# Statistical & Machine Learning Approaches for Marketing

## Section 7 - Neural Networks

**Minh Phan**
m.phan@ieseg.fr
IÉSEG School of Management

# Outline

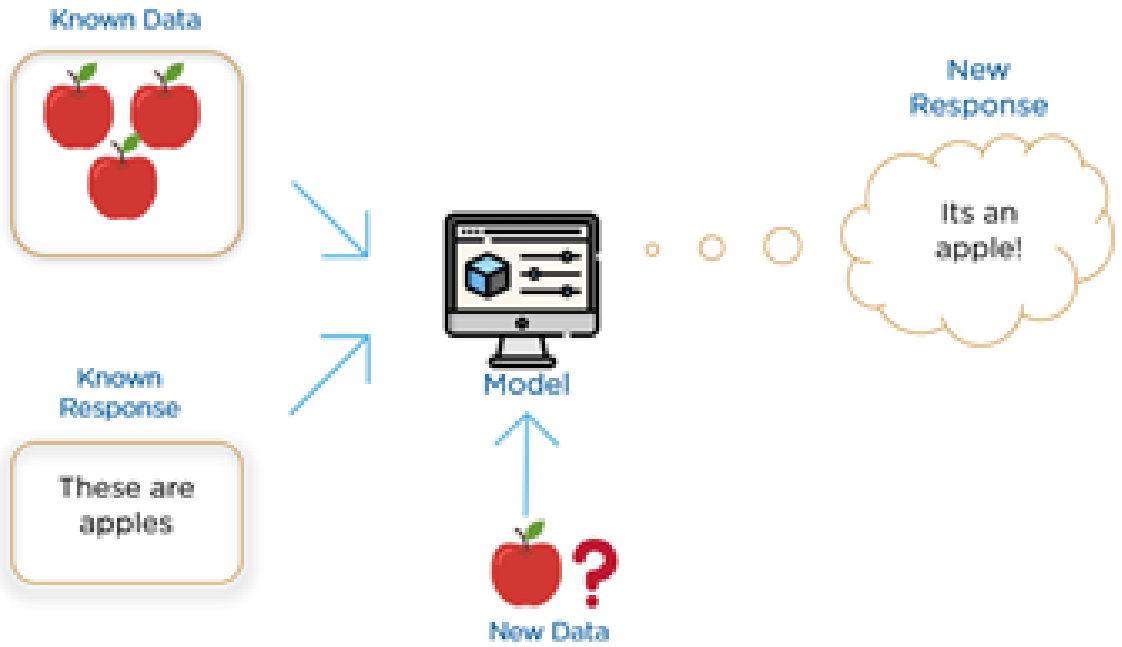
- I. Review: Unsupervised Learning
- II. Neural Networks
- Lab: Neural Networks

# Outline

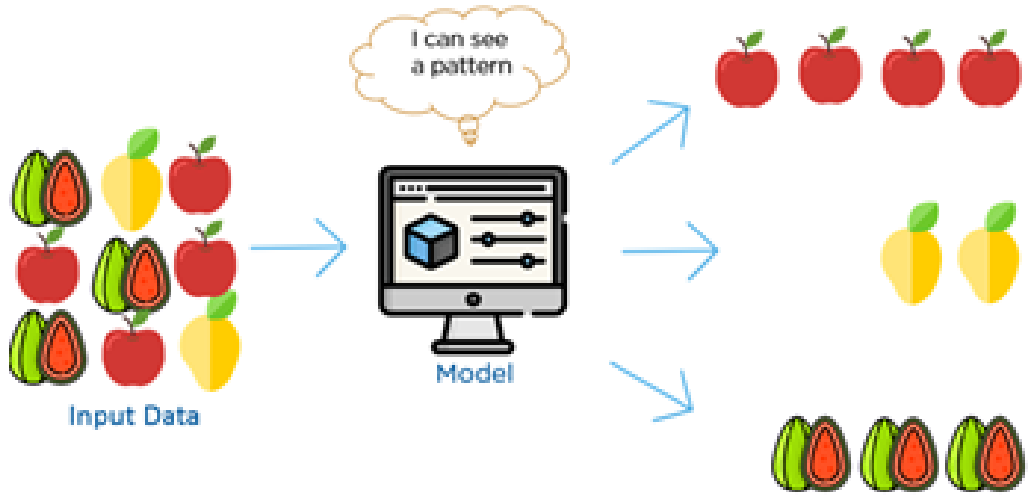
- I. Review: Unsupervised Learning
- II. Neural Networks
- Lab: Neural Networks

# I. Review: Unsupervised Learning | Supervised vs. Unsupervised

Supervised Learning
(Predicting)




Unsupervised Learning
(Exploring)




Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.
Source: Sandeep Dayananda. (2018). Link: https://www.quora.com/What-is-the-difference-between-supervised-and-unsupervised-learning-algorithms

# I. Review: Unsupervised Learning | Why?

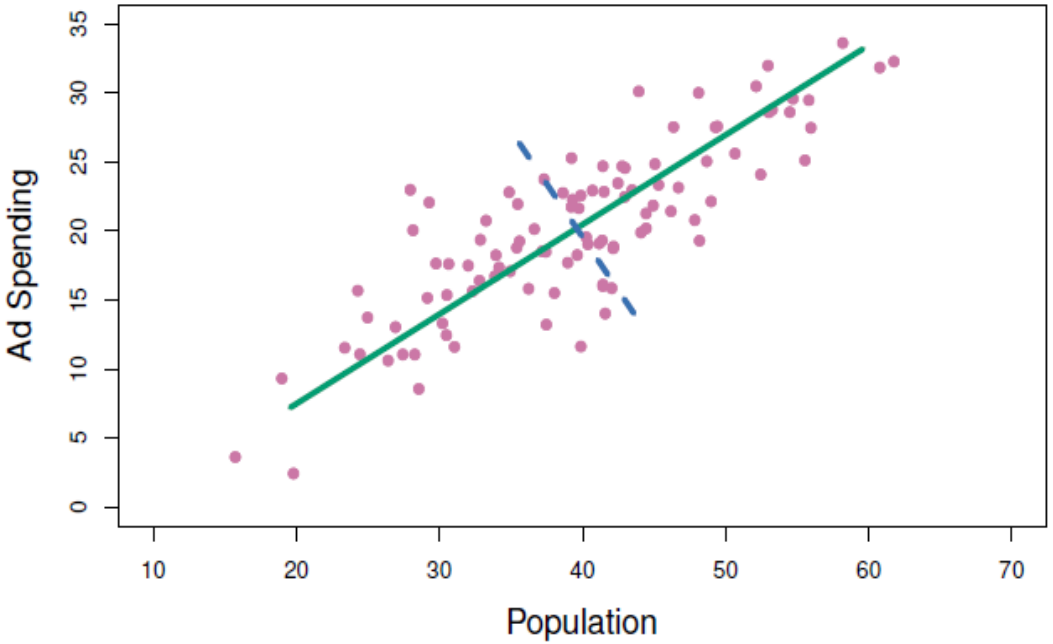
- Is there an informative way to visualize the data?

- Can we discover subgroups among the variables or among observations?

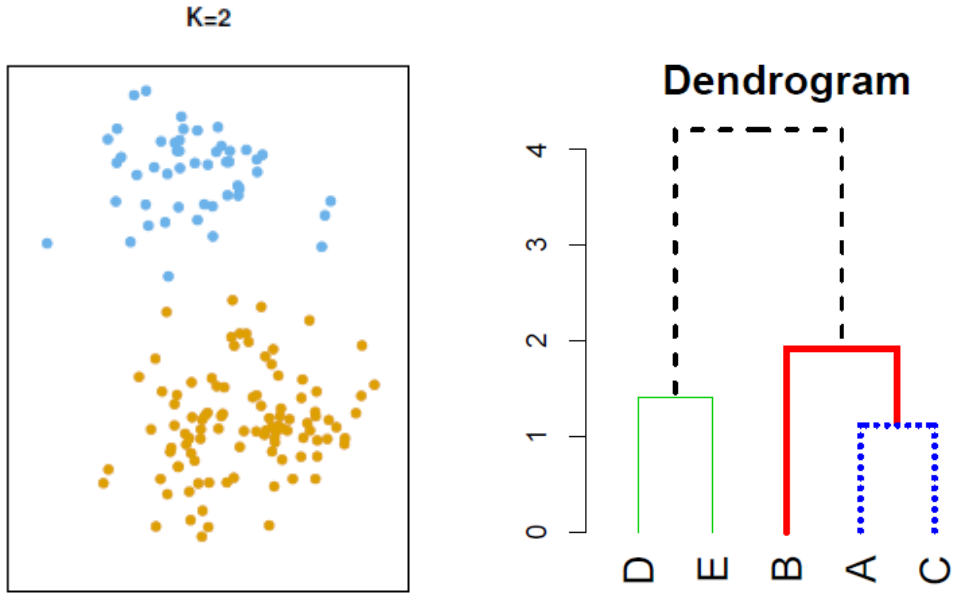
- Applications:
  - Grouping breast cancer patients by their gene expression;
  - Grouping shoppers by their browsing and purchase histories;
  - Grouping movies by ratings.

Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# I. Review: Unsupervised Learning | How?




Principal Components Analysis (PCA)

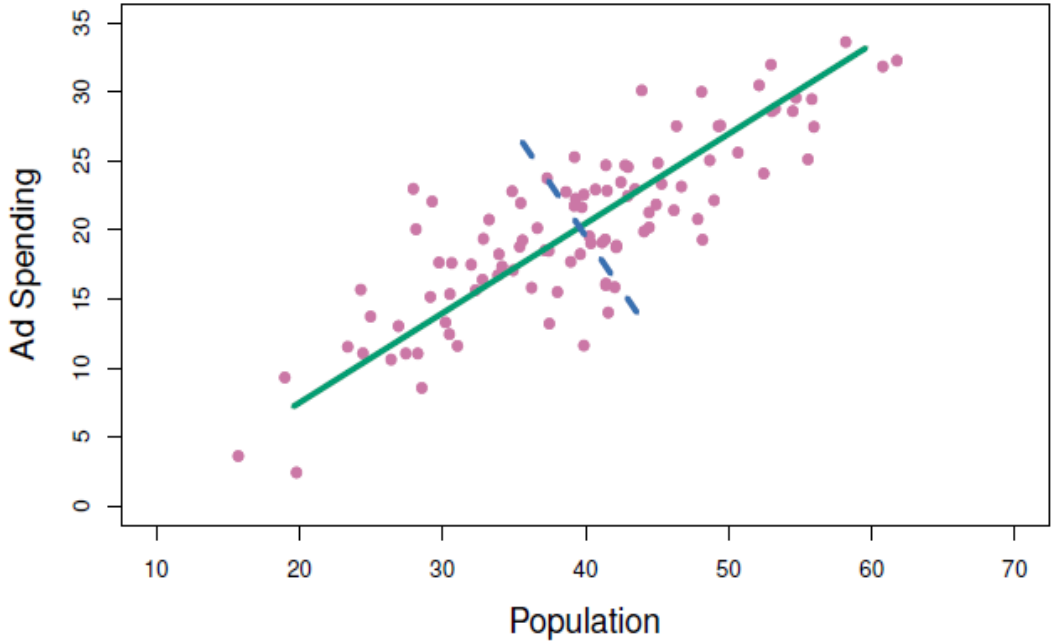


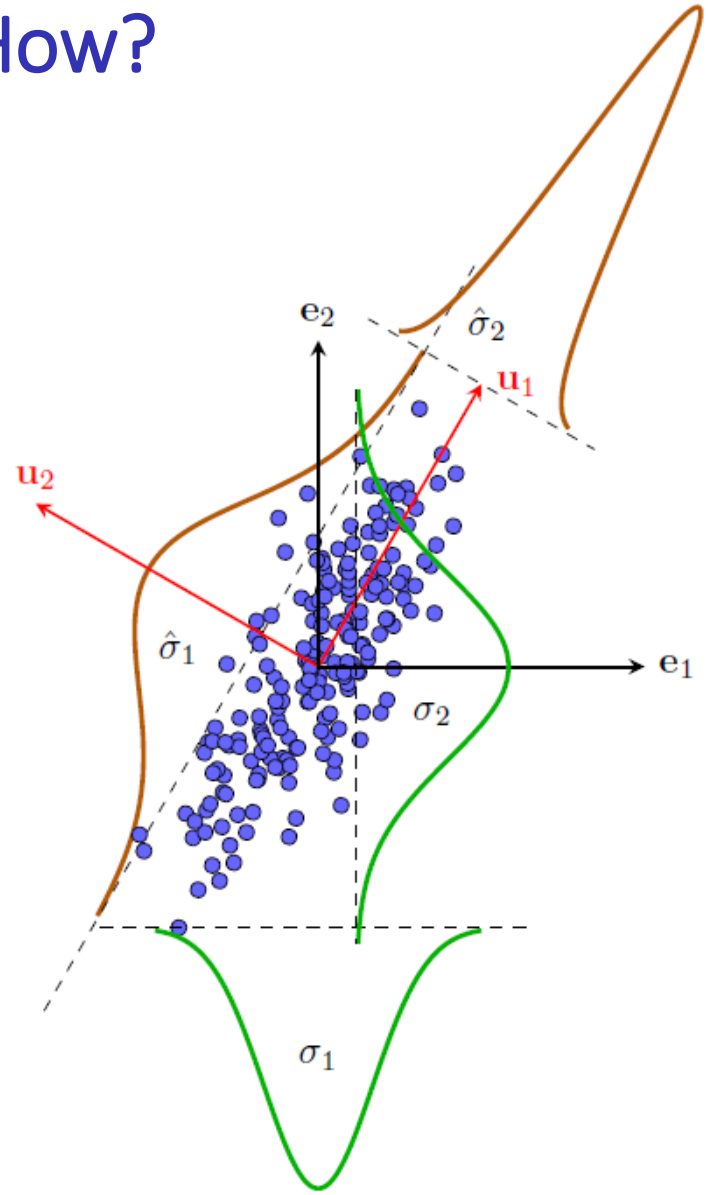

Clustering (K-Means, Hierarchical clustering)

Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# I. Review: Unsupervised Learning | PCA | How?

- Maximize variance.




Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.
Source: Vu Huy Tiep (2018). *Machine Learning Co Ban*. Link: https:// machinelearningcoban.com

# I. Review: Unsupervised Learning | PCA | Proportion Variance Explained (PVE)

- The variance explained by the $m^{th}$ principle component:

$$\mathrm{Var}(Z_m) = \frac{1}{n}\sum_{i=1}^{n} z_{im}^2$$

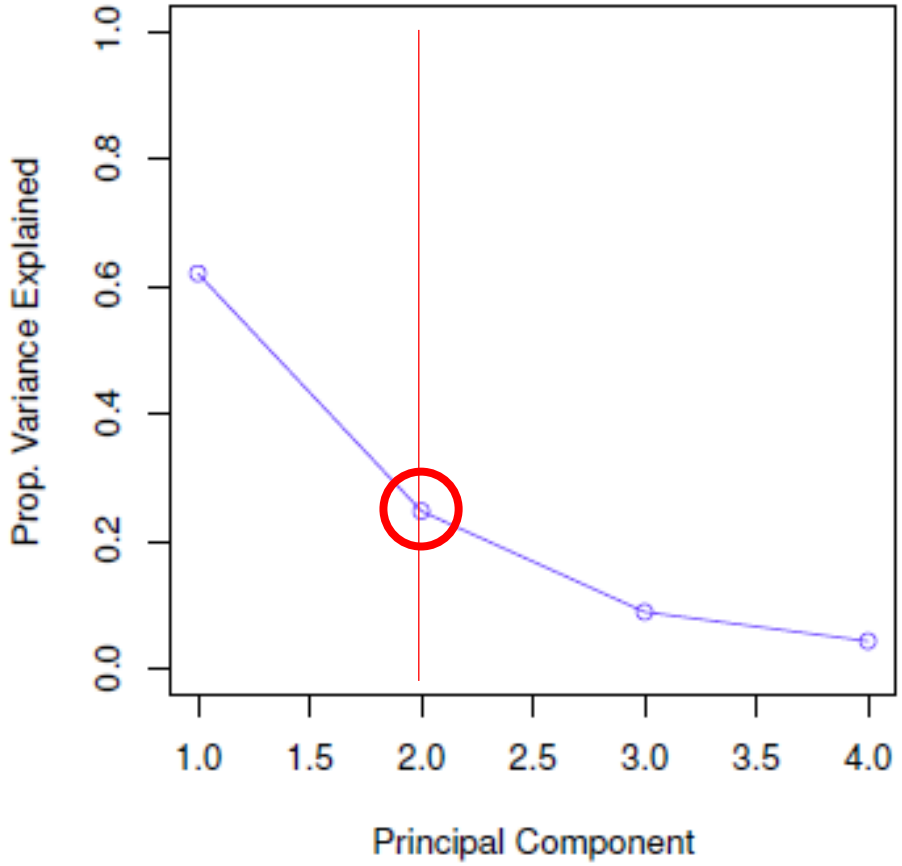
- PVE of the $m^{th}$ principal component:

