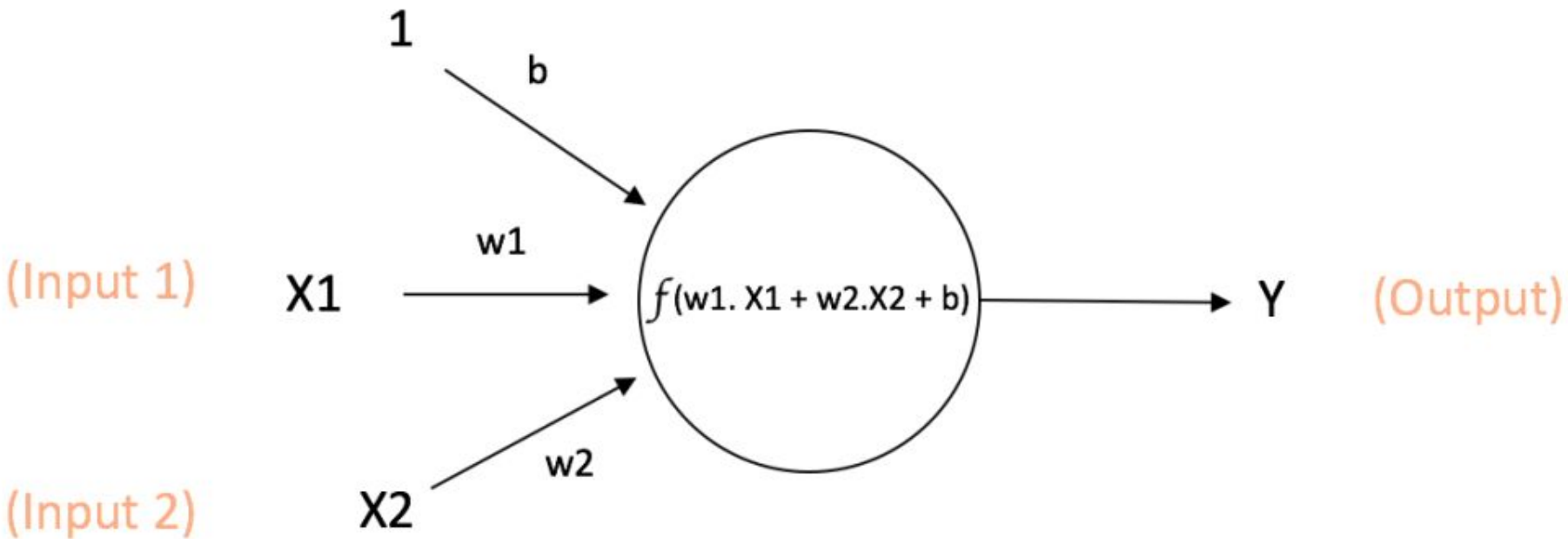
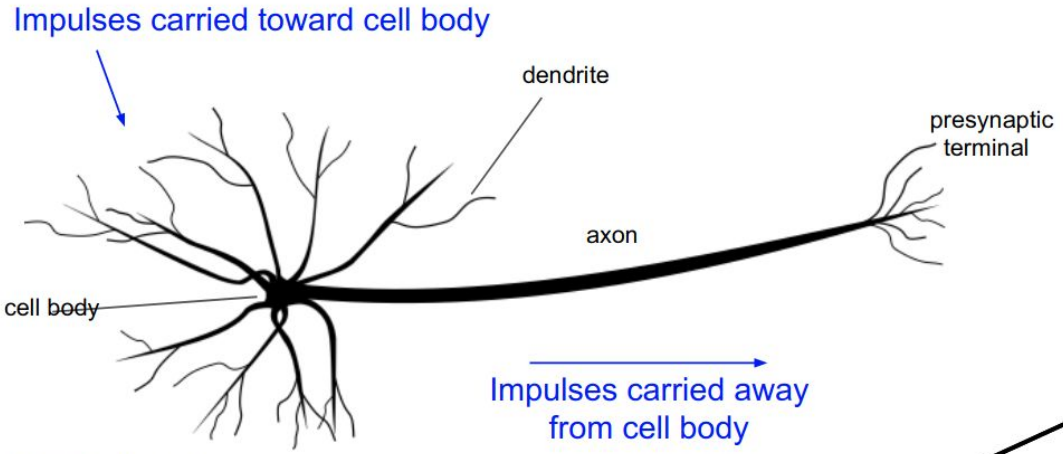
Neural Network

