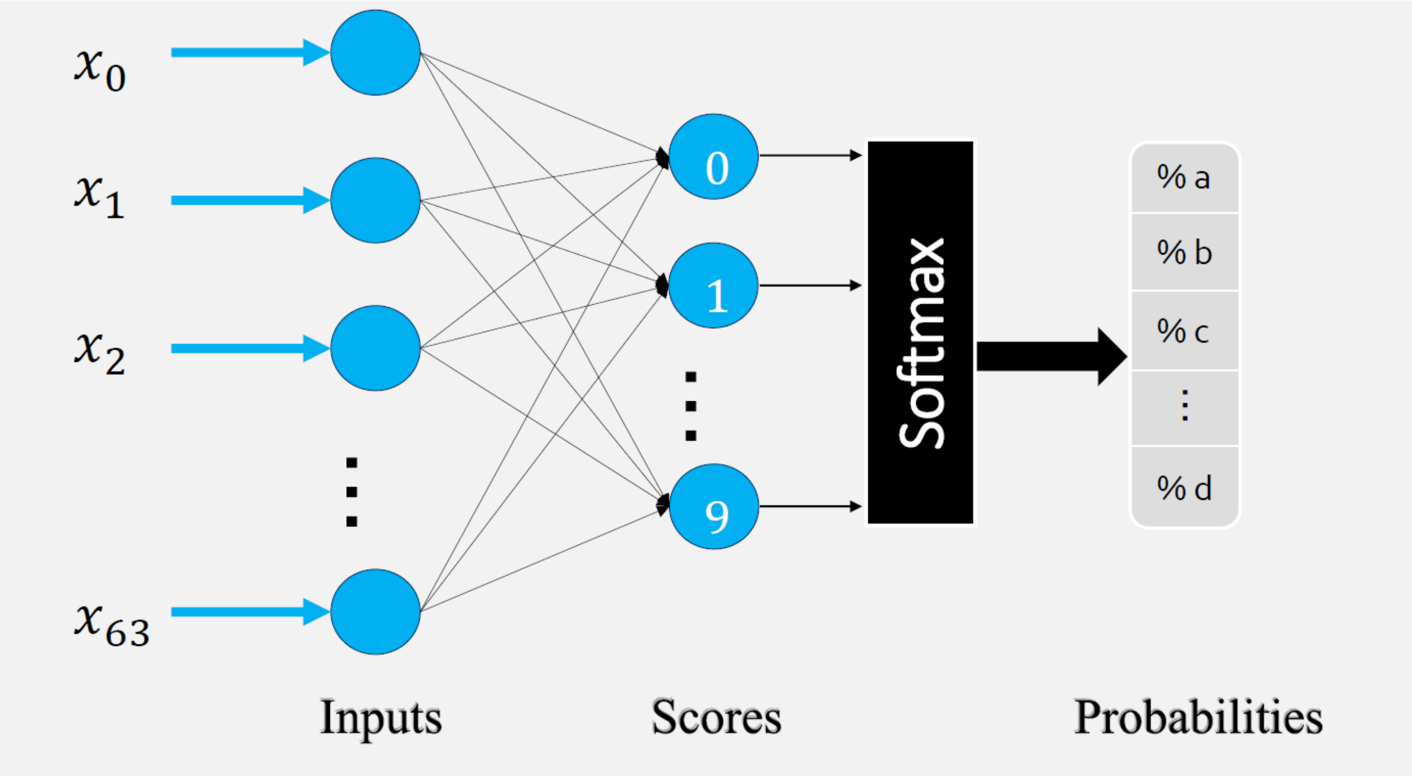
- An activation function
- Turns a vector of raw scores (logits) into probabilities that sum to 1.