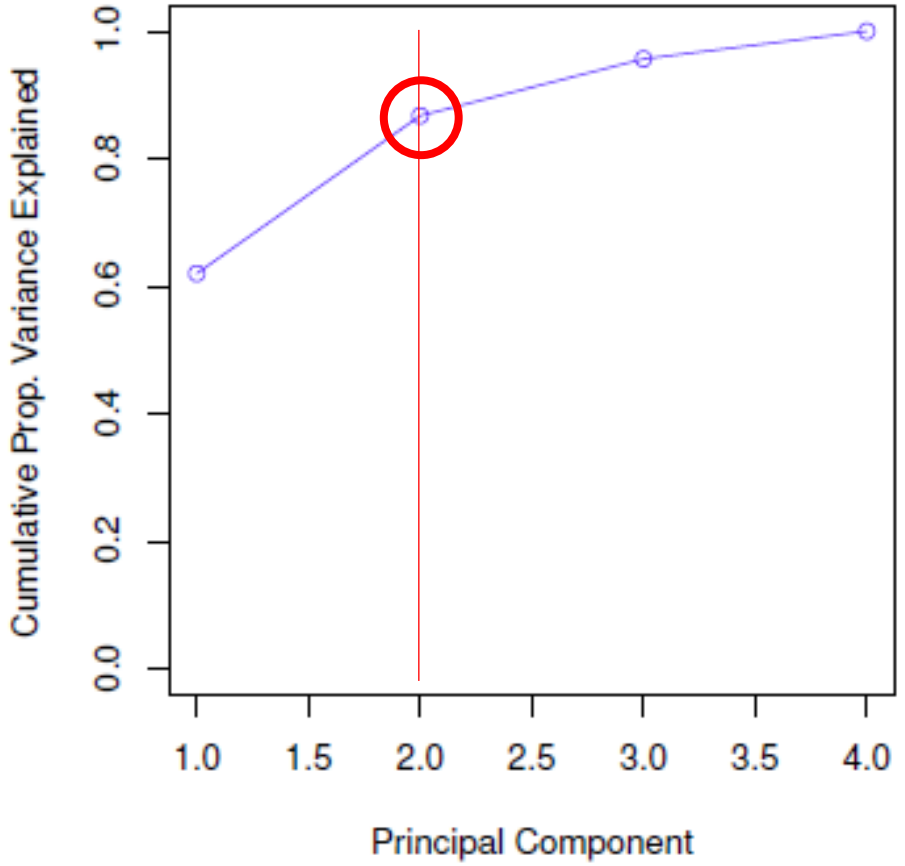
$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p}\sum_{i=1}^{n} x_{ij}^2}$$

Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# I. Review: Unsupervised Learning | PCA | Proportion Variance Explained (PVE)

- US Crime Data: Scree plot – PVE and Cumulative PVE

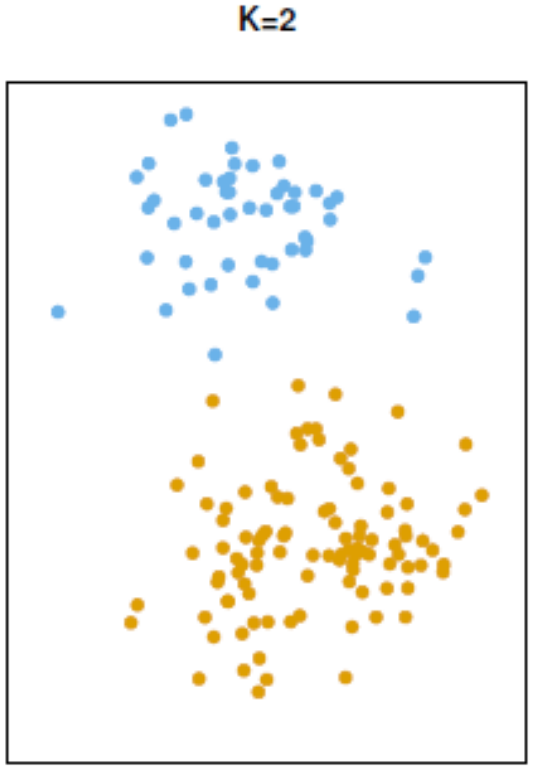


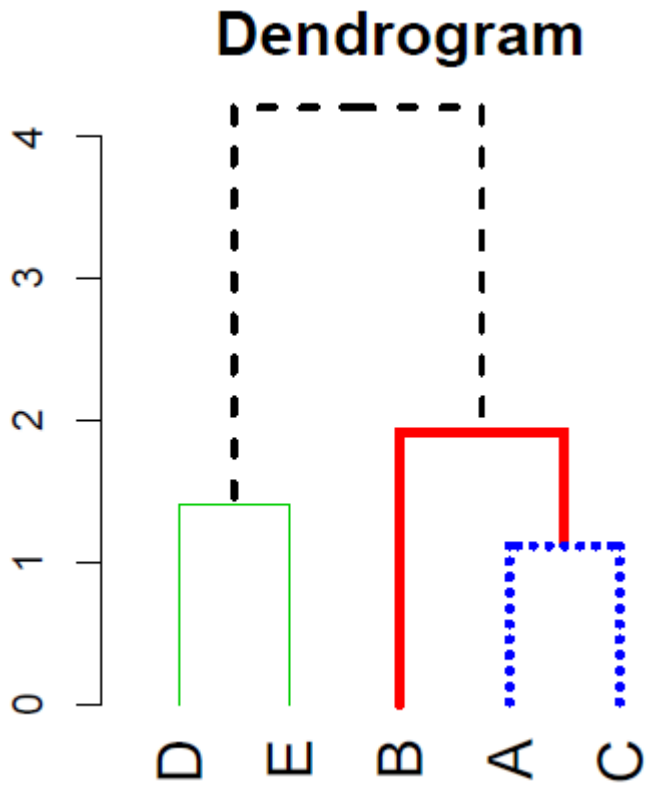




Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# I. Review: Unsupervised Learning | Clustering | How?

- Two clustering methods:
  - K-means clustering (pre-specified k clusters)
  - Hierarchical clustering (dendrogram)




Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# I. Review: Unsupervised Learning | K-means clustering

- Run K-Means multiple times to get the best result, i.e. min($\sum WCV$) = 235.8.

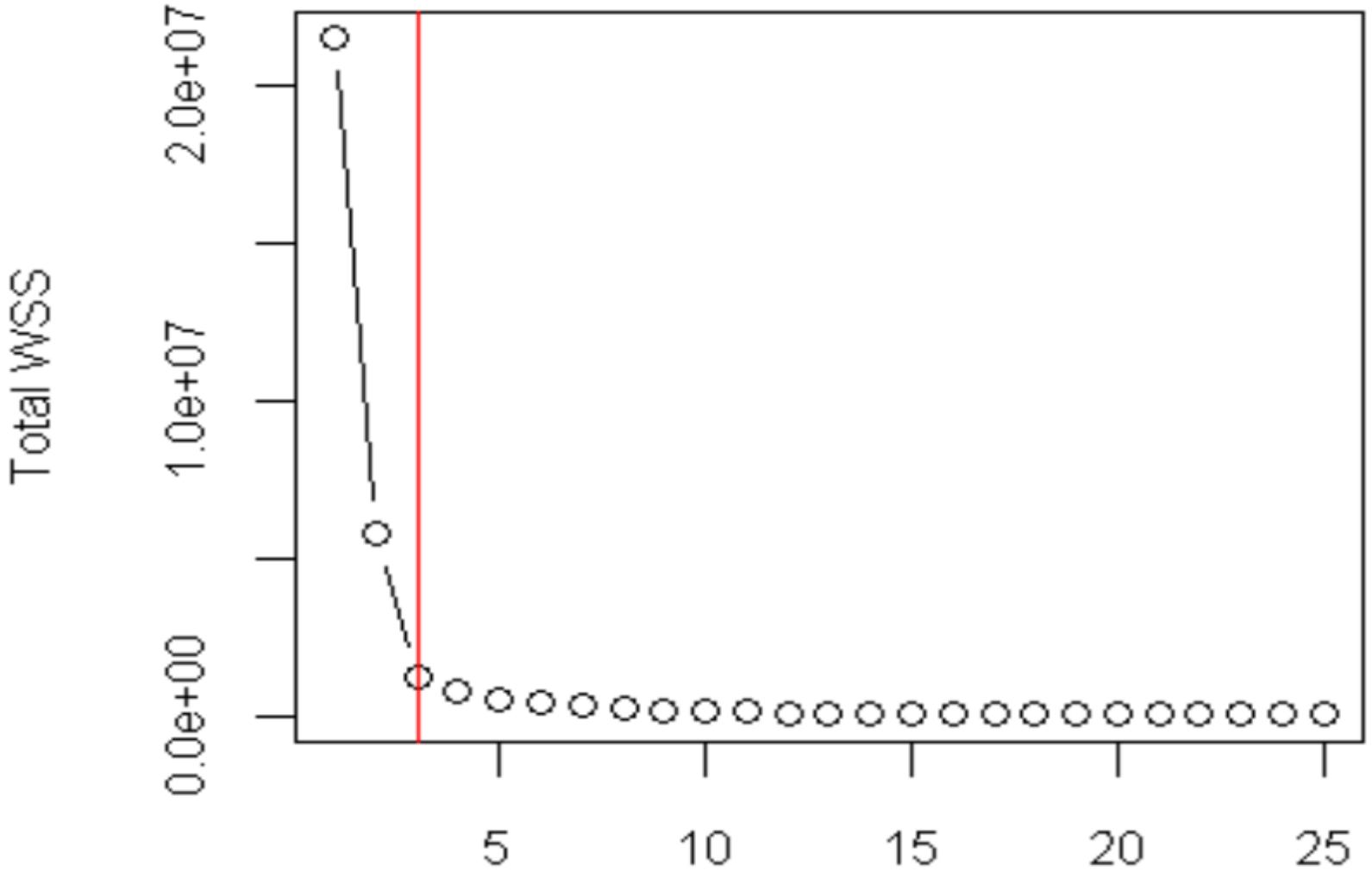
- Within-Cluster Variance:

$$\mathrm{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$


Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# I. Review: Unsupervised Learning | K-means clustering

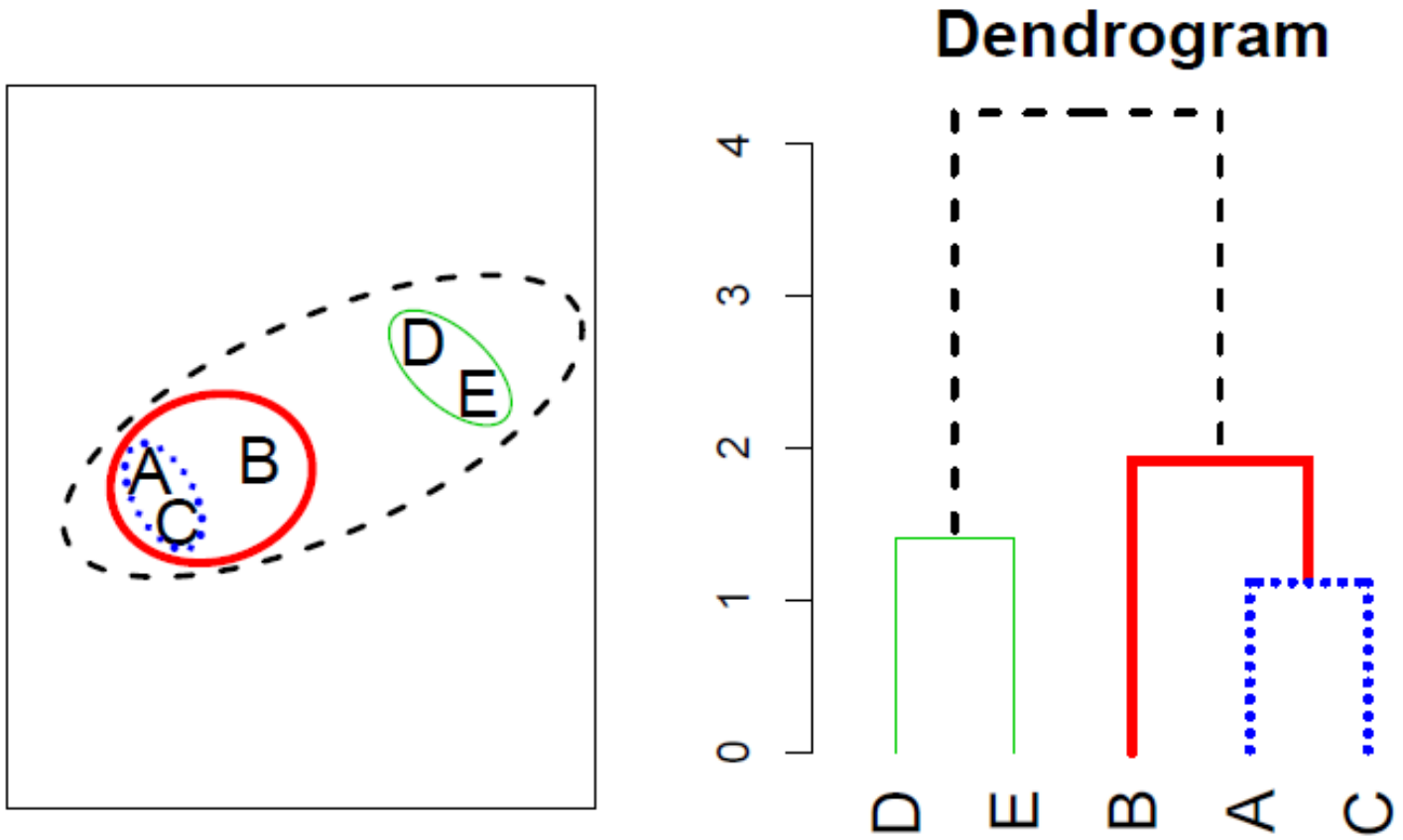
- Scree plot, elbow method.




Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

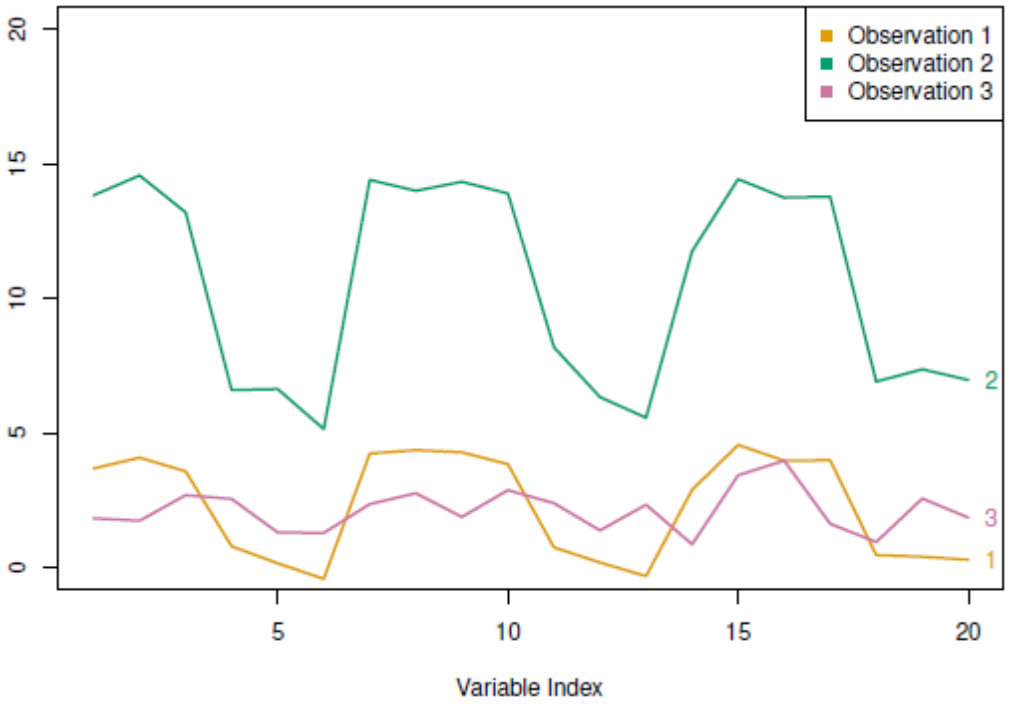
# I. Review: Unsupervised Learning | Hierarchical Clustering

- Which are the two important parameters of hierarchical clustering method?

