Machine Learning

- Gradient Descent Algorithm
- Linear Regression
- Non-Linear Regression
- Logistic Regression
- Decision Trees
  - Regression Trees
  - Classification Trees
  - Model complexity and ensemble models
- Clustering Algorithms
  - K-Means
  - Hierarchical clustering
  - DB-Scan
  - Mean Shift
  - GMM
- Support Vector Machine

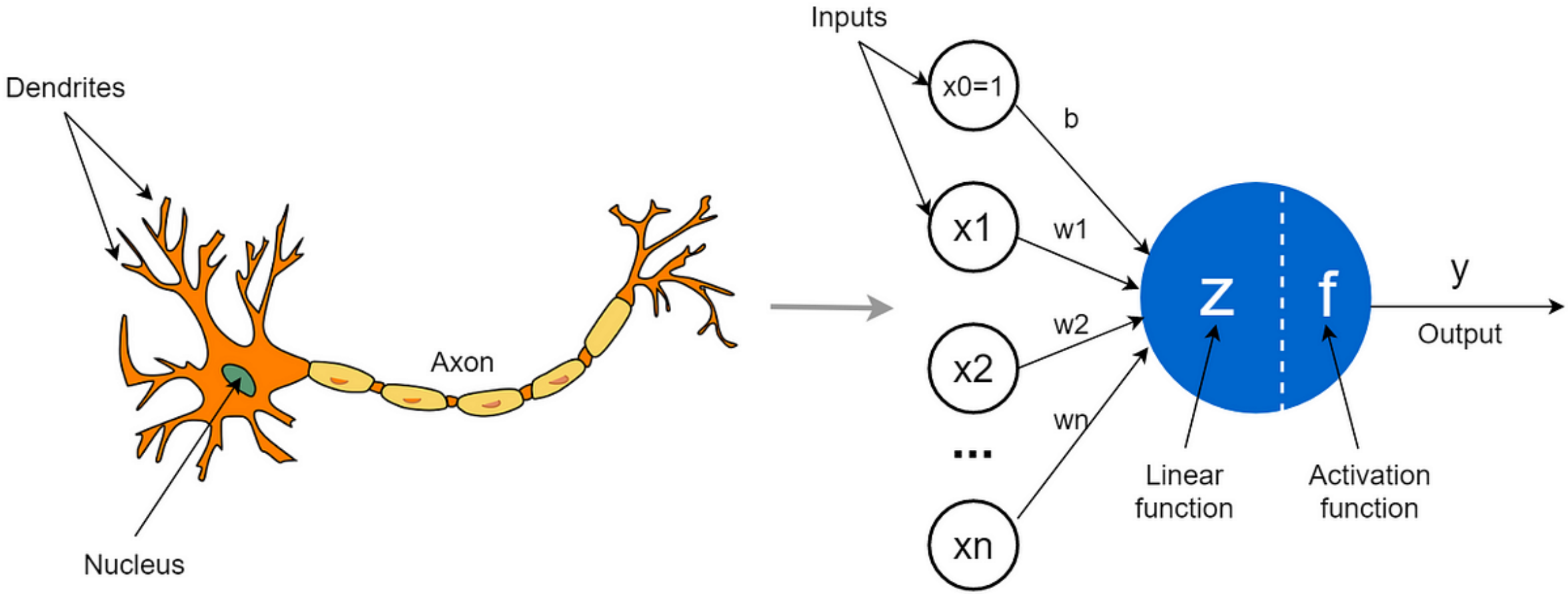
Datasets

- Breast Cancer Wisconsin
- MIMIC-III
- Framingham Heart Study
- Alzheimer’s Disease Neuroimaging Initiative
- Drug discovery
- Microbiome

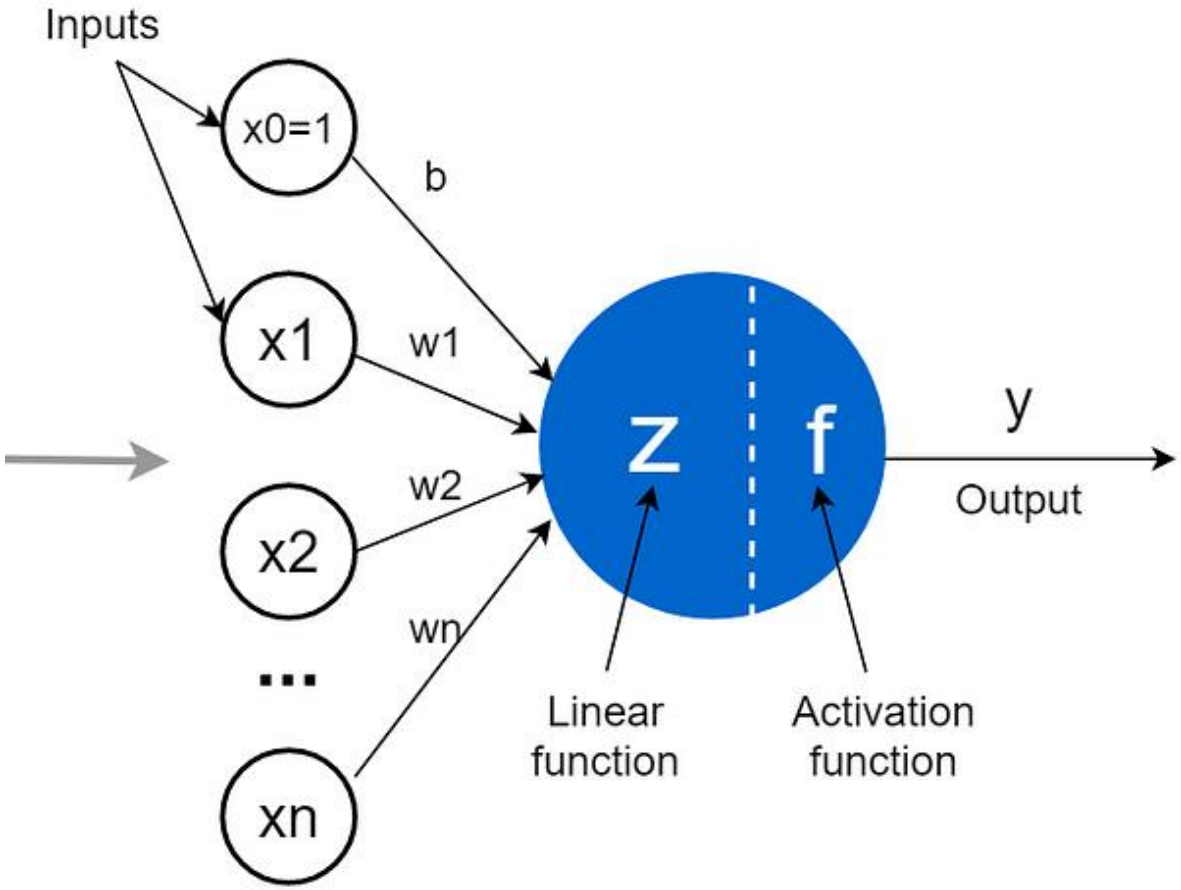
Deep Learning

- MLP
- **CNN**
- **CNN Arquitectures**
- **Autoencoders**
- E: Transfer learning y fine tunning
- Recurrent Neural Networks
- VAE
- GAN
- Transformers
- Diffusion Models
- Paper + Implementation

C9 - Intro to Deep Learning Multilayer Perceptron (MLP)


Dendrites
Axon
Nucleus
Inputs
x0=1
x1
x2
xn
...
b
w1
w2
wn
z
f
y
Output
Linear function
Activation function