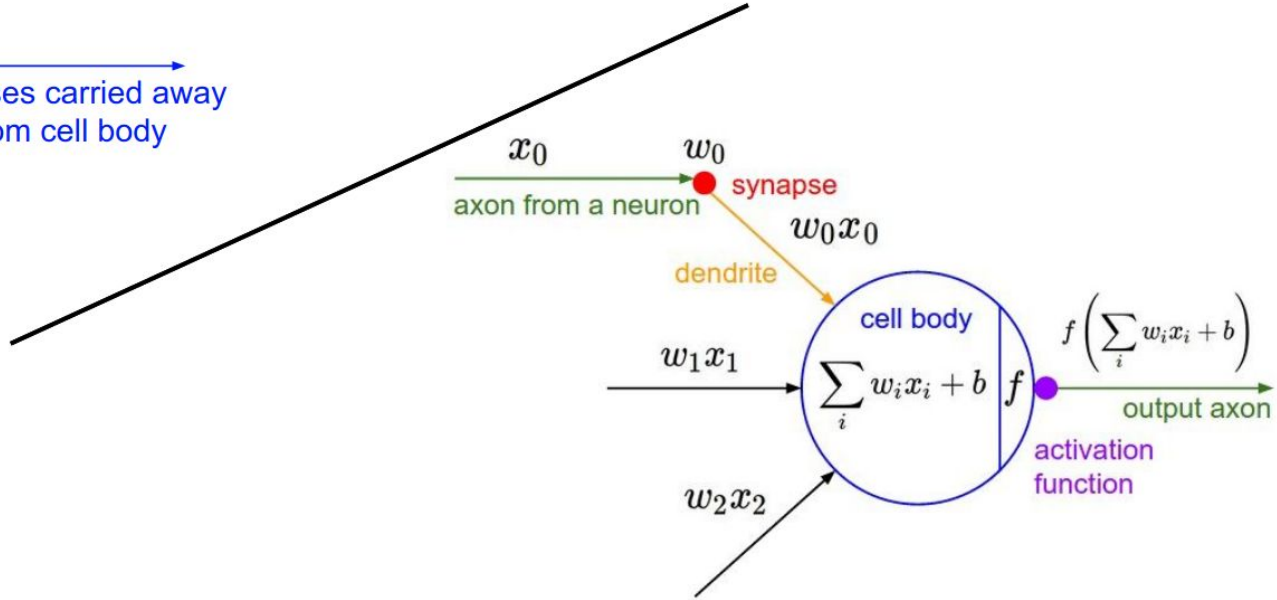
COGNIBOT

AI meets Industry



Output of neuron = Y= f(w1. X1 + w2.X2 + b)