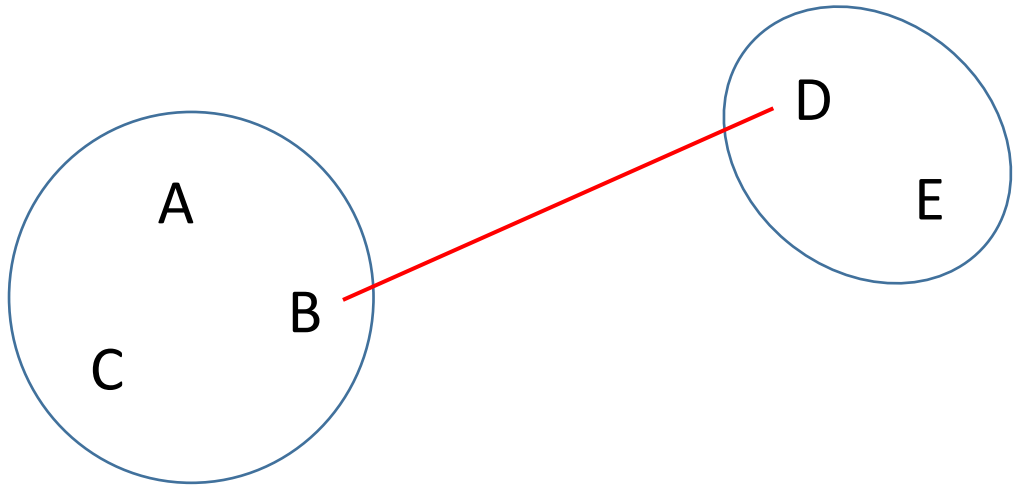



Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# I. Review: Unsupervised Learning | Hierarchical Clustering




**Dissimilarity**




**Linkage**

Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# I. Review: Unsupervised Learning | Hierarchical Clustering

- **Dissimilarity:** How distance is calculated?
  - Euclidean distance
  - Correlation-based distance

- **Linkage:** How to compare the distance of 2 clusters?
  - Complete
  - Single
  - Average
  - Centroid

Source: James, G., Witten, D., Hastie, T., & Tibshirani, R. (2014). *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated.

# Outline

- I. Review: Unsupervised Learning

- II. Neural Networks

- Lab: Neural Networks

# II. Neural Networks | A Brief History

- 1957 – 1960: First idea of perceptron and neuron model
  - **Rosenblatt, F.** (1957). The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.{
    - psychologist, the neurons in human brains, simplified mathematical model of perceptron

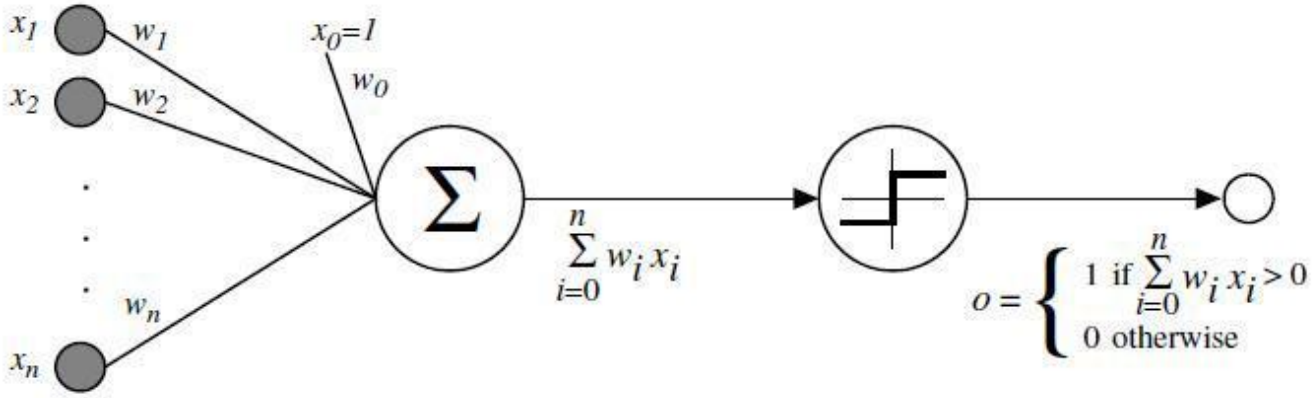



Fig. Diagram of a perceptron

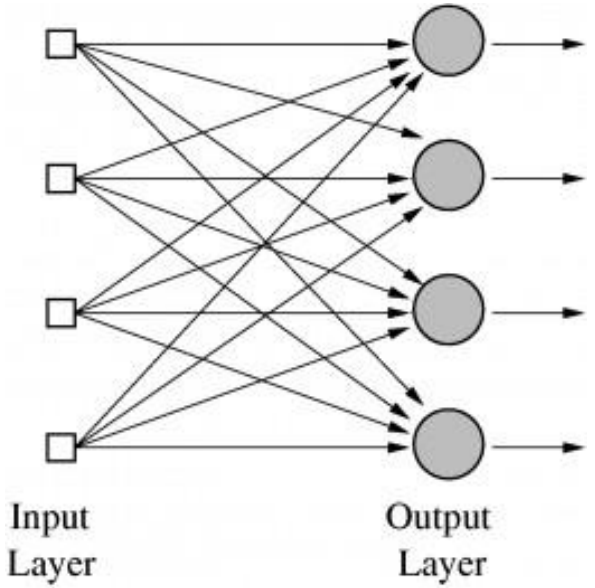



Fig. Single-layer neuron model without hidden layer

Source: Andrey Kurenkov. (2018). A 'Brief' History of Neural Nets and Deep Learning. Link: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

# II. Neural Networks | A Brief History

- 1970 – 1986: Multilayer Neural Networks, backpropagation
  - Rumelhart, D. E., **Hinton, G. E.**, & Williams, R. J. (1985). Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science.
    - **multilayer** neural networks vs. complex learning problems
  - Rumelhart, D. E., **Hinton, G. E.**, and Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323, 533–536
    - well-known paper, explain concisely and clearly **backpropagation** in training neural networks

Fig. Geoffrey Hinton

Source: Andrey Kurenkov. (2018). A 'Brief' History of Neural Nets and Deep Learning. Link: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

# II. Neural Networks | A Brief History

- 1989 – 1998: Developing foundation of Deep Learning
  - **LeCun, Y.**, Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.
    - real-world complex application of backpropagation, handwritten zip code recognition; convolutional neural networks **(CNN)**, subsampling, pooling

Fig. Yann LeCun and the MNIST dataset of handwritten digits

Source: Andrey Kurenkov. (2018). A 'Brief' History of Neural Nets and Deep Learning. Link: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

# II. Neural Networks | A Brief History

- 1989 – 1998: Developing foundation of Deep Learning
  - **Hinton, G. E.** & Zemel, R. S. (1993), Autoencoders, Minimum Description Length and Helmholtz Free Energy., in Jack D. Cowan; Gerald Tesauro & Joshua Alspector, ed., 'NIPS' , Morgan Kaufmann, , pp. 3-10 .
    - **unsupervised** learning with neural networks

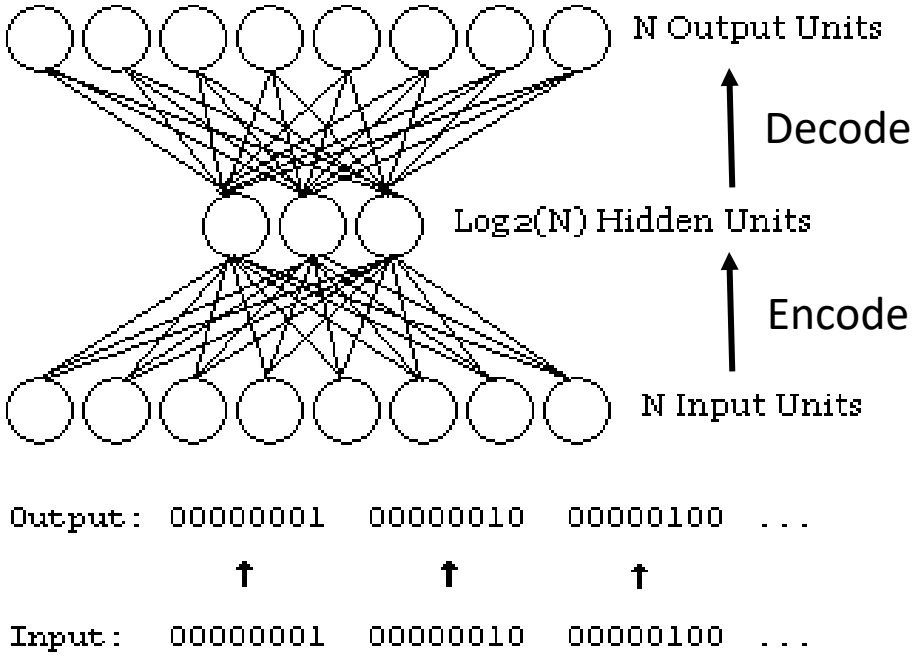



Fig. Autoencoder neural networks

Source: Andrey Kurenkov. (2018). A 'Brief' History of Neural Nets and Deep Learning. Link: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

# II. Neural Networks | A Brief History

- 1989 – 1998: Developing foundation of Deep Learning
  - **Bengio, Y.** (1993). A connectionist approach to speech recognition. In Advances in Pattern Recognition Systems Using Neural Network Technologies (pp. 3-23).
    - recurrent neural networks **(RNN)**
  - **LeCun, Y.**, & **Bengio, Y.** (1995). Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10), 1995.
    - summary many neural networks methods

Fig. Yoshua Bengio

Source: Andrey Kurenkov. (2018). A 'Brief' History of Neural Nets and Deep Learning. Link: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

# II. Neural Networks | A Brief History

- 2000 – present: Era of Deep Learning
  - Raina, R., Madhavan, A., & **Ng, A. Y.** (2009, June). Large-scale deep unsupervised learning using graphics processors. In Proceedings of the 26th annual international conference on machine learning (pp. 873-880).
    - **GPU is 70 times faster than CPU in NN training**, lots of training data, parallel computing
  - **Hinton, G.**, Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine, 29(6), 82-97.
    - Deep neural network, speech recognition


Fig. A PC with many GPU cards to train deep learning model

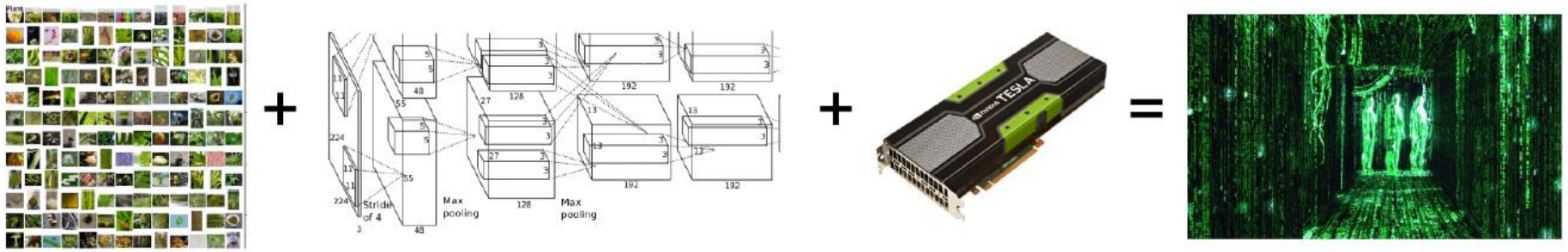
Source: Andrey Kurenkov. (2018). A 'Brief' History of Neural Nets and Deep Learning. Link: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

# II. Neural Networks | A Brief History

- 2000 – present: Era of Deep Learning
  - **Hinton, G. E.**, Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.
    - **dropout**

  - Krizhevsky, A., Sutskever, I., & **Hinton, G. E.** (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
    - state-of-the-art neural networks methods, ImageNet 15.3% error rates

The Deep Learning "Computer Vision Recipe"




Source: Andrey Kurenkov. (2018). A 'Brief' History of Neural Nets and Deep Learning. Link: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

# II. Neural Networks | A Brief History

Turing Award 2018 – The “godfathers of AI”
From left to right: Yann LeCun, Geoffrey Hinton, Yoshua Bengio

Source: James Vincent. (2019). ‘Godfathers of AI’ honored with Turing Award, the Nobel Prize of computing.
Link: https://www.theverge.com/2019/3/27/18280665/ai-godfathers-turing-award-2018-yoshua-bengio-geoffrey-hinton-yann-lecun

# II. Neural Networks | Why?

- Can work with high complex pattern recognition.
- Can learn and do things that we don’t know exactly how to do.
- Building block for advanced machine learning techniques.
- The recent evolution of computer and tools.

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Dimitrios Siganos. Why neural networks?. Link: https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/ds12/article1.html

# II. Neural Networks | Why?

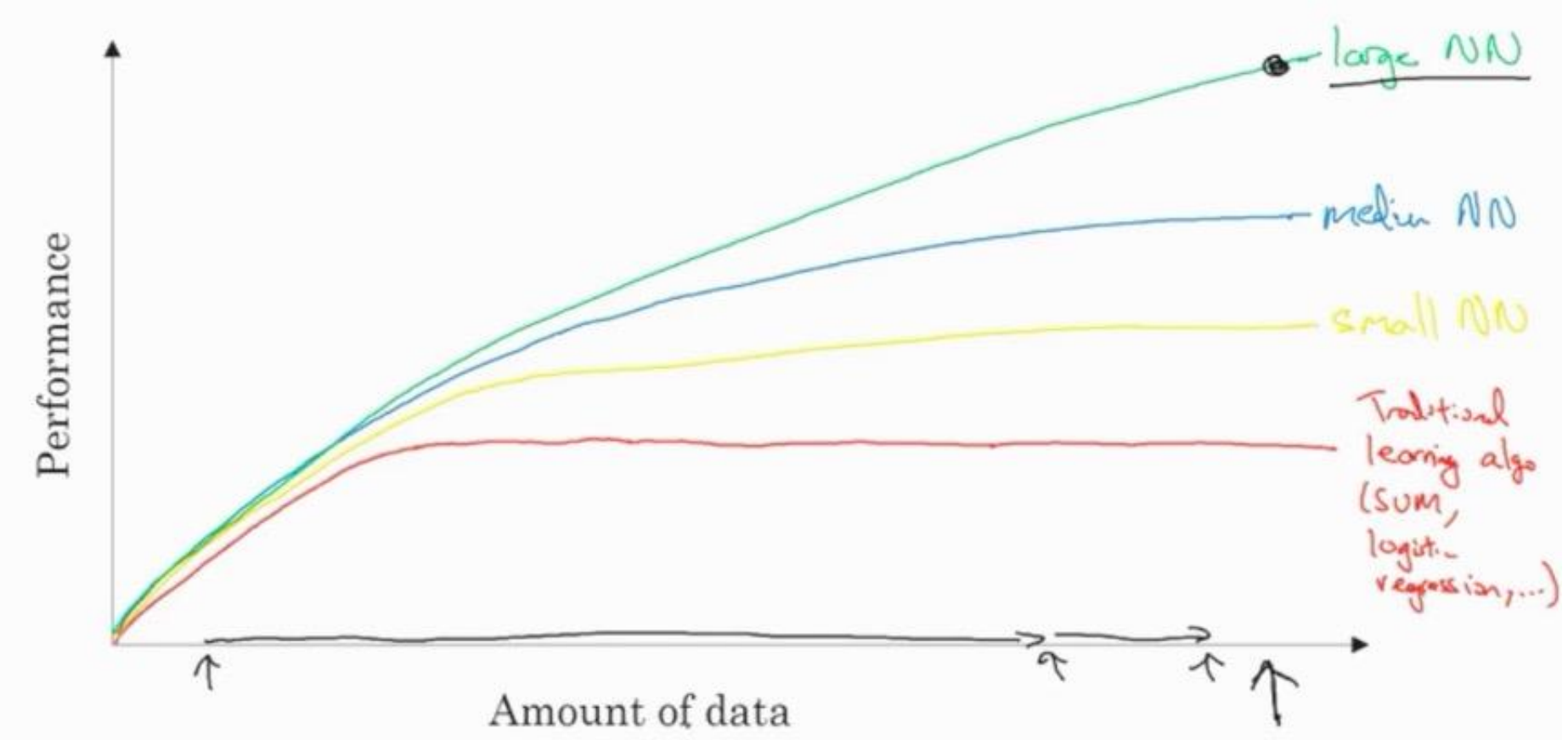
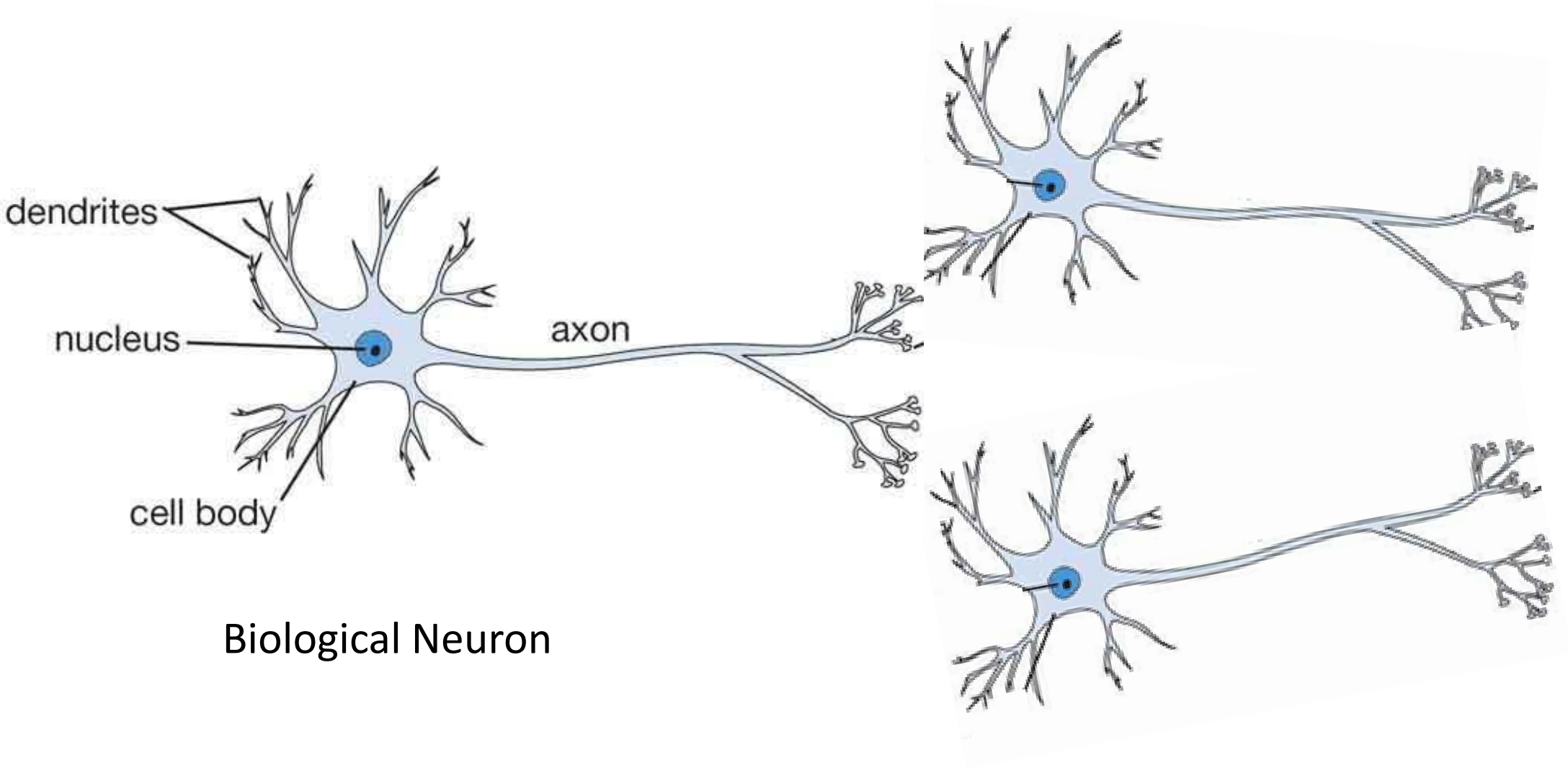