Perceptron



$$z = (w1.x1 + w2.x2 + \ldots + wn.xn) + b$$

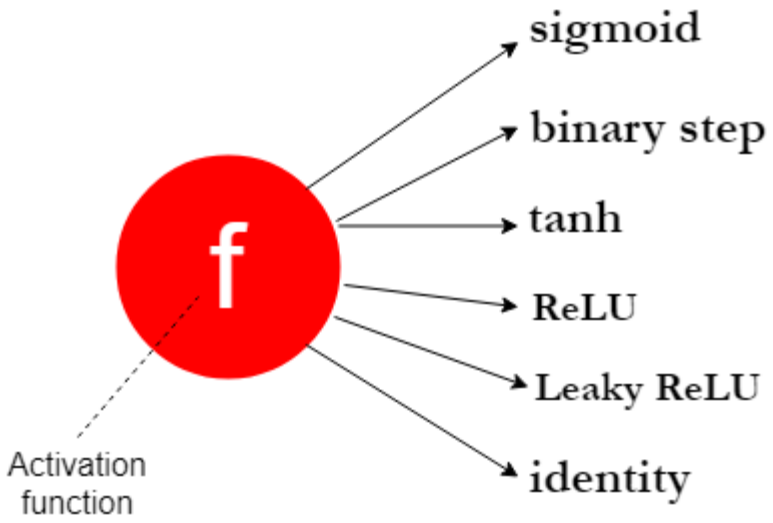
$$y = f((w1.x1 + w2.x2 + \ldots + wn.xn) + b)$$

Output

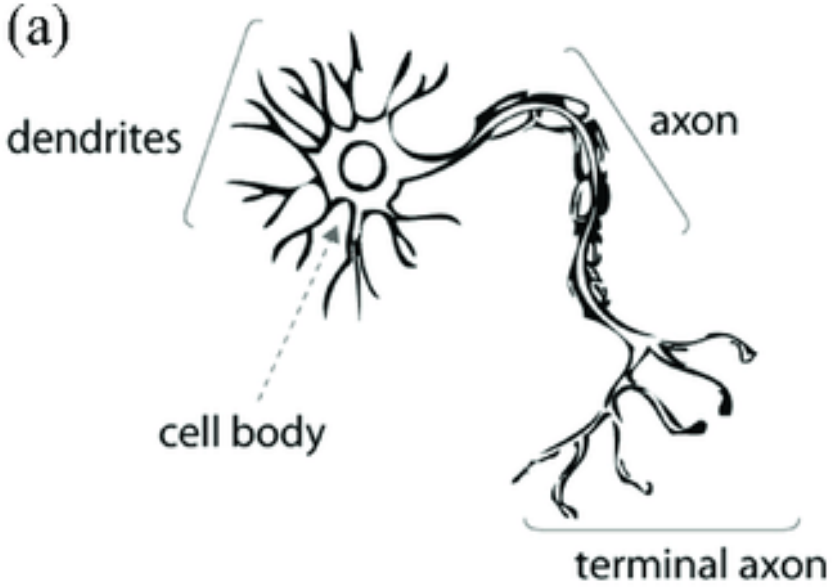
Activation function (non-linear)

Weighted sum of inputs (linear combination)

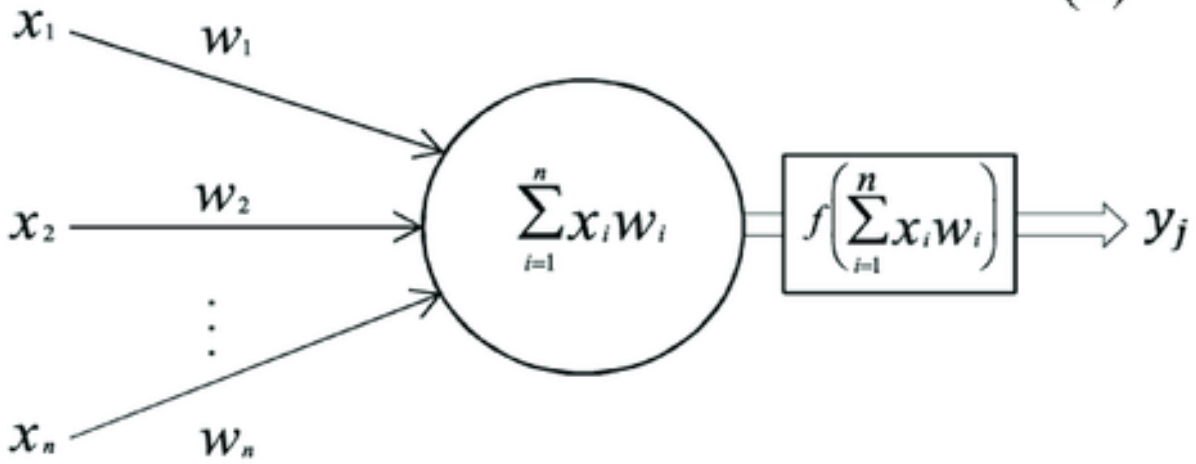
Bias



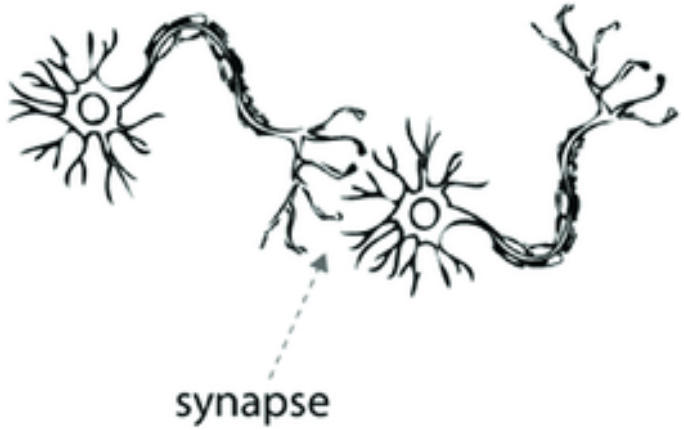
(a)


dendrites
axon
cell body
terminal axon

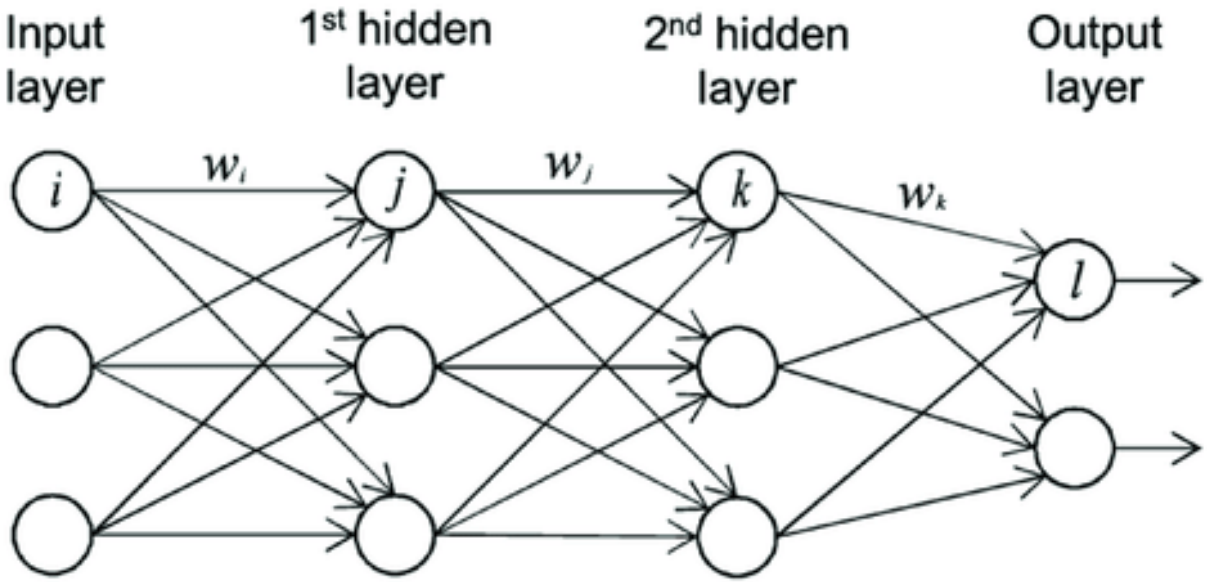
(b)


x_1
w_1
x_2
w_2
x_n
w_n
$\sum_{i=1}^{n} x_i w_i$
$f\left(\sum_{i=1}^{n} x_i w_i\right)$
y_j