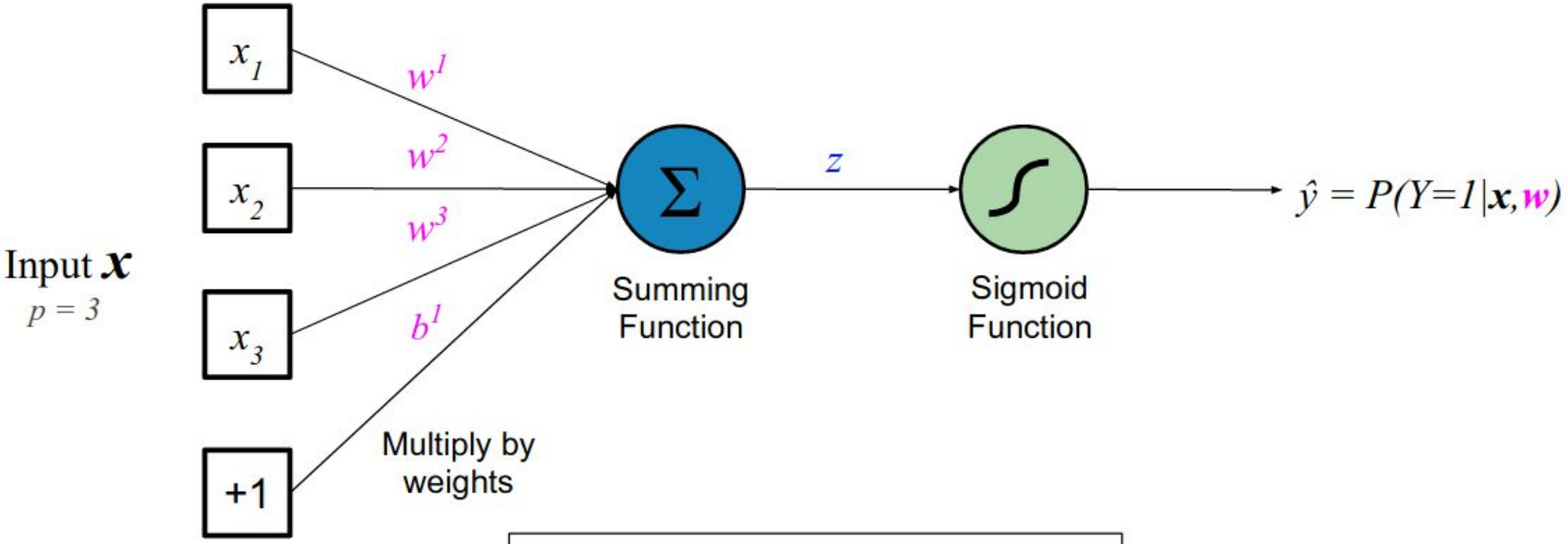


NN & Regression

COGNIBOT

AI meets Industry

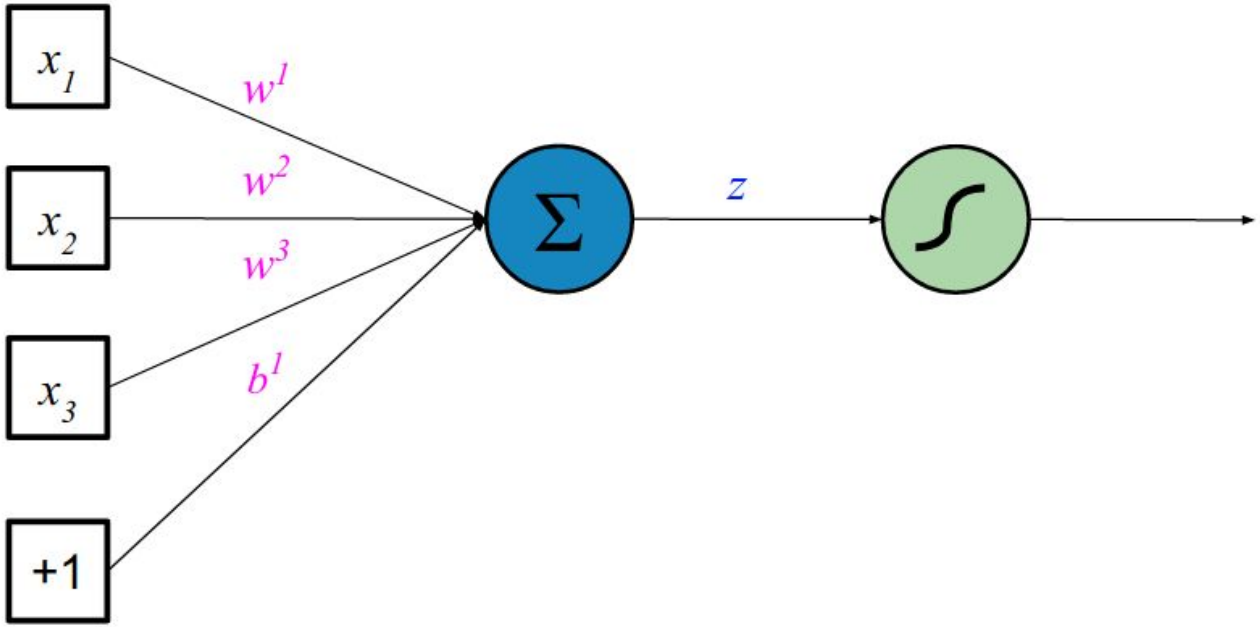
Expanded Logistic Regression



$$\underset{1\times1}{z} = \underset{1\times p}{\boldsymbol{w}^T}\underset{p\times1}{\boldsymbol{x}} + \boldsymbol{b}$$

$$y = sigmoid(z) = \frac{e^z}{1+e^z}$$

“Neuron”