Fig. Scale drives neural networks/deep learning progress

Source: Andrew Ng. Neural Networks and Deep Learning. Link: https://www.coursera.org/learn/neural-networks-deep-learning

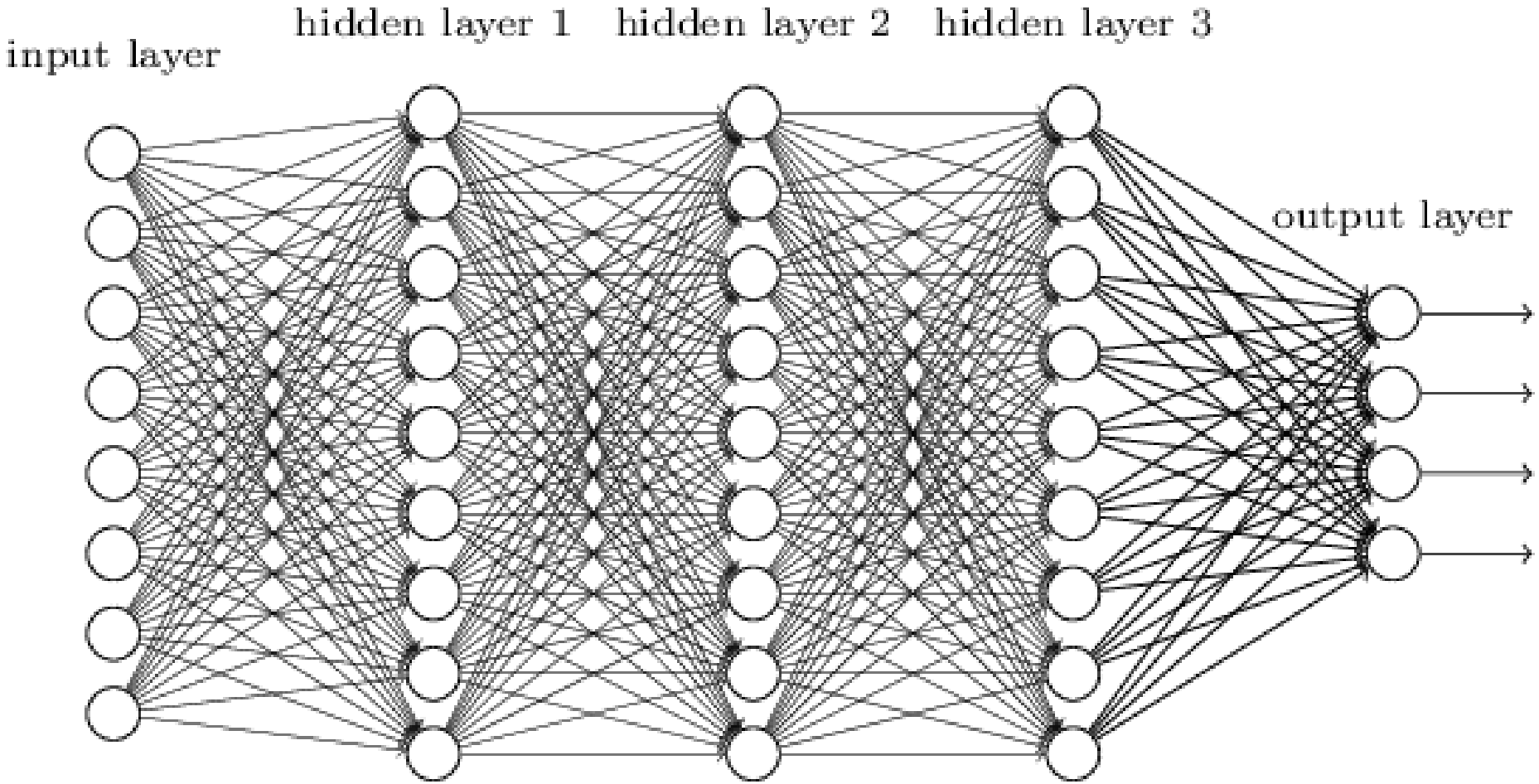
# II. Neural Networks | What?




Biological Neuron

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: CS231n: Convolutional Neural Networks for Visual Recognition. Link: cs231n.github.io

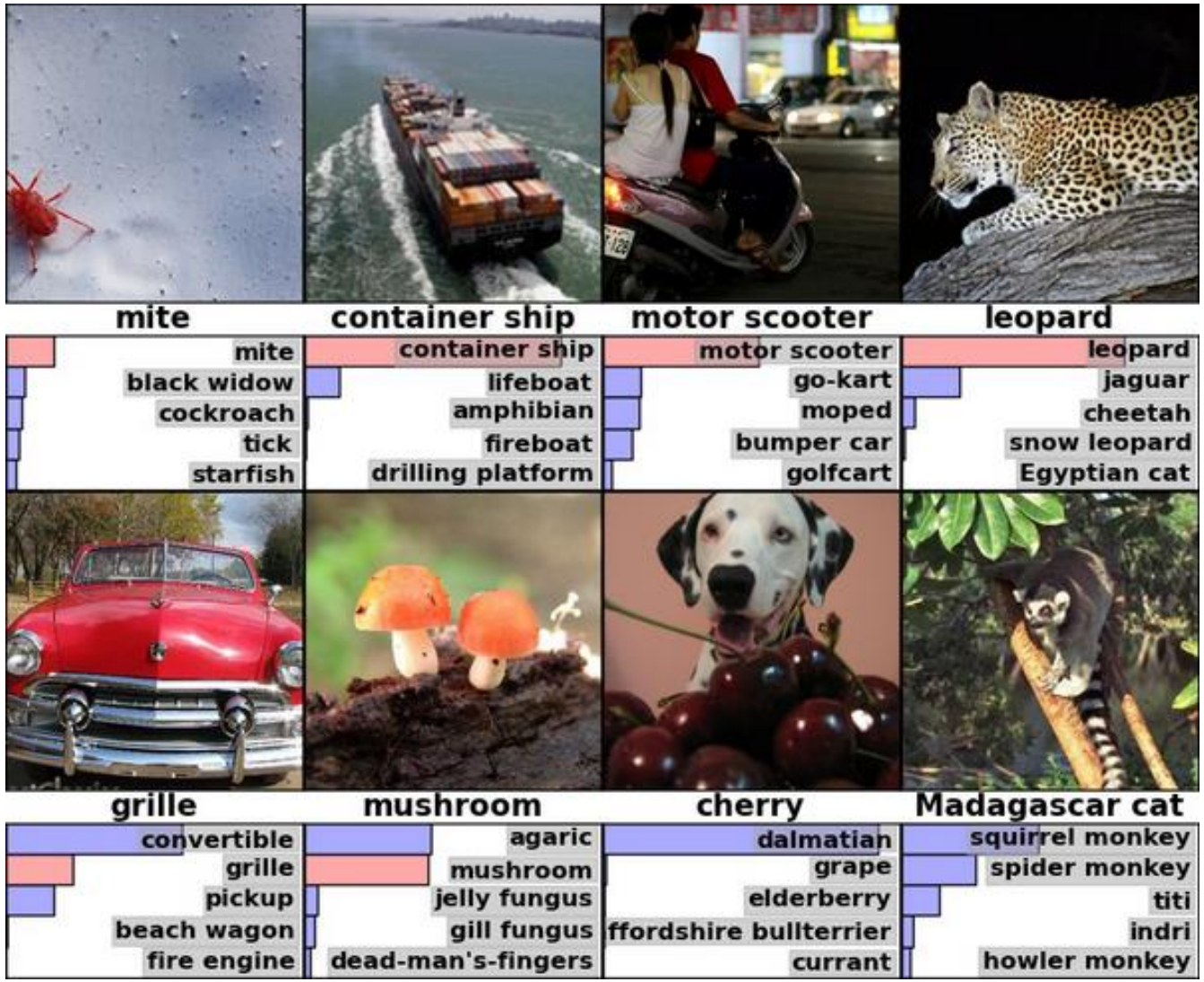
# II. Neural Networks | What?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Creating a Neural Network from Scratch — TensorFlow for Hackers. Link: https://medium.com/@curiousily/tensorflow-for-hackers-part-iv-neural-network-from-scratch-1a4f504dfa8

# II. Neural Networks | What?

- Neural Network applications:
  - Character Recognition
  - Image Compression
  - Stock Market Prediction
  - Medicine, Electronic Nose
  - Security, loan Applications
  - Etc.



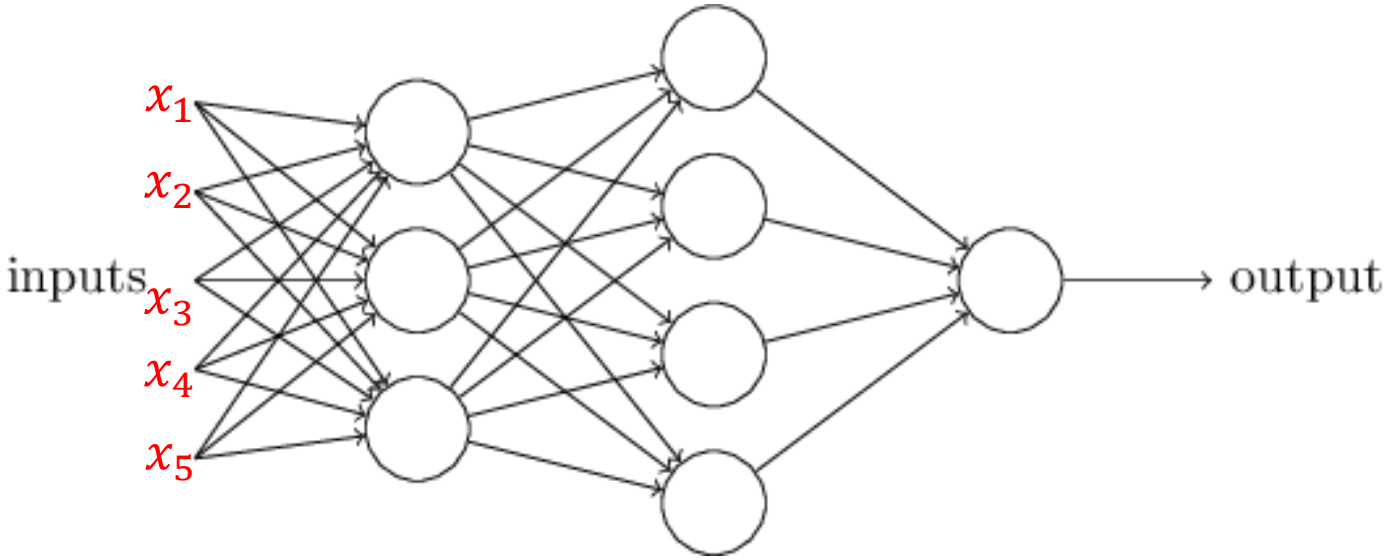

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Large Scale Visual Recognition Challenge 2010 (ILSVRC2010). Link: http://image-net.org/challenges/LSVRC/2010/

# II. Neural Networks | What?


Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Large Scale Visual Recognition Challenge 2010 (ILSVRC2010). Link: http://image-net.org/challenges/LSVRC/2010/

# II. Neural Networks | Building blocks

- Neuron, activation function
  - Perceptrons
  - Sigmoid neurons
- Neural networks architecture
  - Single hidden layer
  - Multiple hidden layers
  - Convolutional Neural Network (CNN)
- Calculation
  - Gradient descent
  - Backpropagation