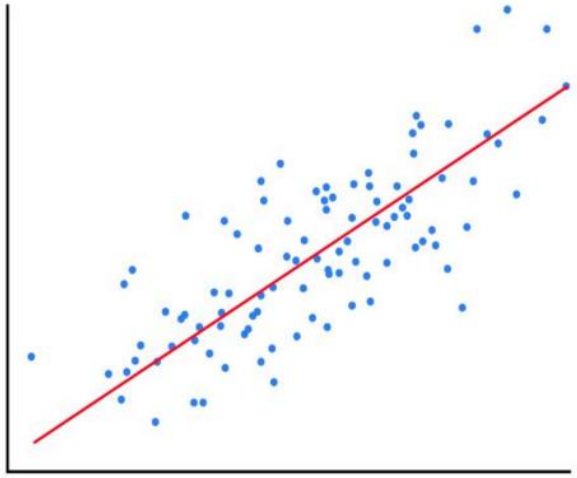
(c)

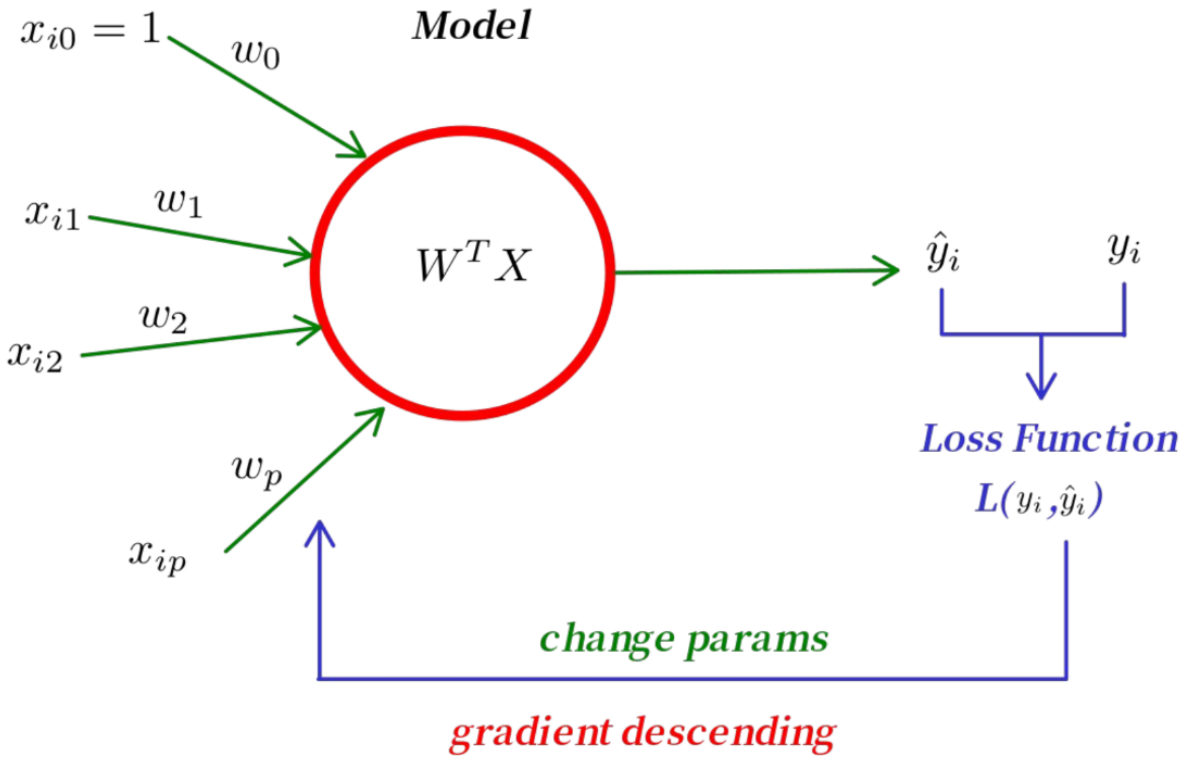

synapse

(d)


Input layer
1st hidden layer
2nd hidden layer
Output layer
w_i
w_j
w_k
i
j
k
l

Perceptron as a Linear Regression





Model: $f(x) = w_0 + w_1 x$

Parameters: w_0, w_1

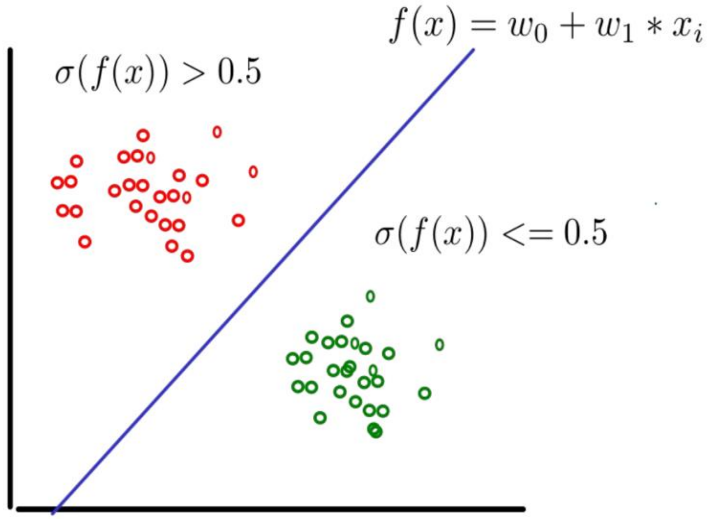
loss: $loss(y, f(x)) = \frac{1}{2n} \sum_{i=0}^{n} (y_i - f(x_i))^2$

gradiente: $\frac{\partial loss}{\partial w_0} = \frac{1}{n} \sum_{i=0}^{n} (y_i - f(x_i))(-1)$

$\frac{\partial loss}{\partial w_0} = \frac{1}{n} \sum_{i=0}^{n} (y_i - f(x_i))(-x_i)$

change parameters: $w_i = w_i - \alpha \frac{\partial loss}{\partial w_i}$

Perceptron as a Logistic Regression



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

Model: $$f(x) = \frac{1}{1+e^{-W^T X}}$$

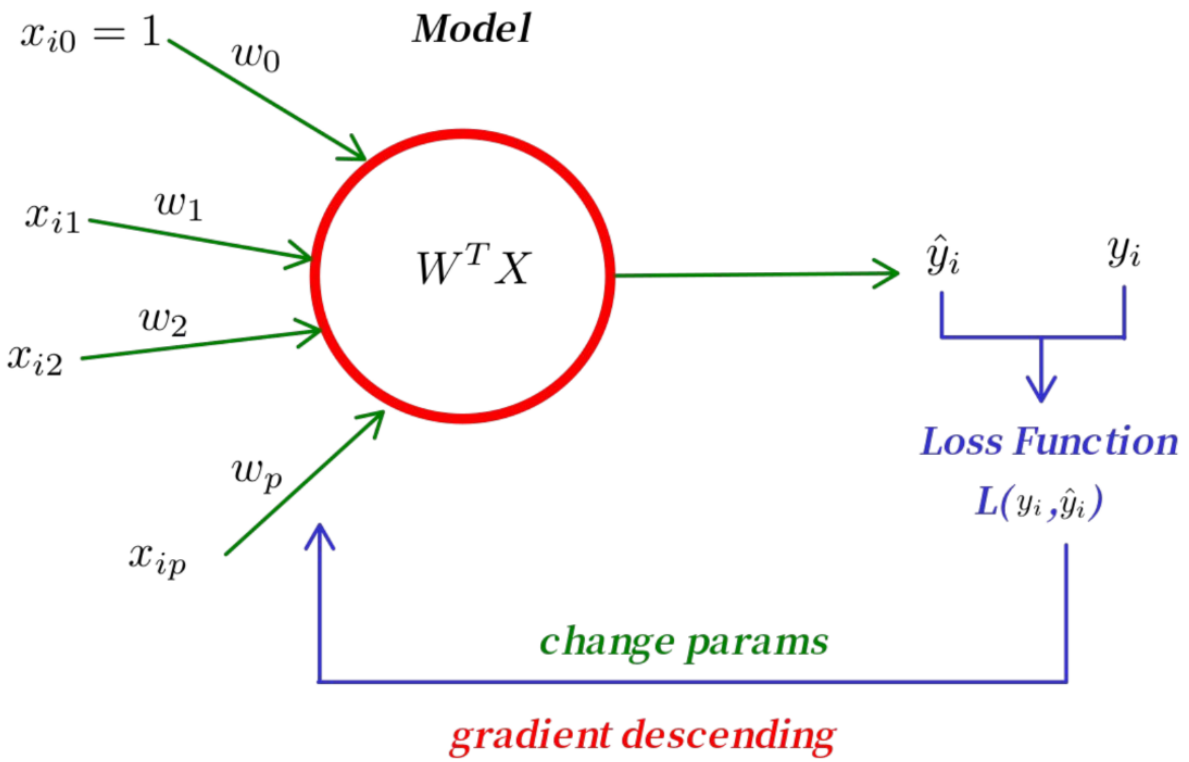
Parameters: $$w_0, w_1, w_2, ..., w_p$$

loss:

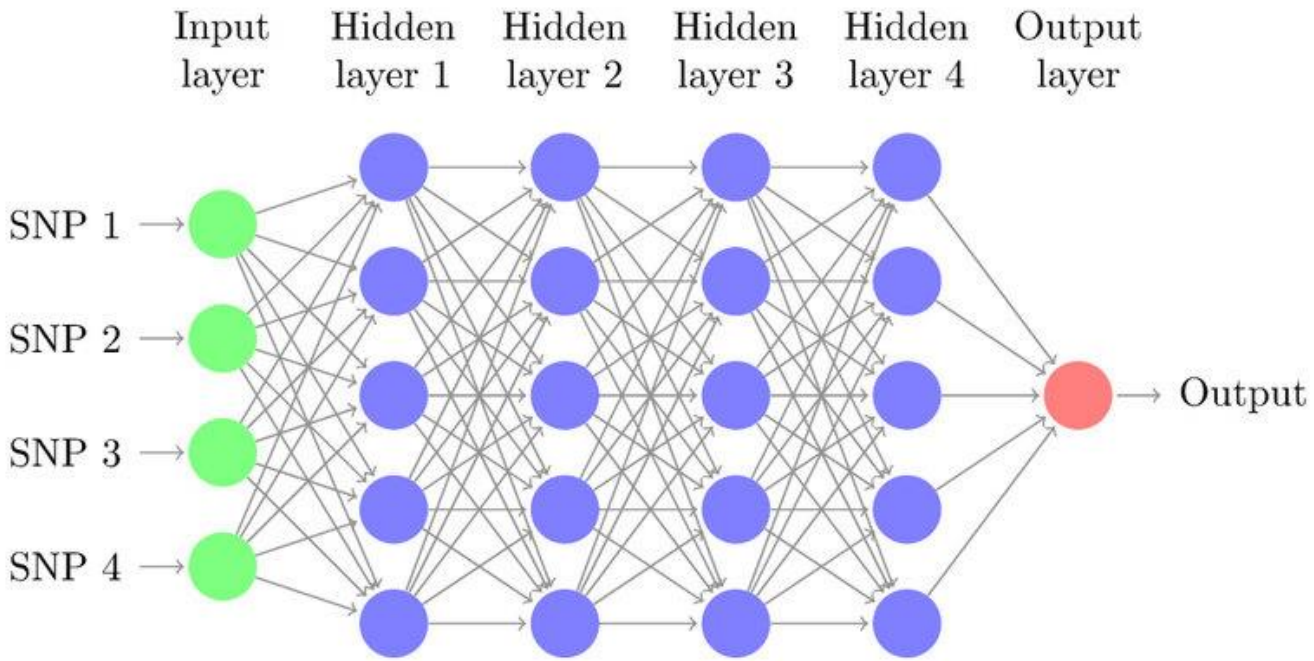
$$loss(y, f(x)) = -\frac{1}{n}\sum_{i=0}^{n}[y_i log(f(x_i)) + (1-y_i) log(1 - log(f(x_i)))]$$

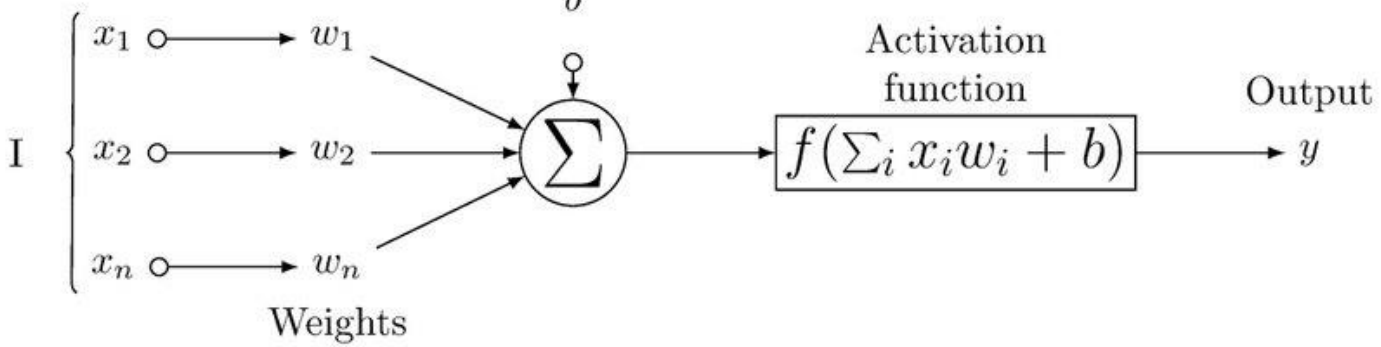
gradiente: $$\frac{\partial loss}{\partial w_j} = \frac{1}{n}\sum_{i=0}^{n}(y_i - f(x_i))(-x_{ij})$$

change parameters: $$w_i = w_i - \alpha \frac{\partial loss}{\partial w_i}$$



Multilayer perceptron





Universal approximation theorem

f() = “Perro”