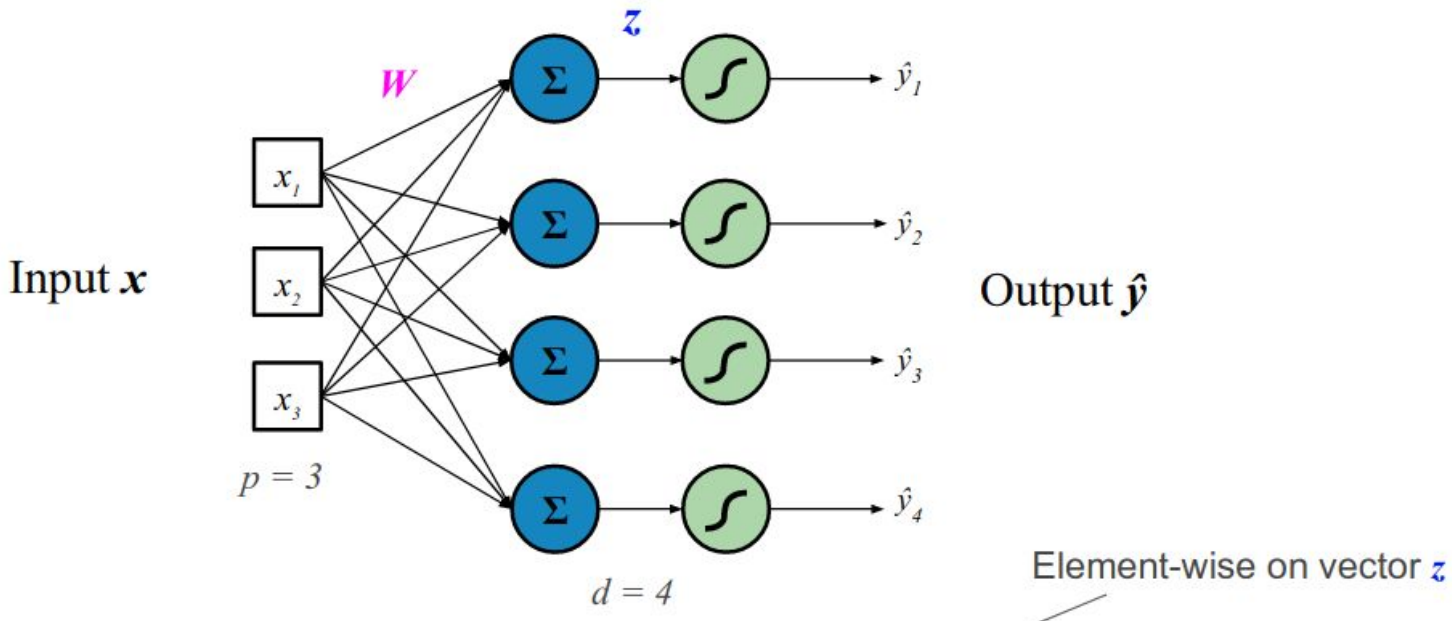


1-Layer network

COGNIBOT

AI meets Industry

1-Layer Neural Network (with 4 neurons)



Element-wise on vector z

$$\underset{d\times 1}{z} = \underset{d\times p}{W^T}\ \underset{p\times 1}{x}$$

$$\underset{d\times 1}{\hat{y}} = sigmoid(\underset{d\times 1}{z}) = \frac{e^z}{1+e^z}$$