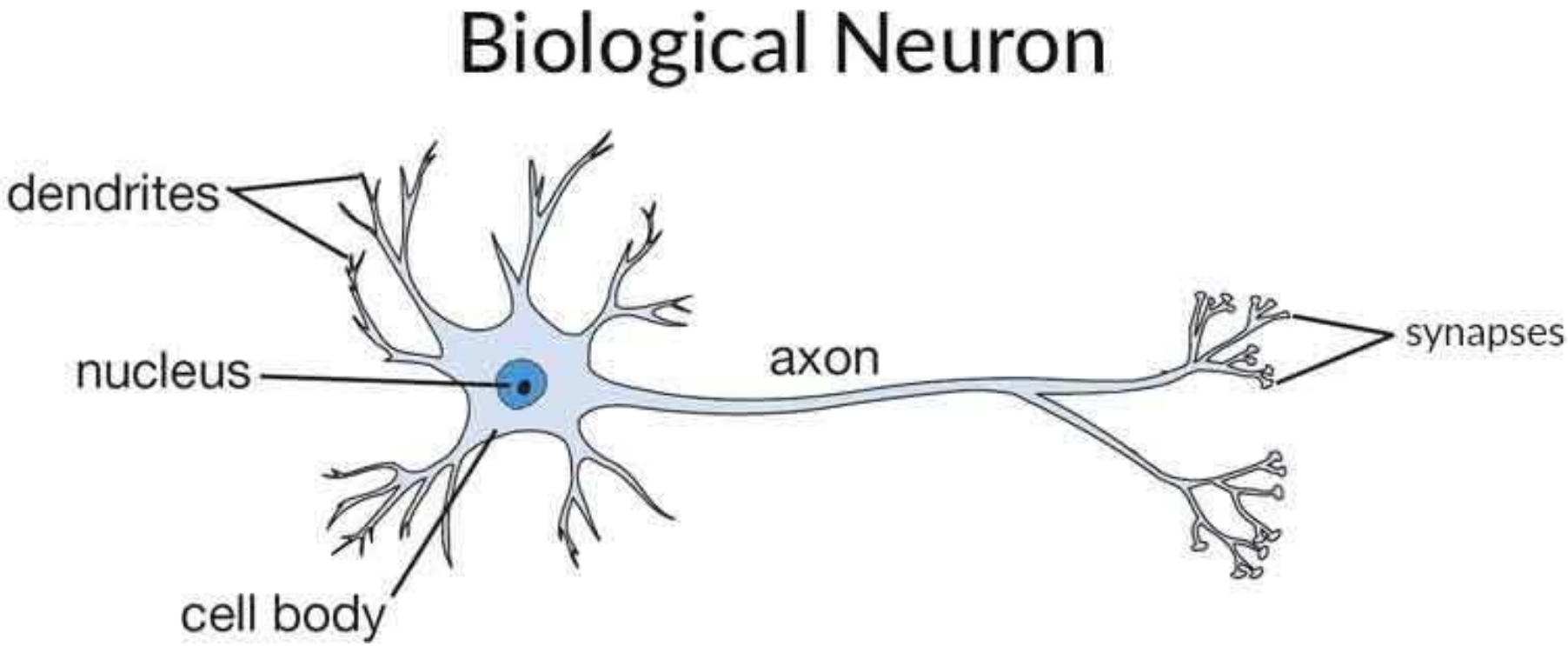

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Building blocks | New annotations

| **Statistical Learning**<br>**(Statistician community)** | **Machine Learning**<br>**(Computer science community)** |
|---|---|
| Estimated function:<br>$\hat{f}(X) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_p X_p$ | Estimated function (hypothesis):<br>$h_\theta(X) = \theta_0 + \theta_1 X_1 + \cdots + \theta_p X_p$ |
| Function parameter:<br>$\beta\ (beta)$ | Function parameter (gradient descent):<br>$\theta\ (theta)$<br><br>Function parameter (neural network):<br>$f(X) = b + \sum_j w_j x_j$<br>$w\ (weight)$<br>$b\ (bias)$ |

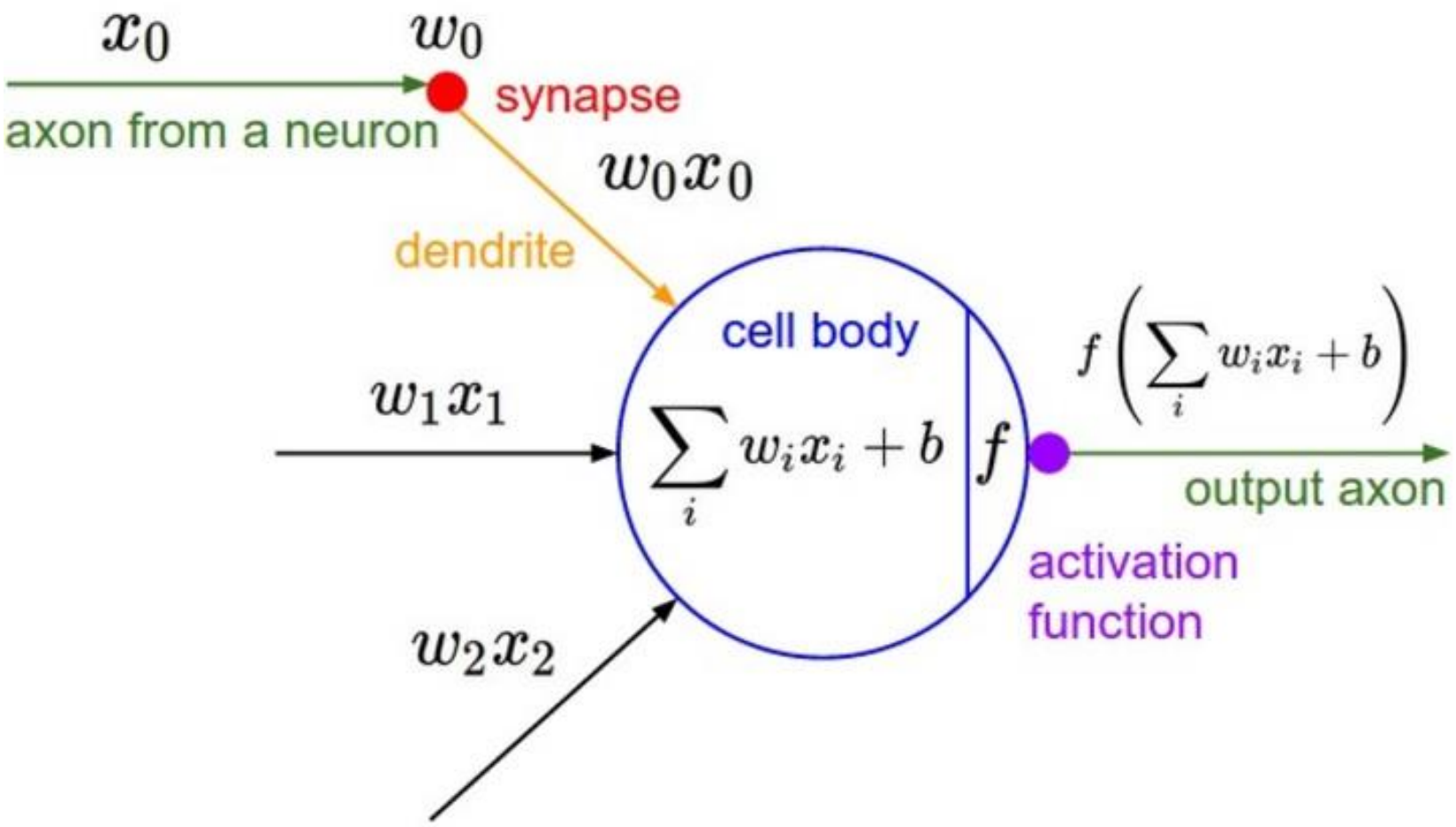
Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Neuron



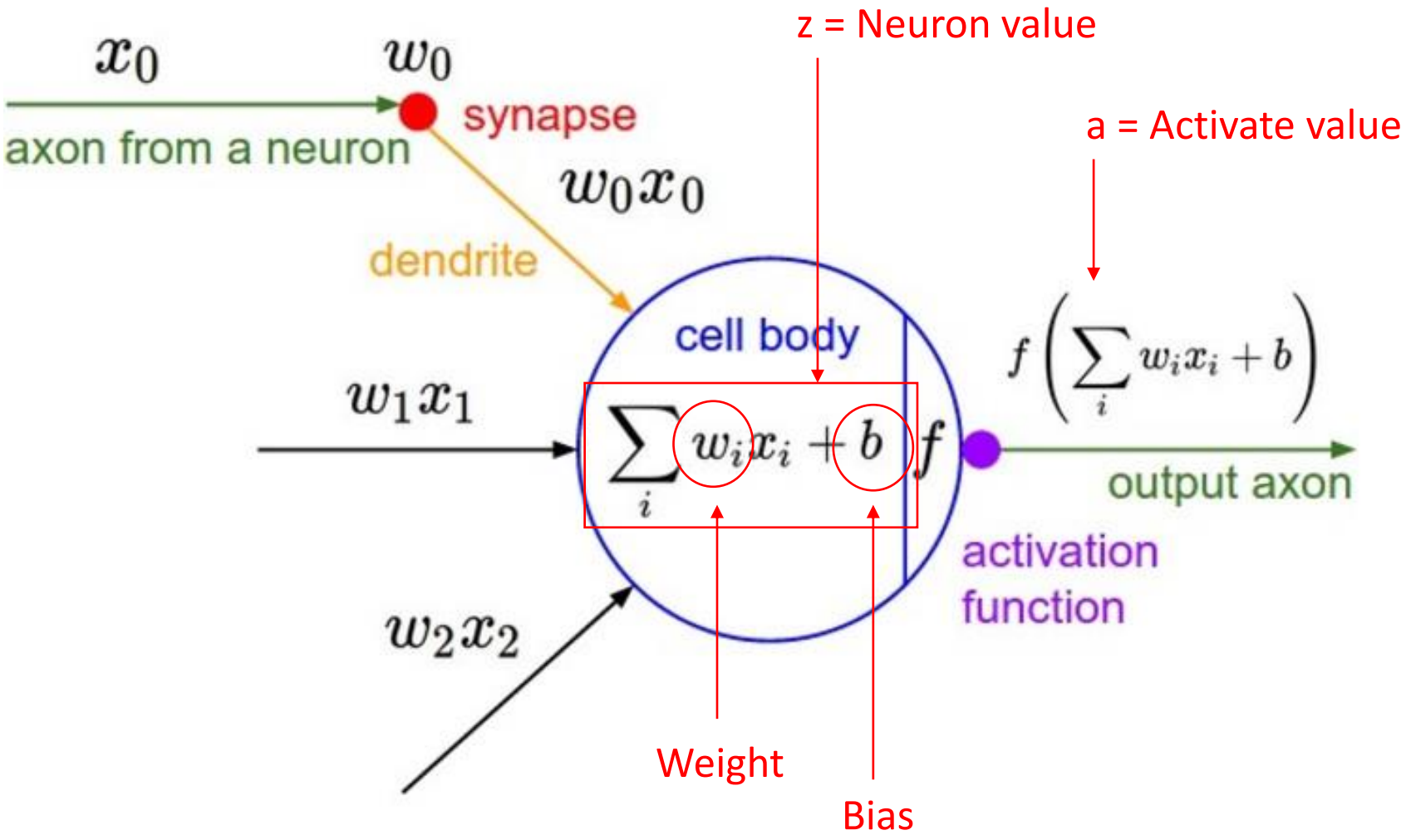

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: CS231n: Convolutional Neural Networks for Visual Recognition. Link: cs231n.github.io

# II. Neural Networks | Neuron




Source: Jeremy Jordan. (2017). *Neural networks: representation.* Link: https://www.jeremyjordan.me/intro-to-neural-networks/

# II. Neural Networks | Neuron



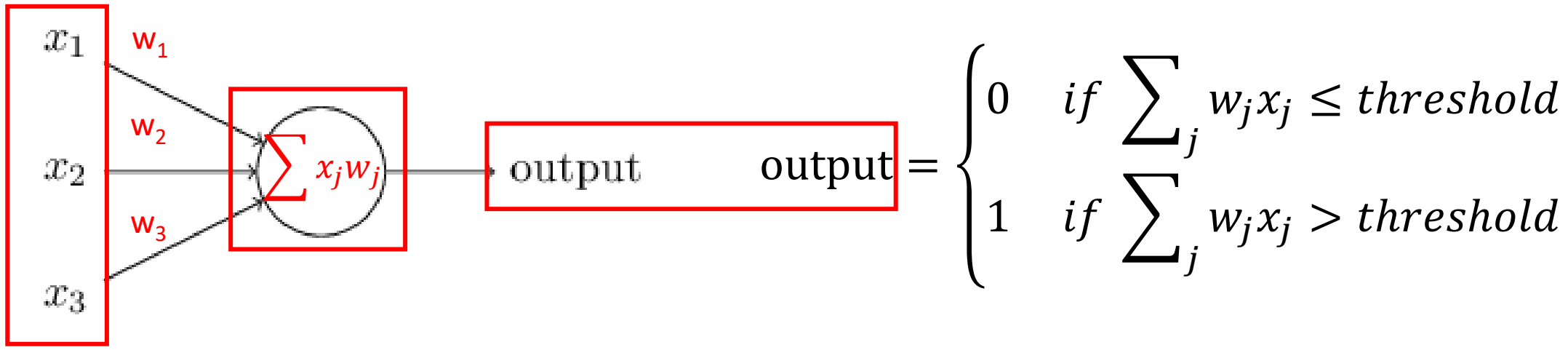

Source: Jeremy Jordan. (2017). *Neural networks: representation.* Link: https://www.jeremyjordan.me/intro-to-neural-networks/

# II. Neural Networks | Neuron | Perceptrons

- Developed in 1950s and 1960s by Frank Rosenblatt (psychologist).

- Simulating the biological neuron, fire or not fire.

- Making decisions by weighting up evidence.




$$\text{output} = \begin{cases} 0 & if \sum_j w_j x_j \leq threshold \\ 1 & if \sum_j w_j x_j > threshold \end{cases}$$

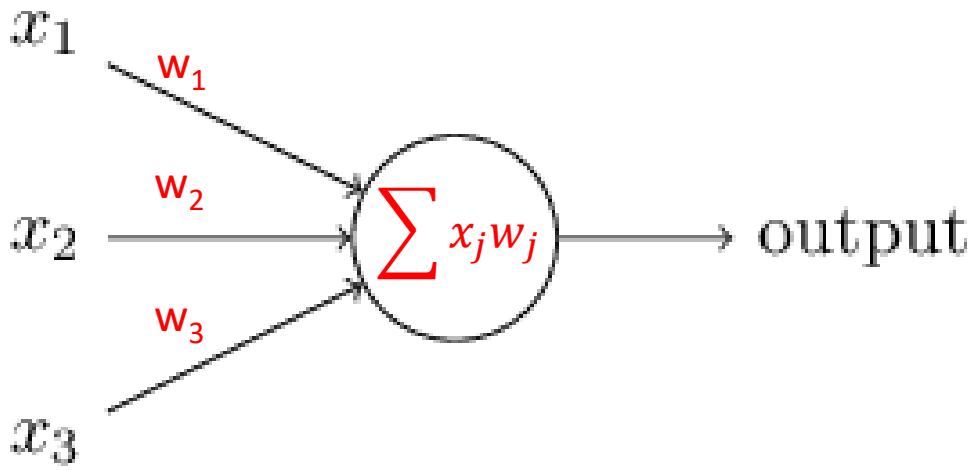
Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Neuron | Perceptrons

**Example 1:** Making decision to go to cheese festival in the weekend.

- $x_1$: Is the weather good? [0, 1]
- $x_2$: Will your friends join you? [0, 1]
- $x_3$: Is the festival close to your place? [0, 1]

- Build your decision models using $w_1$, $w_2$, $w_3$ and the threshold.



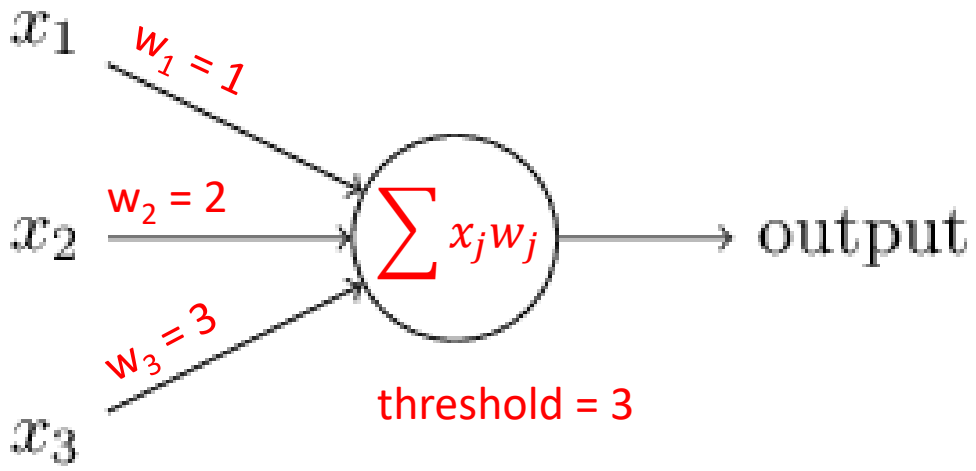

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Neuron | Perceptrons

**Example 1:** Making decision to go to cheese festival in the weekend.

- $x_1$: Is the weather good? [0, 1]
- $x_2$: Will your friends join you? [0, 1]
- $x_3$: Is the festival close to your place? [0, 1]

- Case 1: Good weather + No friend joining + Festival is close ➔ Decision?