f() = 

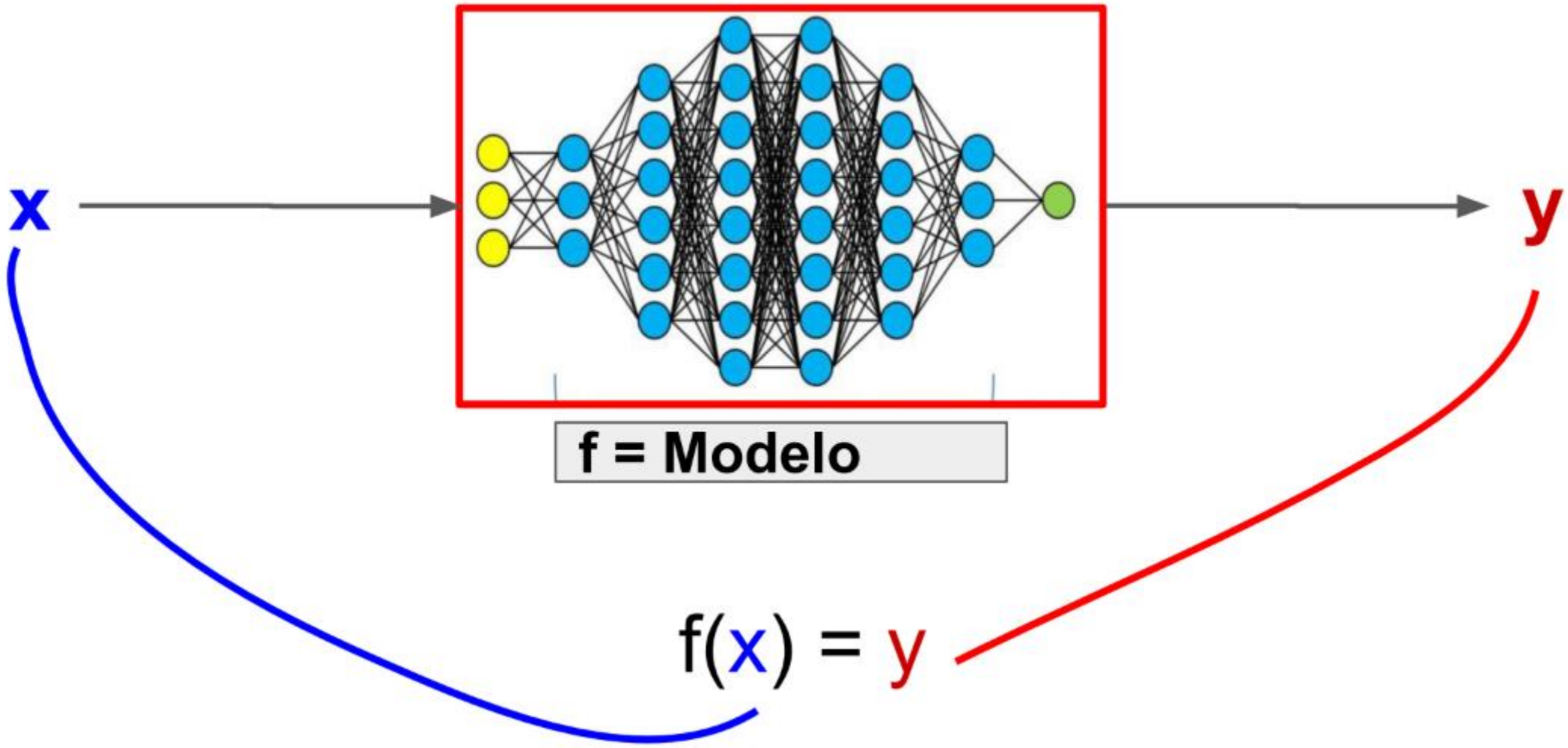
f() = 

f() = 

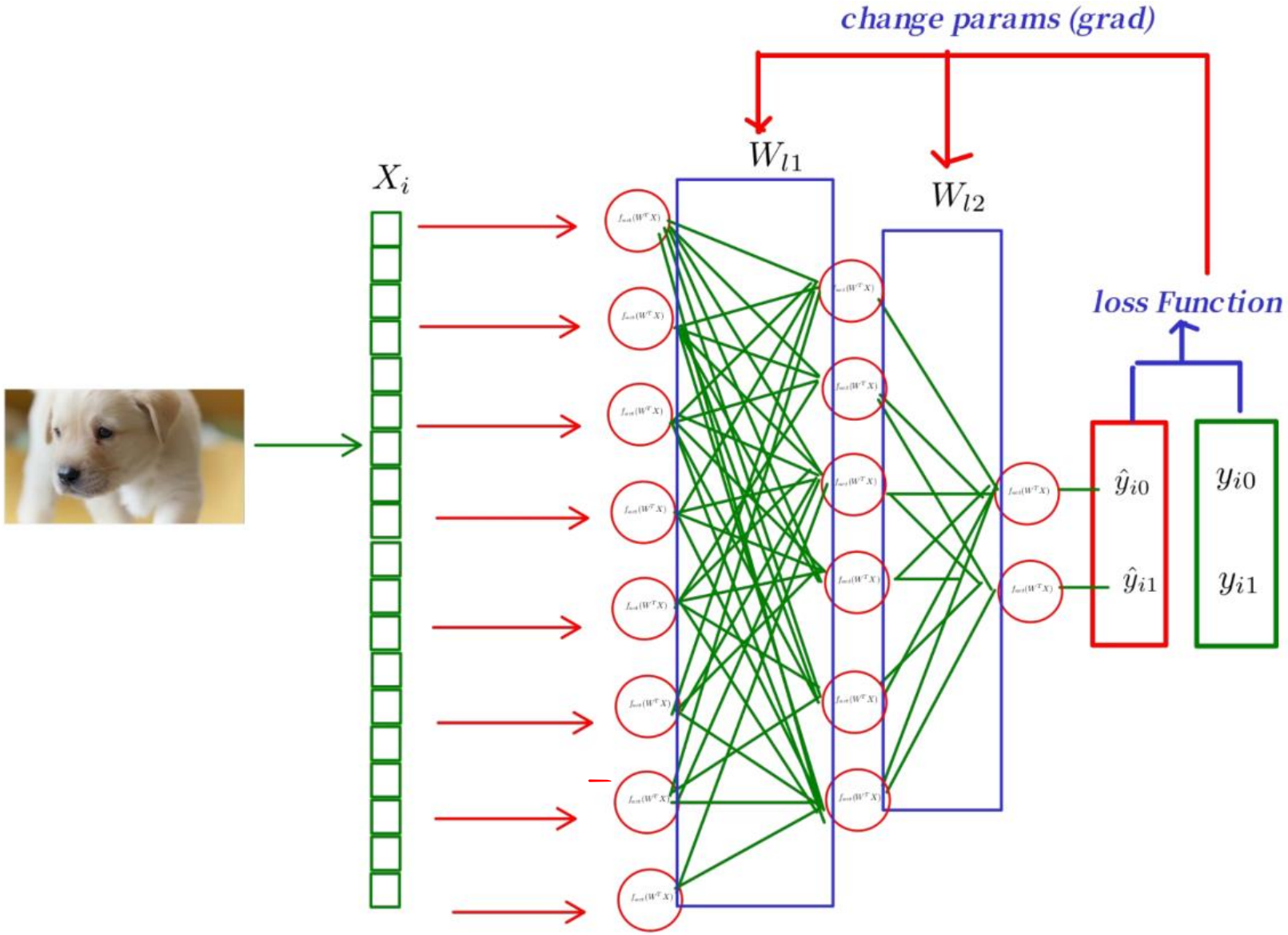
f() = 

f() = “Gaviota”