Activation function

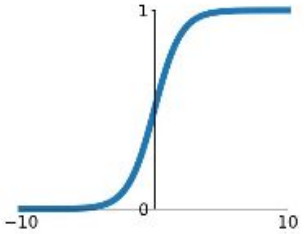
COGNIBOT

AI meets Industry

Activation Functions

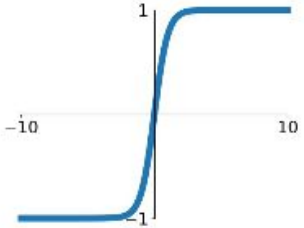
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



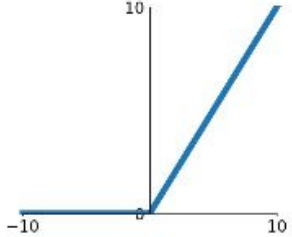
tanh

$$\tanh(x)$$



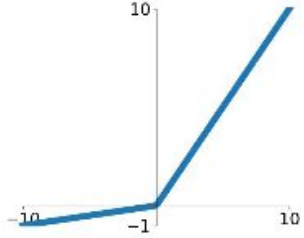
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

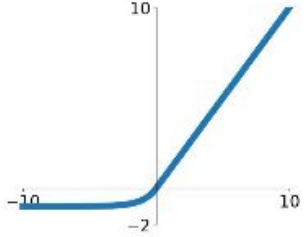


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

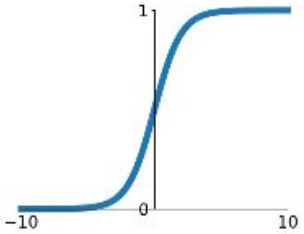
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation Functions

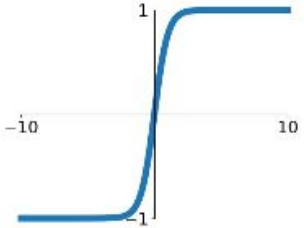
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



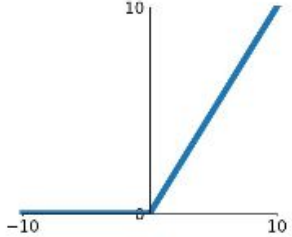
tanh

$$\tanh(x)$$



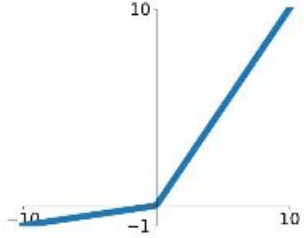
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

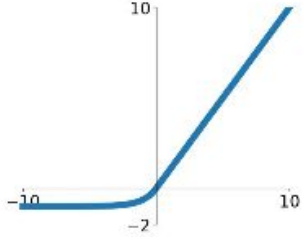


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Training

COGNIBOT

AI meets Industry

COGNIBOT
AI meets Industry

See Google Dev.

COGNIBOT

AI meets Industry

Implementation

COGNIBOT

AI meets Industry

Full implementation of training a 2-layer Neural Network needs ~20 lines:

```
1  import numpy as np
2  from numpy.random import randn
3
4  N, D_in, H, D_out = 64, 1000, 100, 10
5  x, y = randn(N, D_in), randn(N, D_out)
6  w1, w2 = randn(D_in, H), randn(H, D_out)
7
8  for t in range(2000):
9    h = 1 / (1 + np.exp(-x.dot(w1)))
10   y_pred = h.dot(w2)
11   loss = np.square(y_pred - y).sum()
12   print(t, loss)
13
14   grad_y_pred = 2.0 * (y_pred - y)
15   grad_w2 = h.T.dot(grad_y_pred)
16   grad_h = grad_y_pred.dot(w2.T)
17   grad_w1 = x.T.dot(grad_h * h * (1 - h))
18
19   w1 -= 1e-4 * grad_w1
20   w2 -= 1e-4 * grad_w2
```

Define the network

Forward pass

Calculate the analytical gradients

Gradient descent