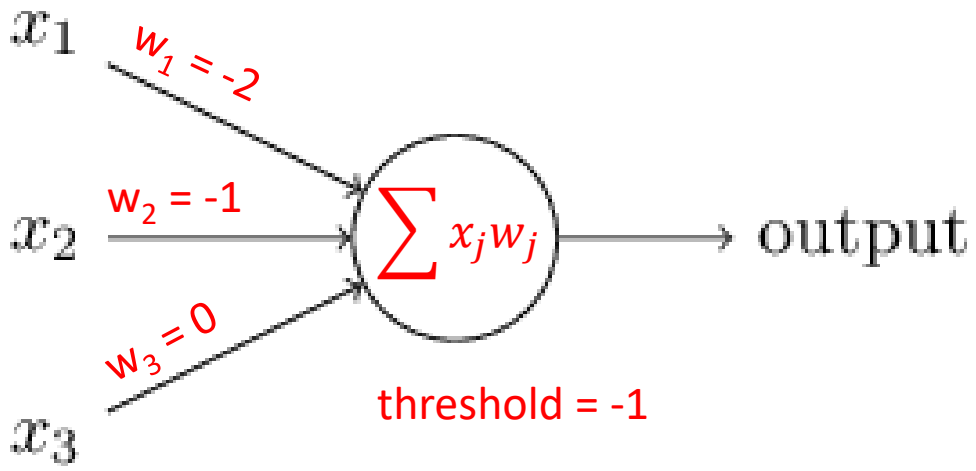

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Neuron | Perceptrons

**Example 1:** Making decision to go to cheese festival in the weekend.

- $x_1$: Is the weather good? [0, 1]
- $x_2$: Will your friends join you? [0, 1]
- $x_3$: Is the festival close to your place? [0, 1]

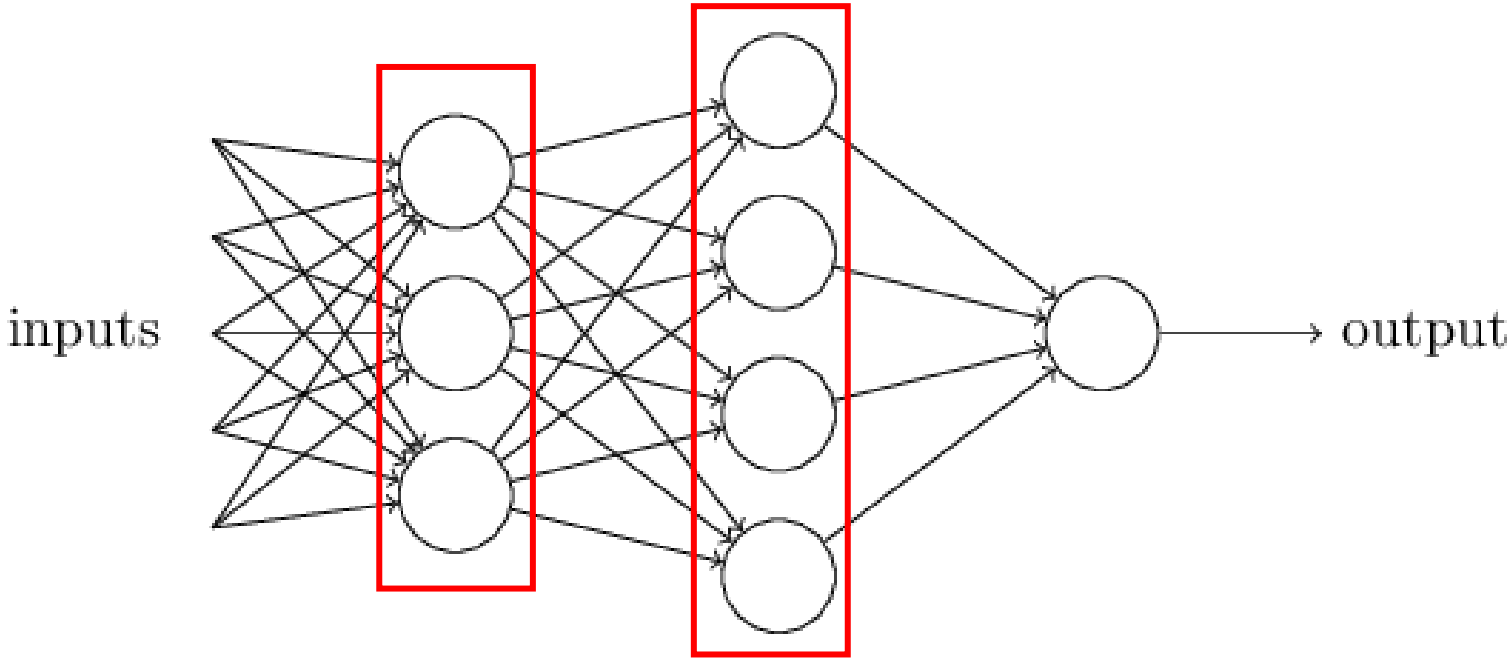
- Case 2: Rainy day + No friend joining + Festival is very far ➔ Decision?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Neuron | Perceptrons

- The first layer of perceptrons: weighting the input evidence.
- The second layer of perceptrons: weighting the results of the first layer.




- **Question:** How many output per perceptron?

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Neuron | Perceptrons

- Rewrite the perceptron rule using dot product and bias:

$$\text{output} = \begin{cases} 0 & if\ w.x + b \leq 0 \\ 1 & if\ w.x + b > 0 \end{cases}$$

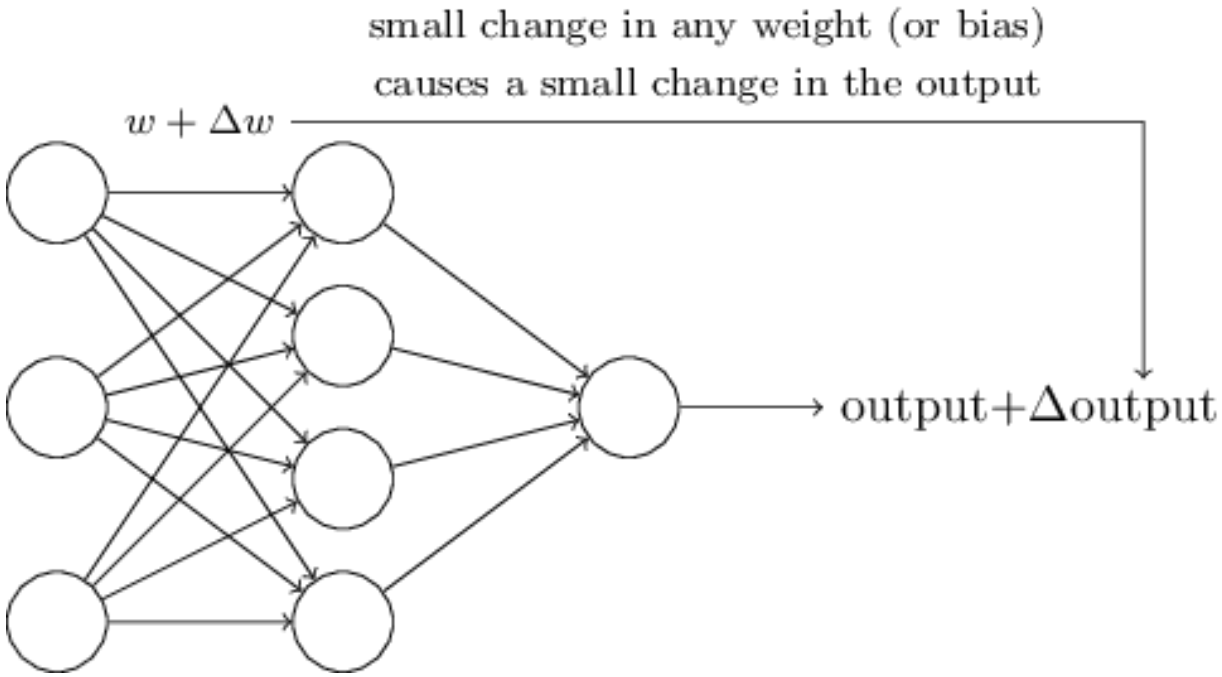
- The dot product: $w.x \equiv \sum_j w_j x_j$

- The bias: $b \equiv -threshold$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Neuron | Perceptrons

- First idea to train the neural networks:




- **Question:** What is the problem with neural networks using perceptions?

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Neuron | Sigmoid neurons

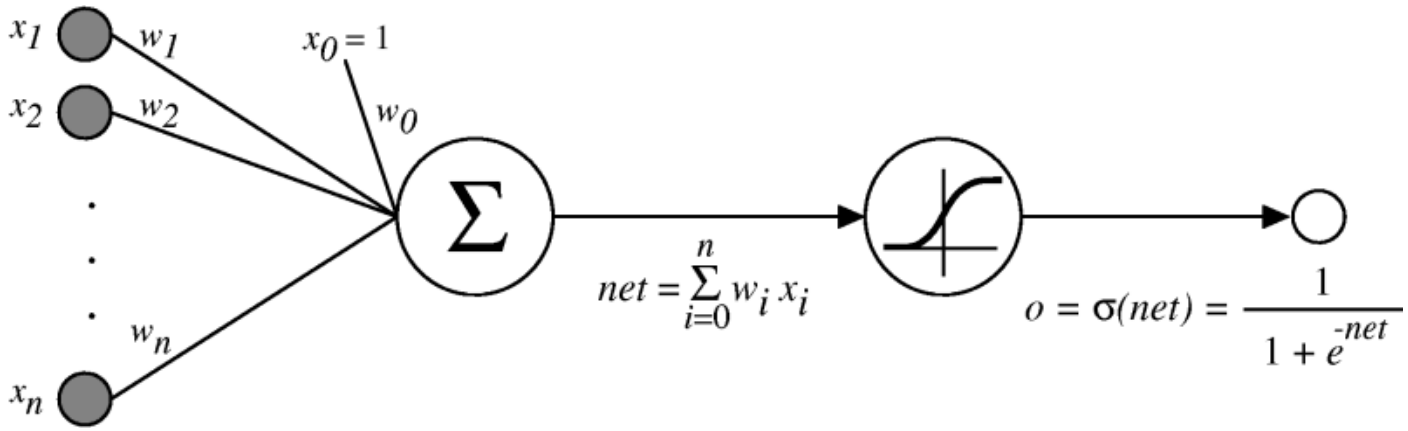
- Similar to perceptrons.

- Small changes in the weights and bias cause only small change in the output.

- The sigmoid function: $\sigma(z) \equiv \frac{1}{1+e^{-z}}$

- The output of sigmoid neuron:

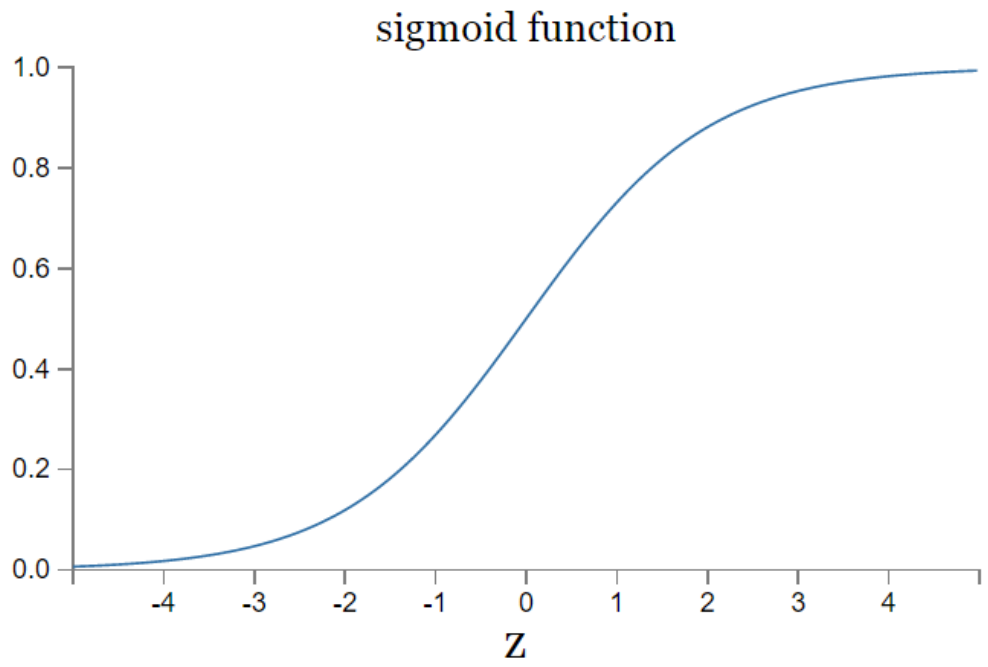
$$\text{output} = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$

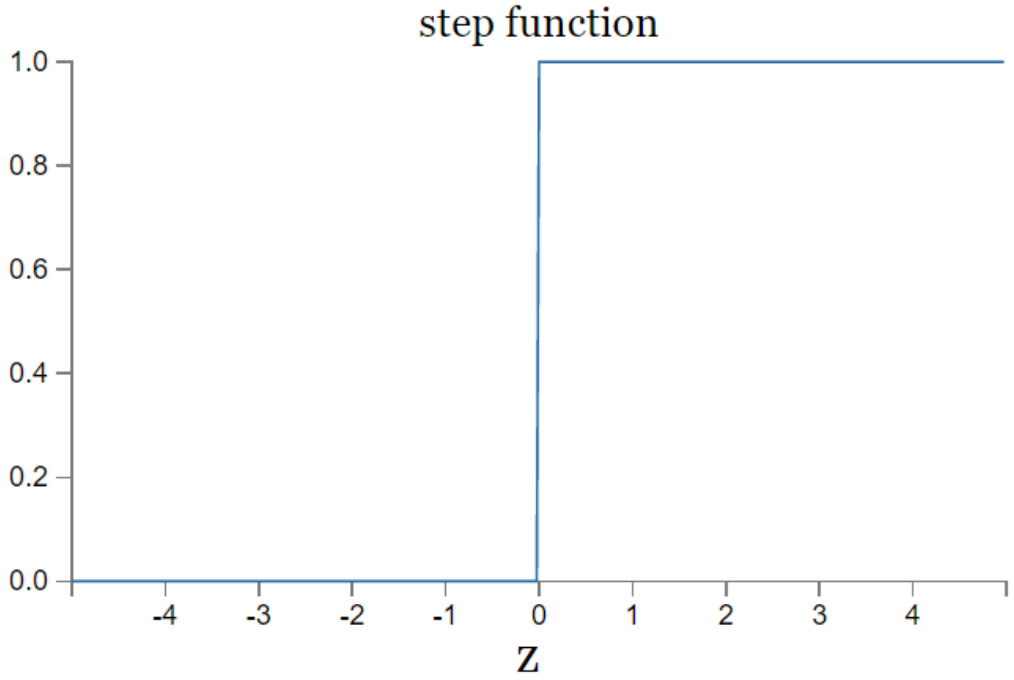



Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Artificial Neural Networks. Link: http://jmvidal.cse.sc.edu/talks/ann/allslides.html

# II. Neural Networks | Neuron | Sigmoid neurons

- Sigmoid activation function is a smoothed version of step function.




$$\text{output} = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)}$$

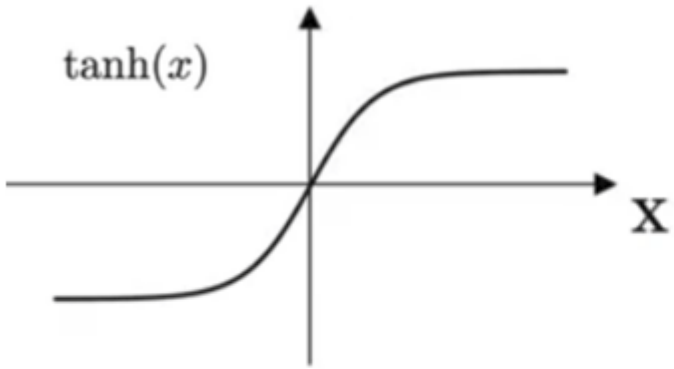
$$\text{output} = \begin{cases} 0 & if\ w.x + b \leq 0 \\ 1 & if\ w.x + b > 0 \end{cases}$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

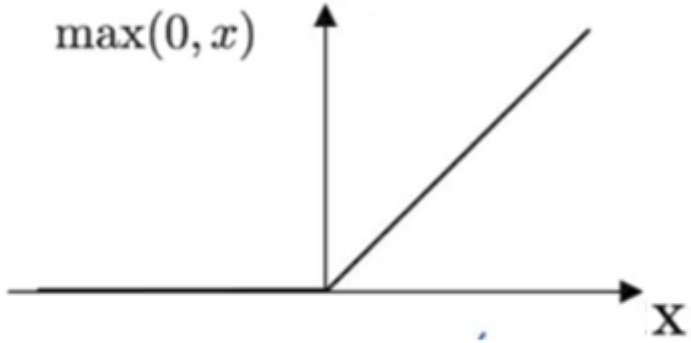
# II. Neural Networks | Neuron | Sigmoid neurons

- Common activation functions:

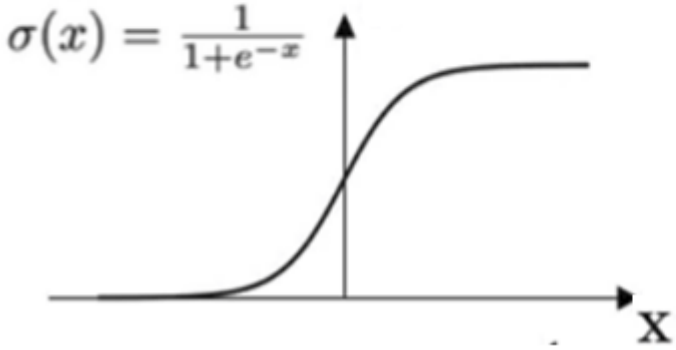
**Hyper Tangent Function**



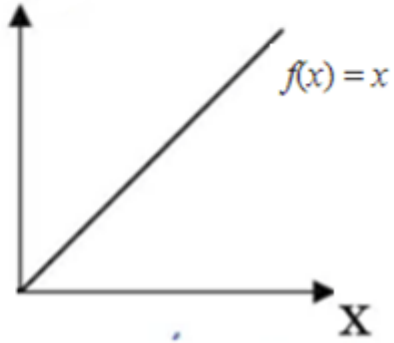

**ReLU Function**




**Sigmoid Function**

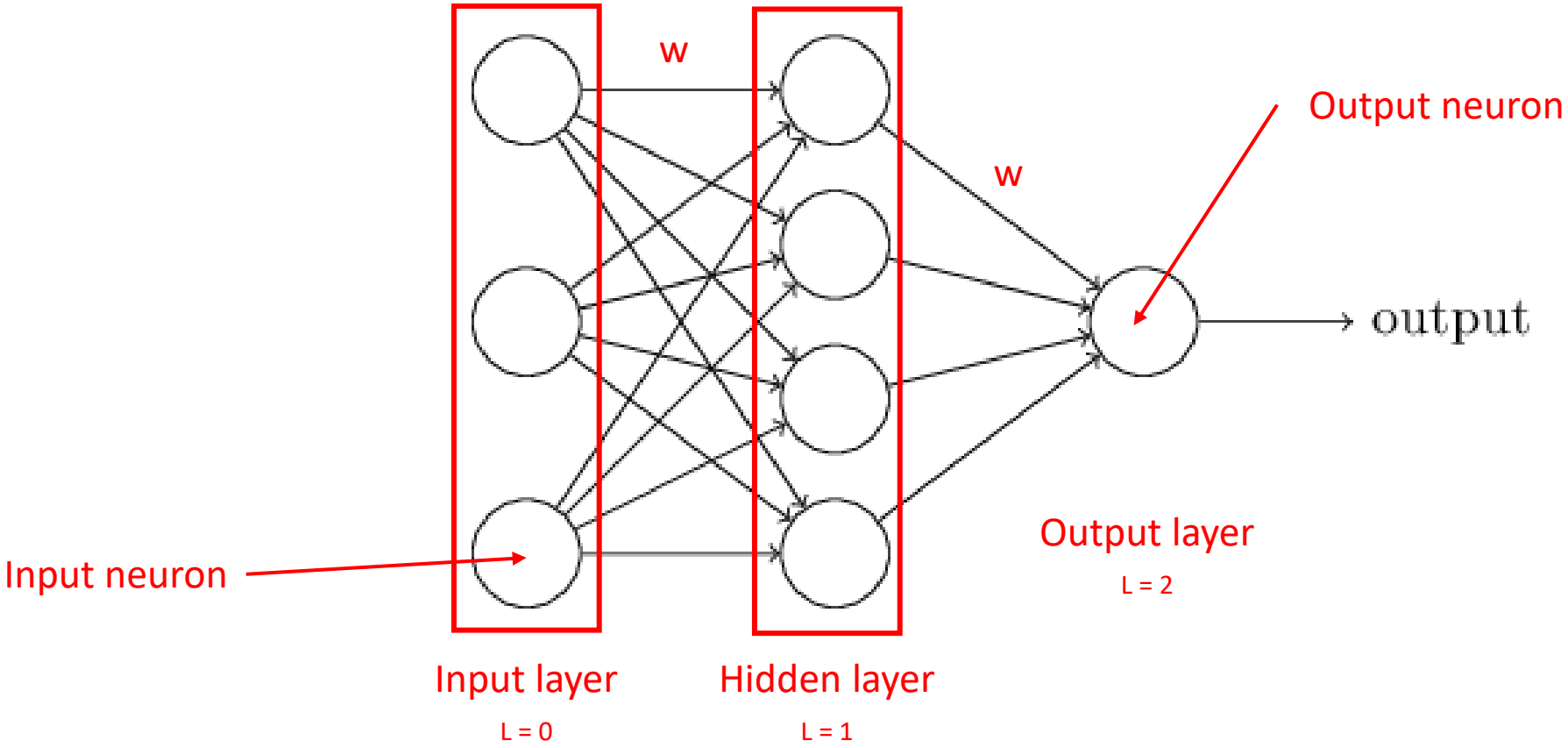



**Identity Function**



# II. Neural Networks | The architecture

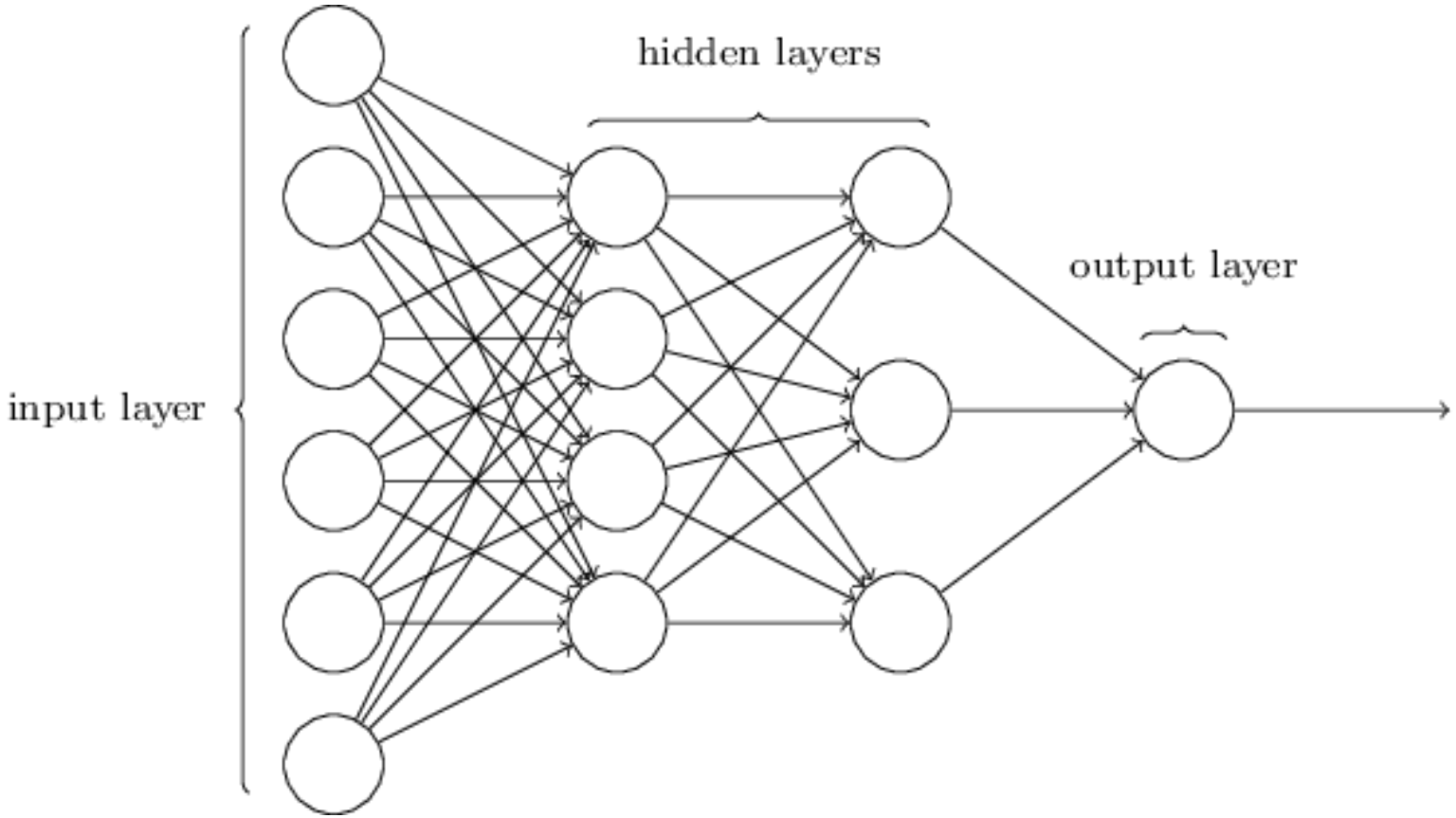
- The architecture of a single hidden layer neuron network.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | The architecture

- The architecture of a multiple hidden layers neuron network.

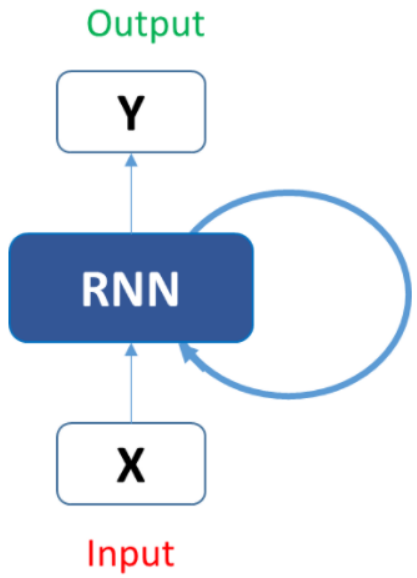



Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | The architecture

- **Feedforward neural networks:** There is no loops in the network, information is always fed forward.

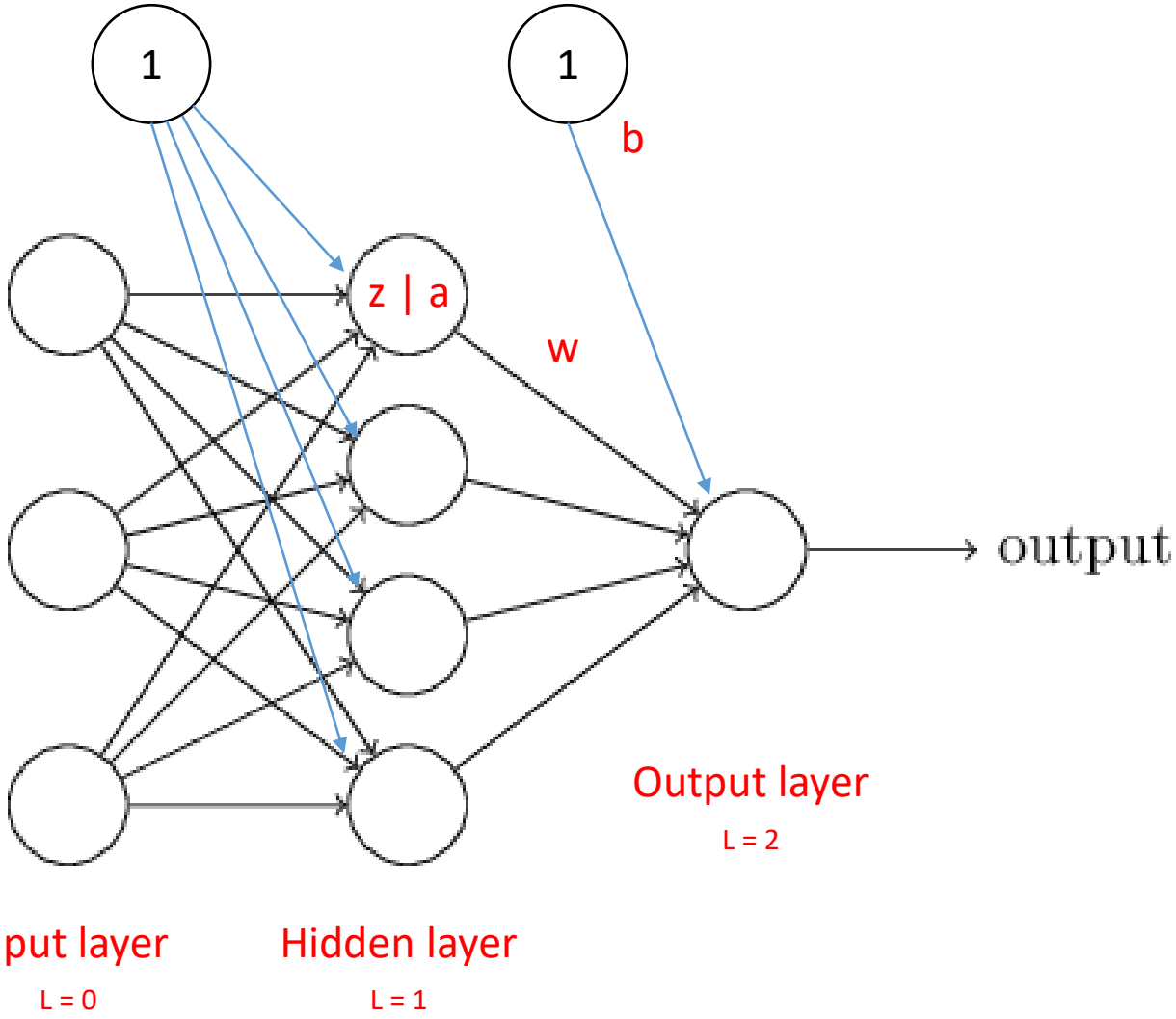
- **Recurrent neural networks (RNN):** Feedback loops are possible. Then, neurons can fire for some limited duration of time before becoming temporarily quiet.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Fundamentals of Deep Learning – Introduction to Recurrent Neural Networks. Link: https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

# II. Neural Networks | The architecture | Parameters

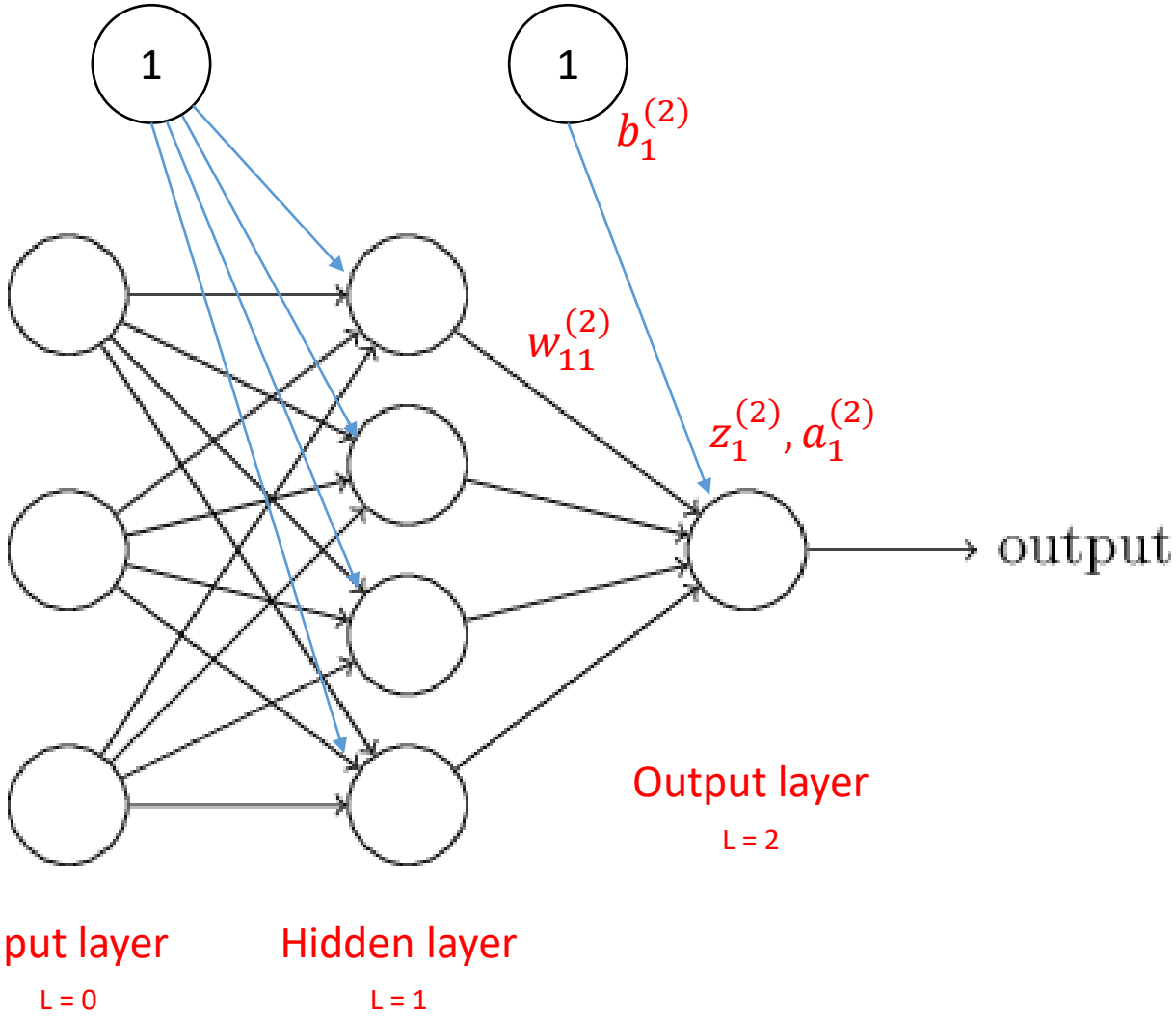
- The layer: $L = 0, 1, 2, \ldots$
- The weight: $w_{ij}^{(L)}$
  - L : the layer
  - i : the position (index) of the neuron
  - j : the position (index) of the weight
- The bias: $b_i^{(L)}$
  - L : the layer
  - i : the position (index) of the neuron
- The value of the neuron: $z_i^{(L)}$
  - L : the layer
  - i : the position (index) of the neuron
- The activate value: $a_i^{(L)}$
  - L : the layer
  - i : the position (index) of the neuron