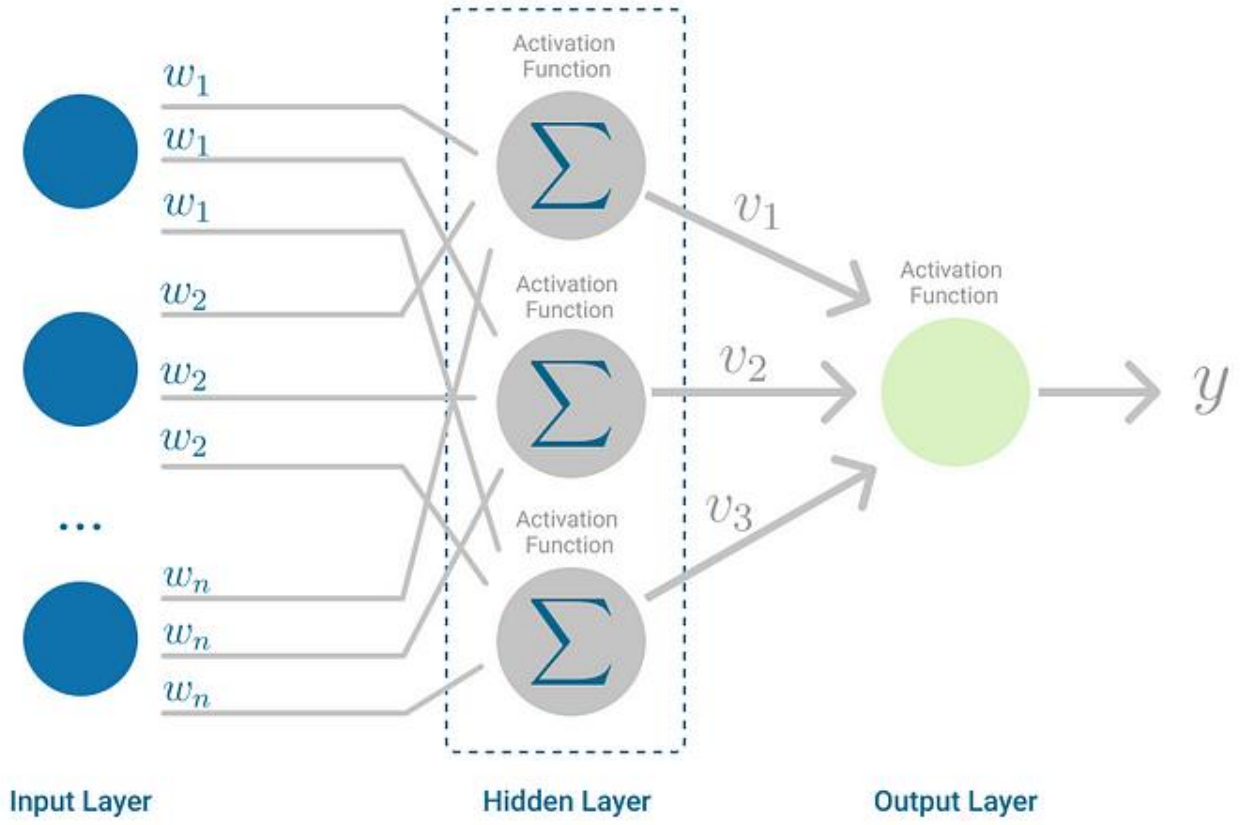
MLP as a model



MLP learning



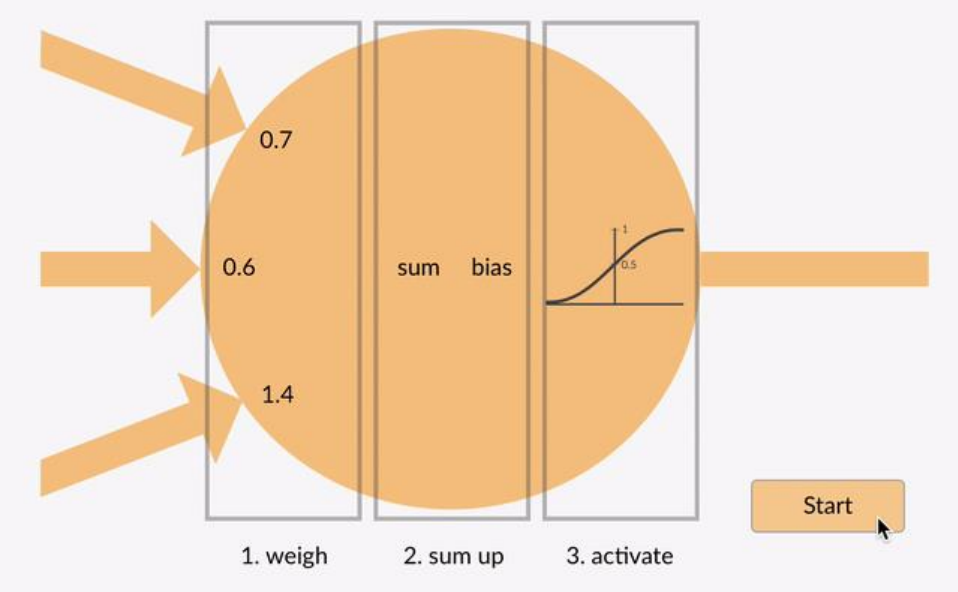
MLP learning

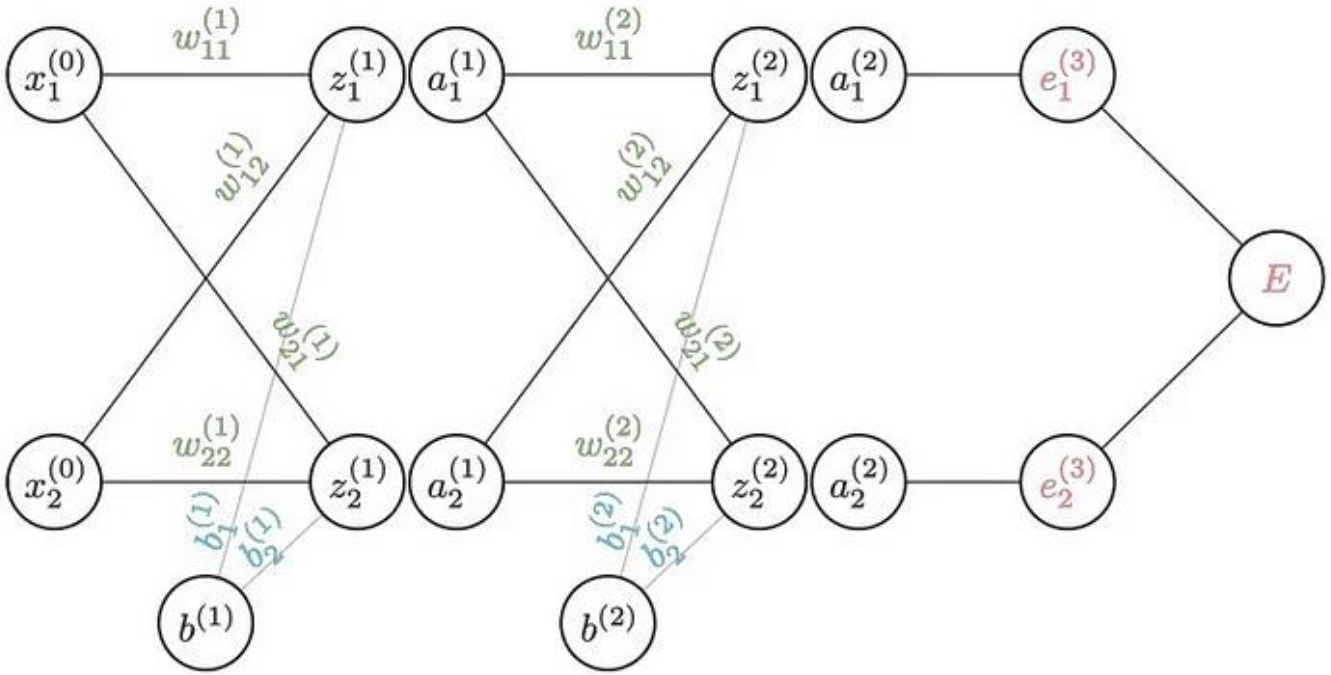


1. Feedforward | Mean Squared Error (MSE) computed

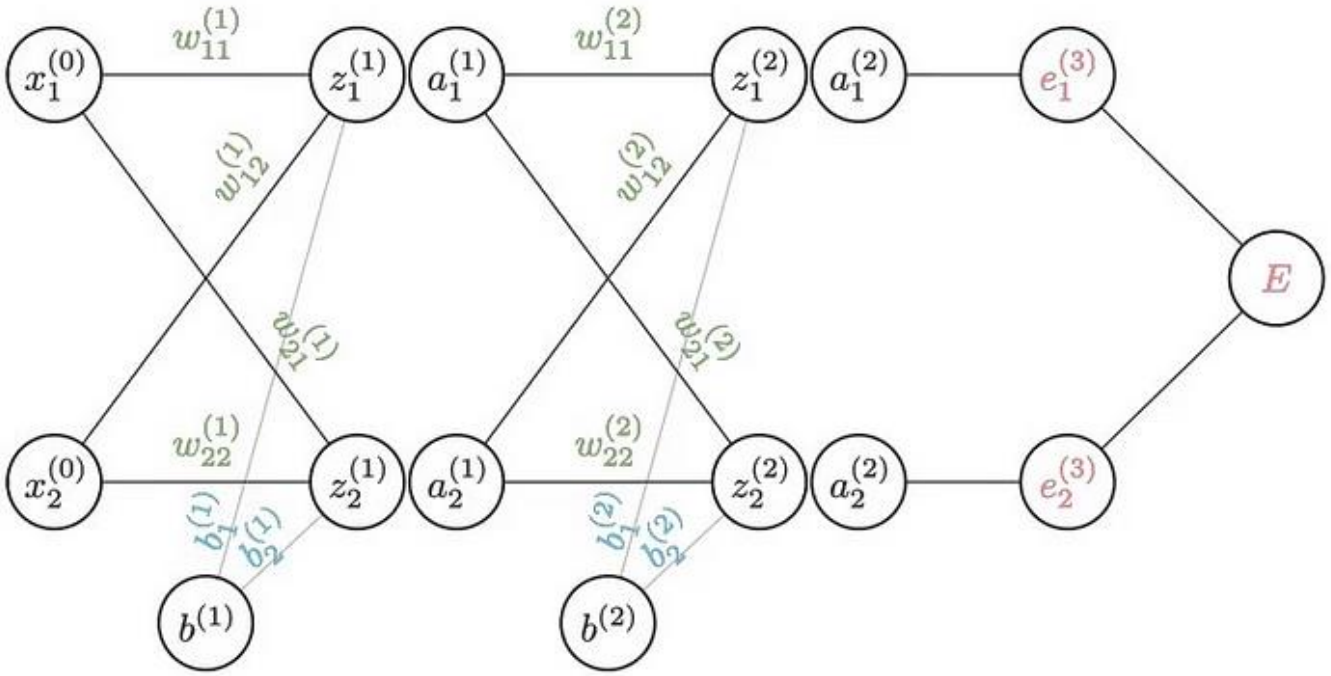
2. Backpropagation | Gradient is computed