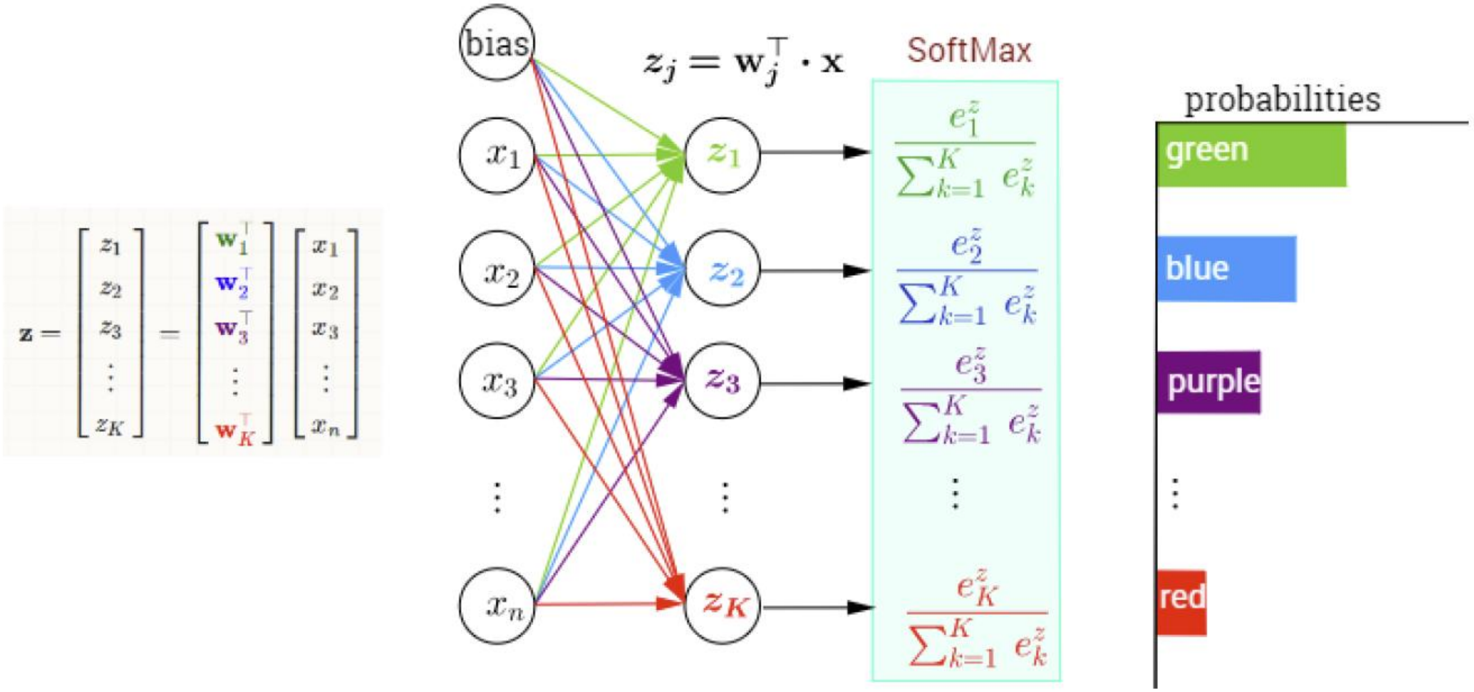
Softmax



Softmax in Neural Networks



Summary



bias
x_1
x_2
x_3
x_n
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
z_1
z_2
z_3
z_K
SoftMax
probabilities
green
blue
purple
red

Loss Function For Classification

Type of Encoding

Cat → 0

Dog → 1

Bird → 2

Horse → 3

Integer Encoding

```python
from sklearn import preprocessing

le = preprocessing.LabelEncoder()

label = ["cat", "dog", "pandas", "fire"]

out = le.fit_transform(label)

print(out)
```

```
[1 2 0 1]
```

Type of Encoding

Cat → [1 0 0 0]

Dog → [0 1 0 0]

Bird → [0 0 1 0]

Horse → [0 0 0 0]

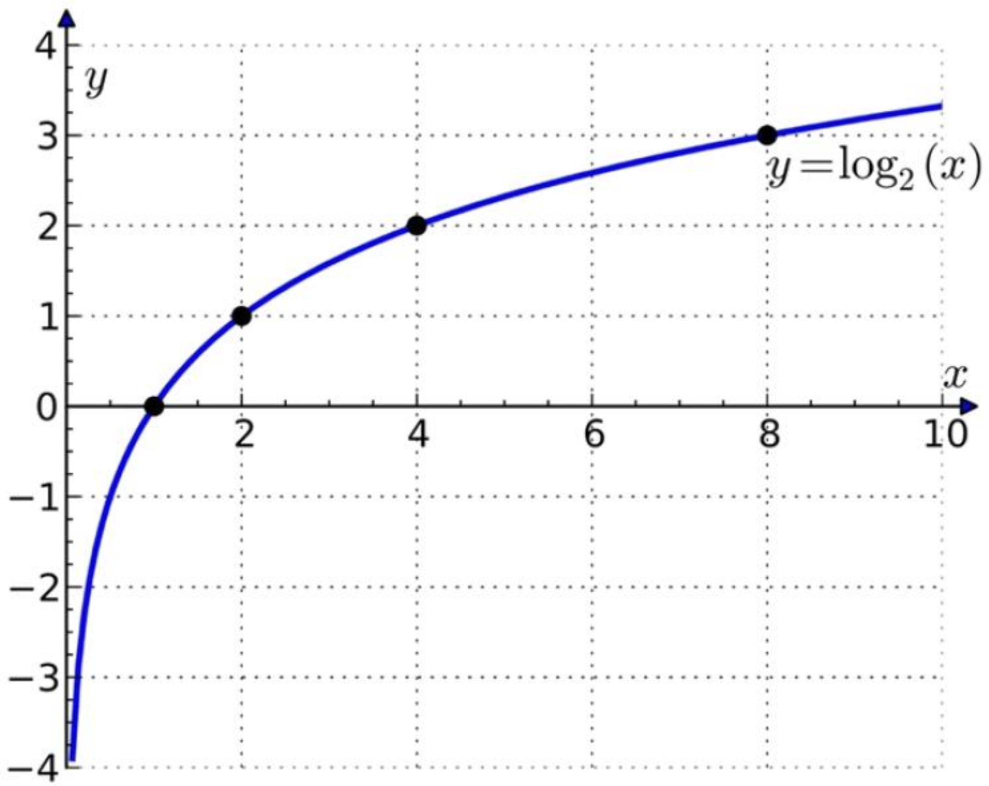
One Hot Encoding

```python
from tensorflow.keras.utils import to_categorical

labels = [1, 2, 0, 1]

encode = to_categorical(labels)

print(encode)
```

```
[[0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]]
```

Cross Entropy Loss

$$loss = -\sum_{i=1}^{n} y_i \log(y_i')$$



Cross Entropy Loss

Prediction	0.15	0.11	0.19	0.37	0.20	0.9	0.12	0.32	0.13	0.07
True Labels	0	0	0	0	0	1	0	0	0	0