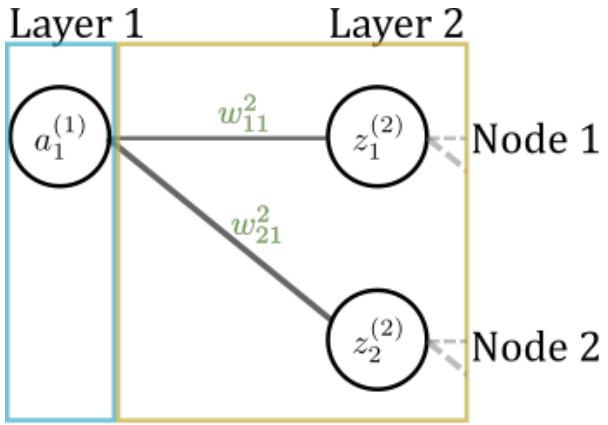
Forward pass

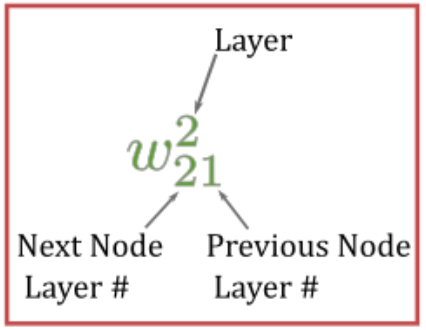


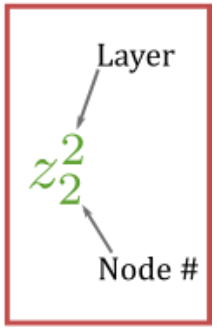


Forward pass

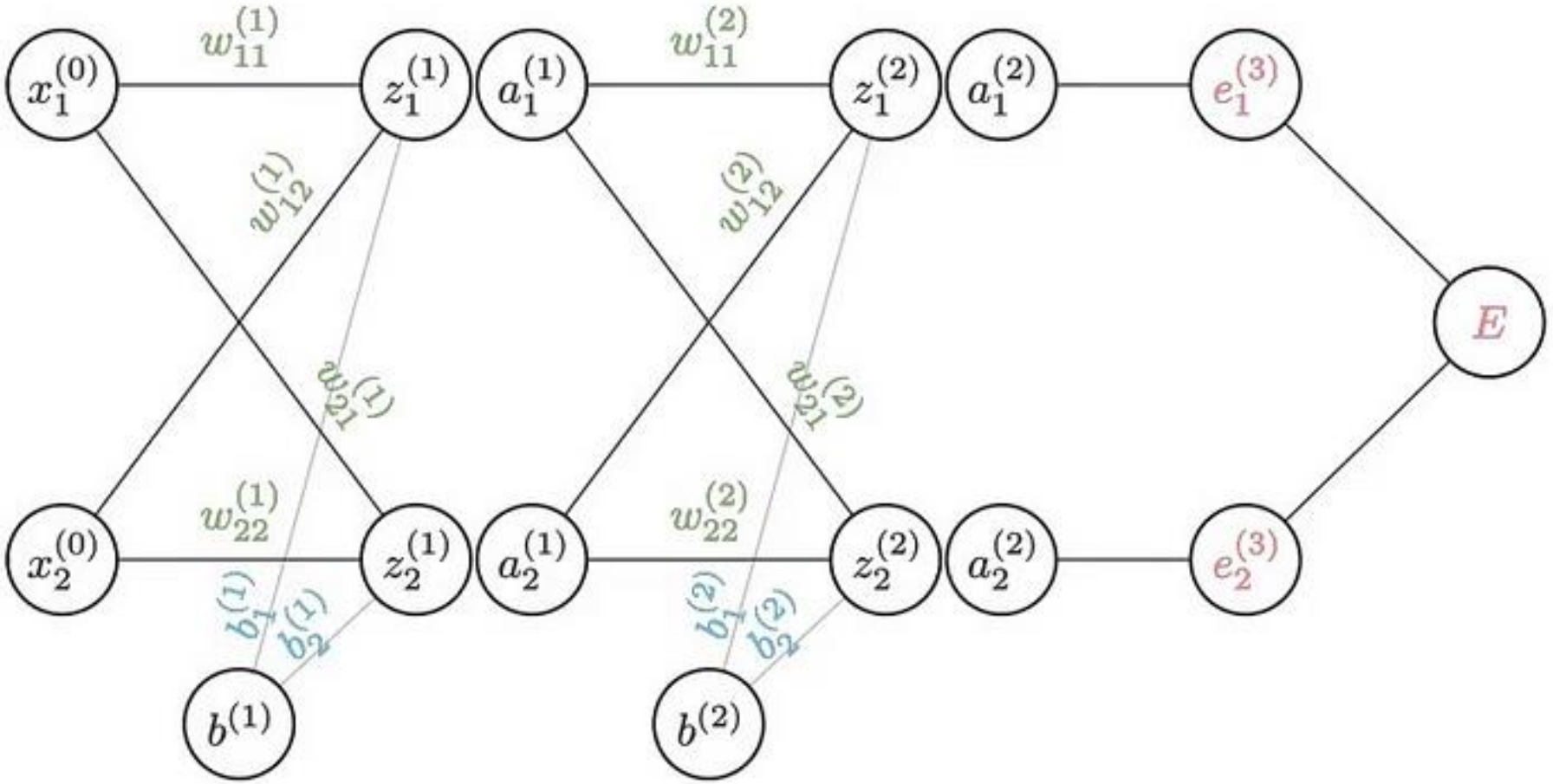






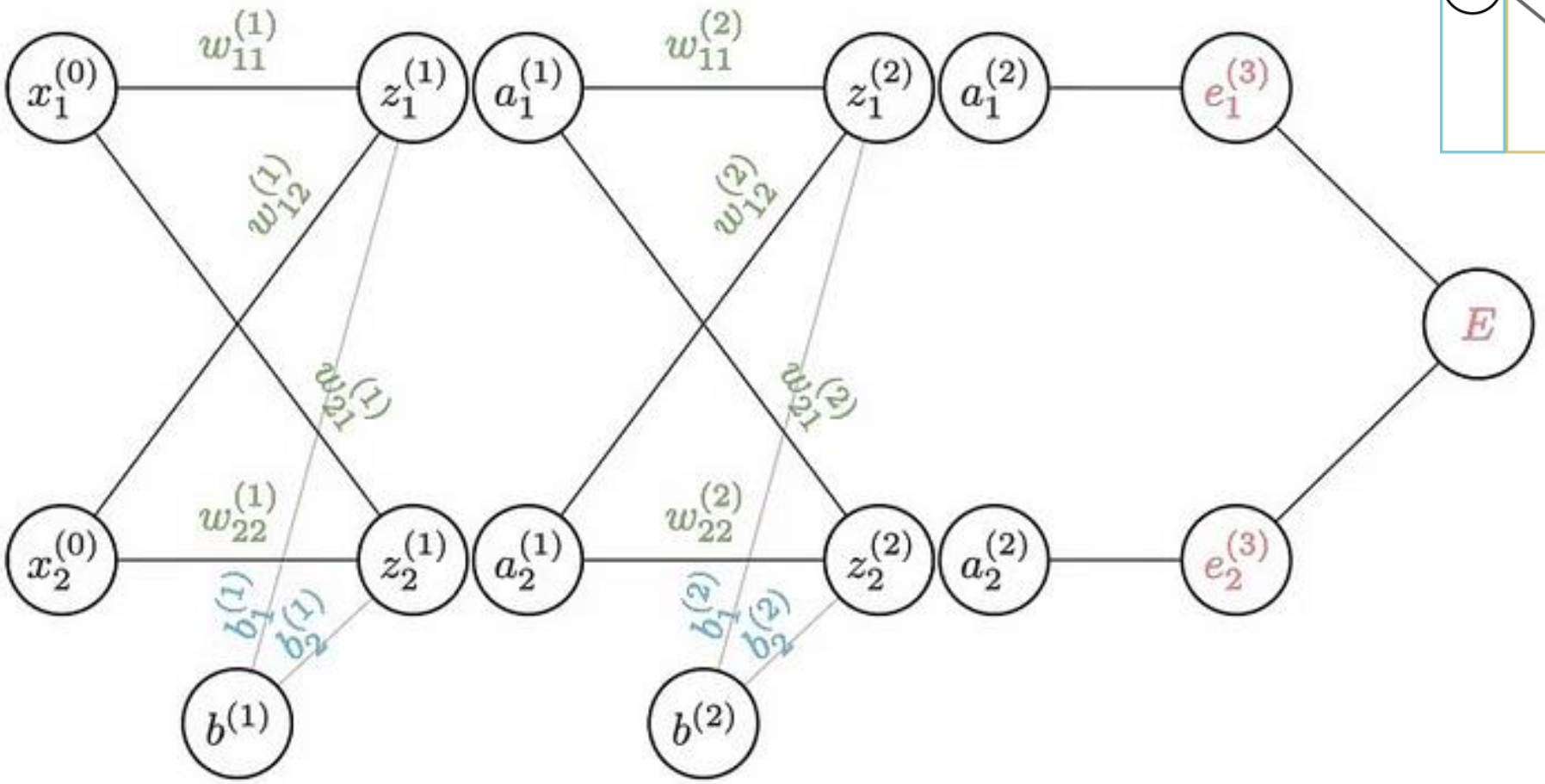


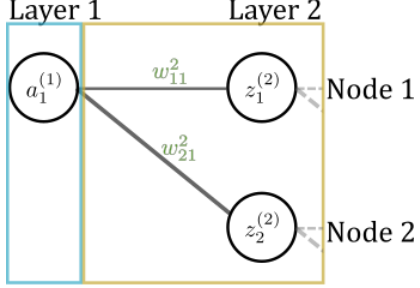
Basics of Backpropagation

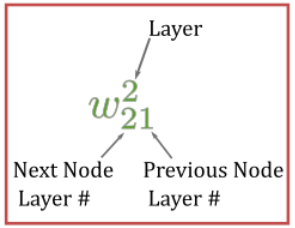


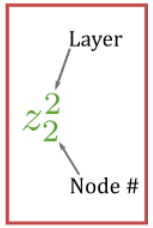
https://towardsdatascience.com/understanding-backpropagation-abcc509ca9d0

my humble ML course

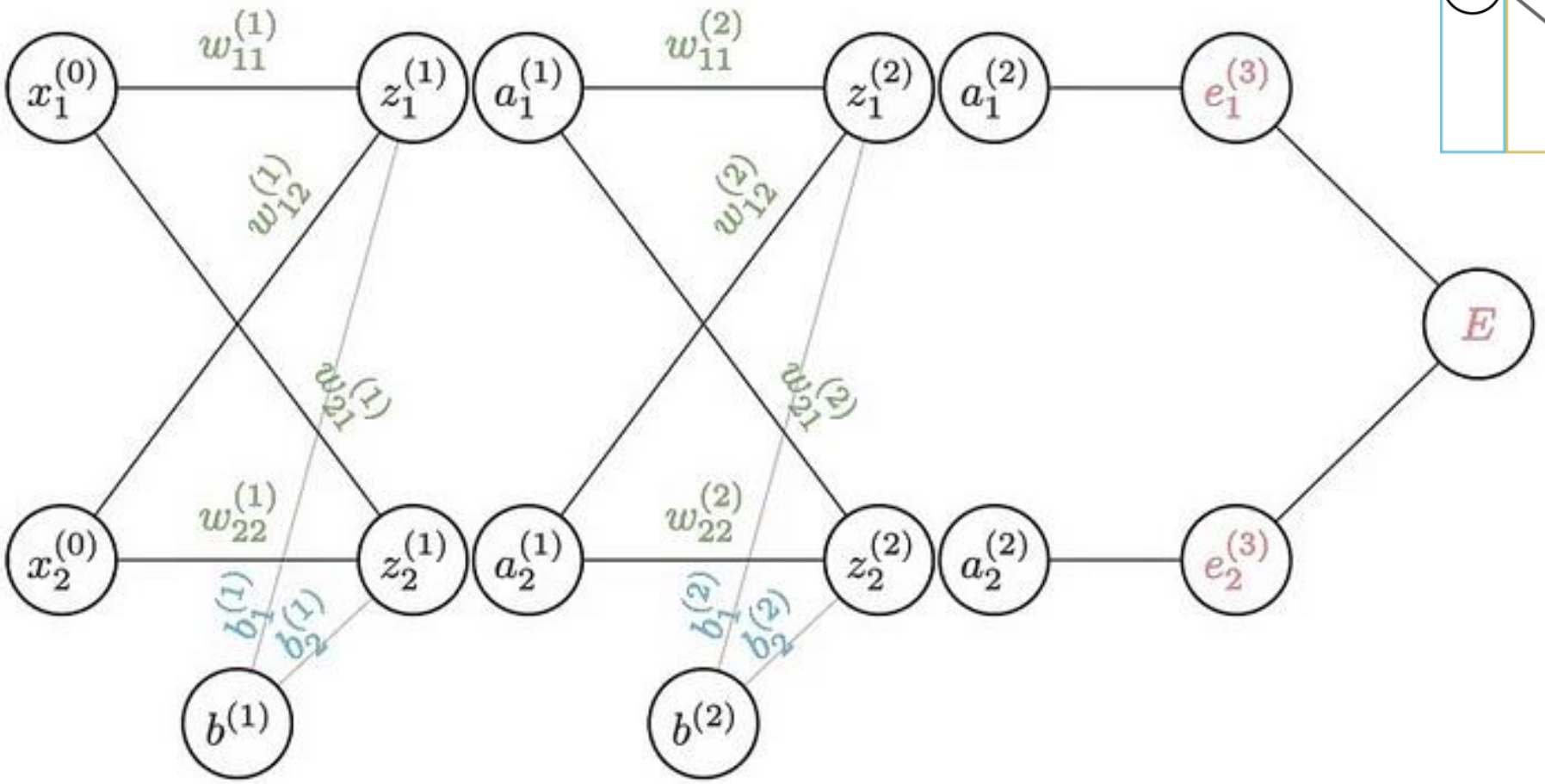


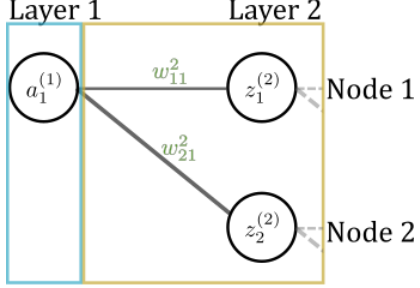


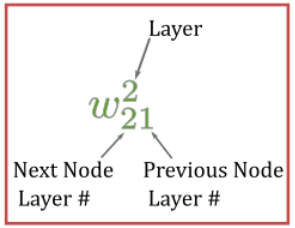


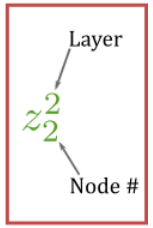


my humble ML course









Bibliography

https://towardsdatascience.com/pytorch-introduction-building-your-first-linear-model-d868a8681a41
https://medium.com/@mohamedyosef101/understanding-neural-networks-by-building-one-from-scratch-using-numpy-61500b9ca882
https://medium.com/@evertongomede/understanding-backpropagation-the-engine-behind-neural-network-learning-a7c2e1acdbf
https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141
https://medium.com/@rukshanpramoditha/list/neural-networks-and-deep-learning-course-a2779b9c3f75
https://towardsdatascience.com/understanding-backpropagation-abcc509ca9d0
https://youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&si=9LsJ8HjltbuLtFqs
https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd

C9 - Gradient Optimizers

Bibliography

https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-gradient-based-optimizers/
https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
https://www.ruder.io/optimizing-gradient-descent/
https://arxiv.org/pdf/1609.04747.pdf
https://medium.com/@nerdjock
https://medium.com/analytics-vidhya/optimizers-gradient-descent-algorithms-part-1-ac9baf446df1
https://naokishibuya.medium.com/gradient-descent-optimizers-80d29f22deb5
https://towardsdatascience.com/effect-of-gradient-descent-optimizers-on-neural-net-training-d44678d27060
https://towardsdatascience.com/neural-network-optimizers-made-simple-core-algorithms-and-why-they-are-needed-7fd072cd2788
https://iq.opengenus.org/types-of-gradient-optimizers/
https://medium.com/@dancerworld60/optimizers-part-1-gradient-descent-and-its-variants-d11dac1d6b2
https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6