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | The architecture | Parameters

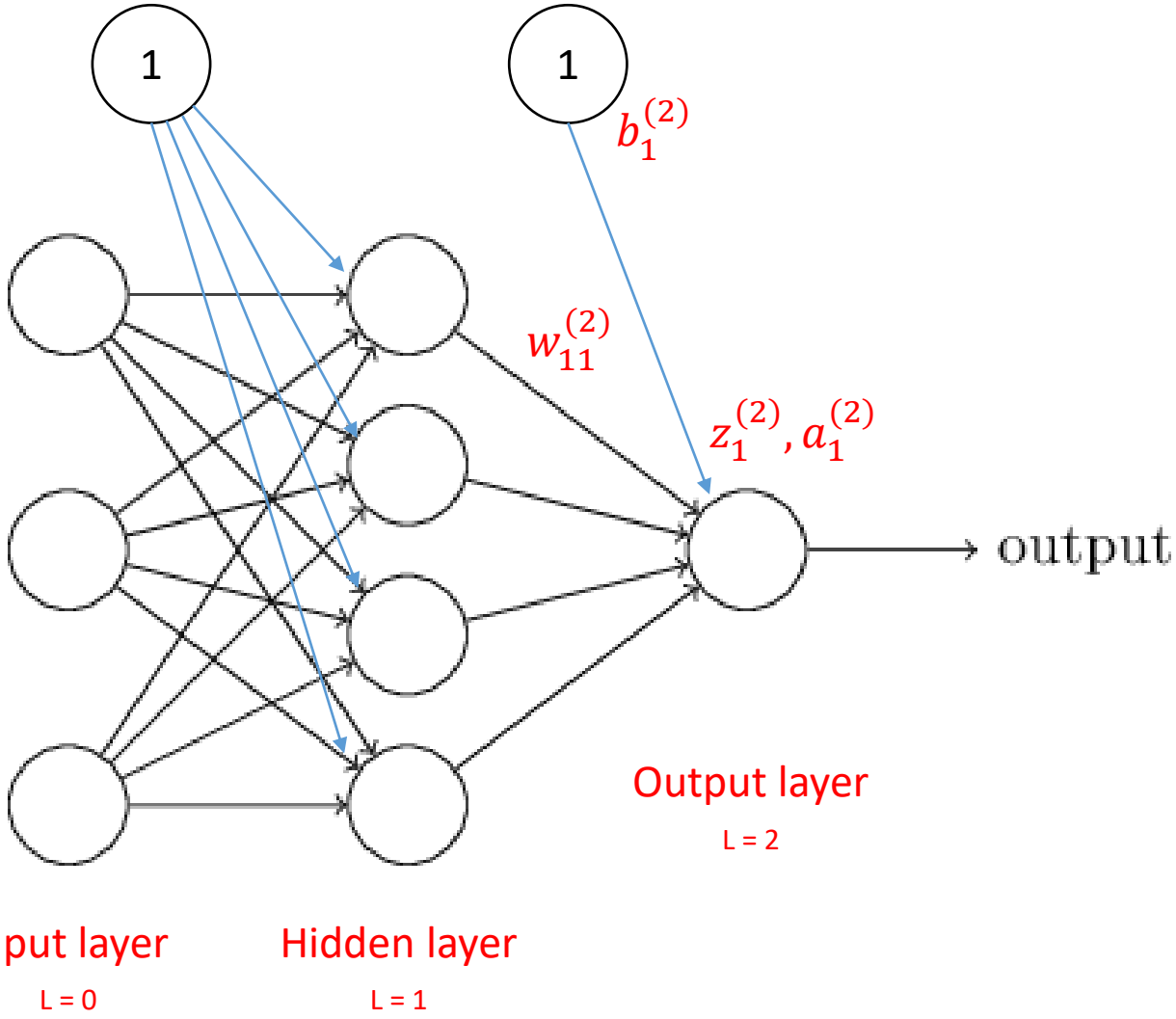
- **Example 2:** Parameters annotations for layer 0, 1, 2.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | The architecture | Parameters

- **Example 2:** Parameters annotations for layer 0, 1, 2.




- **Question:** What are the parameters of Neural Networks model to be estimated ? How many of them ?

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

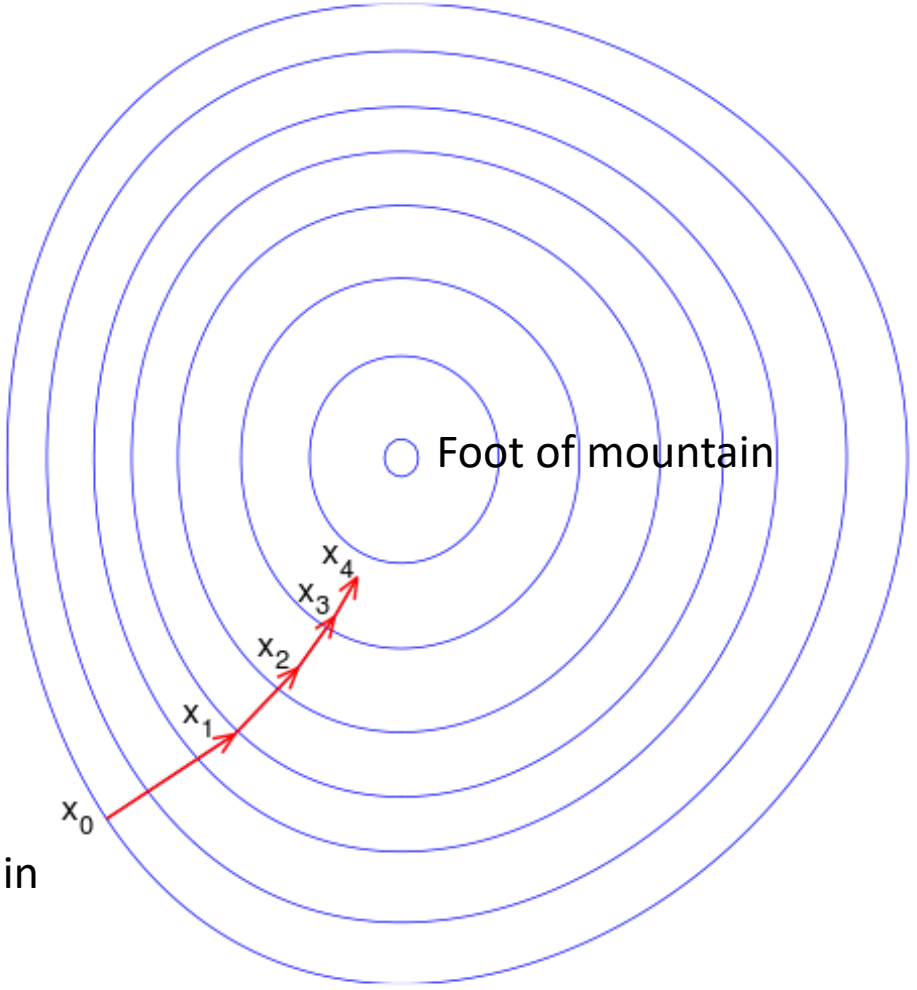
- A hypothetical scenario:
  - You are stuck in the mountain and are trying to get down to the foot of mountain (global minimum).
  - There is heavy fog everywhere, the path down the mountain is no visible.
  - Solution ?

Foggy mountain

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Gradient descent. Link: https://en.wikipedia.org/wiki/Gradient_descent

# II. Neural Networks | Calculation | Gradient descent

- A hypothetical scenario:
  - You are stuck in the mountain and are trying to get down to the foot of mountain (global minimum).
  - There is heavy fog everywhere, the path down the mountain is no visible.
  - Solution:
    - Use the local information
    - Find the steepness of the current position
    - Go downhill at the steepest descent




Top of the mountain

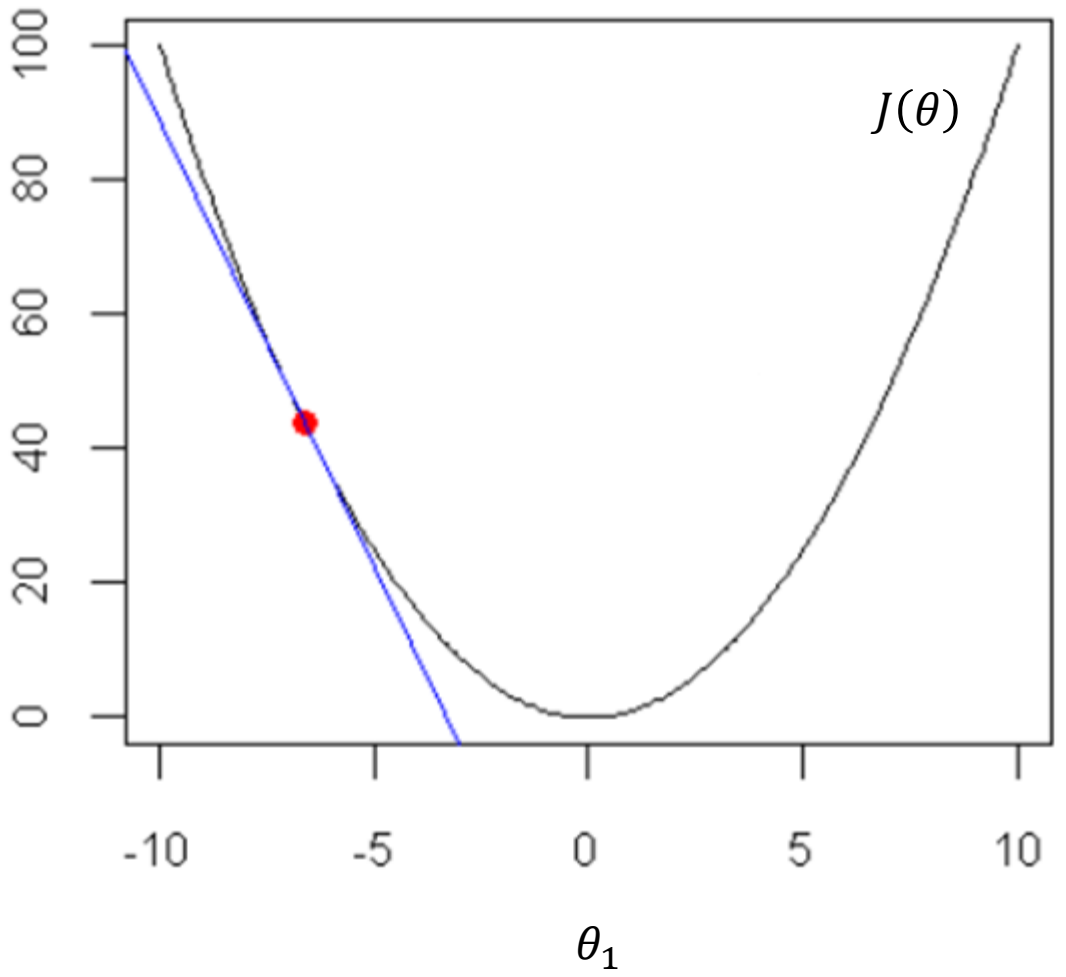
Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Gradient descent. Link: https://en.wikipedia.org/wiki/Gradient_descent

# II. Neural Networks | Calculation | Gradient descent

- The target function to be minimized: $J(\theta)$
  - The mountain terrain
  - Differentiable function

- The input of the target function: $\theta_j$
  - The coordinate grid of the mountain terrain
  - $\theta_1 \approx X_1$
  - $\theta_2 \approx X_2$
  - …

- Partial derivative of $J(\theta)$
  - Steepest direction to go downhill
  - Gradient (partial derivative, sloop)




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

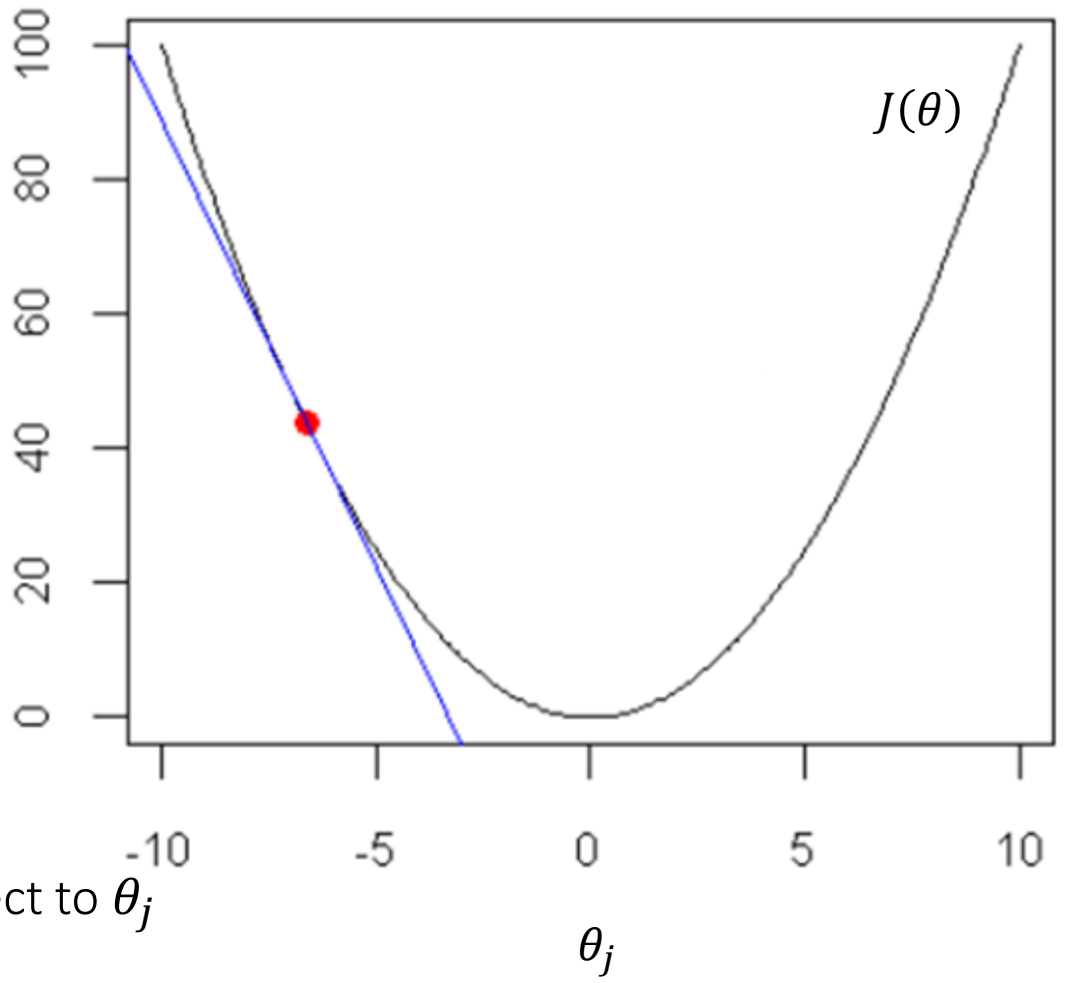
# II. Neural Networks | Calculation | Gradient descent

- Gradient descent in a nutshell:

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}




In which:

- $J(\theta)$ : the target function to be minimized
- $\theta_j$ : the input of the target function
- $\frac{\partial}{\partial \theta_j} J(\theta)$ : partial derivative of the target function respect to $\theta_j$
- $\alpha$ : learning rate

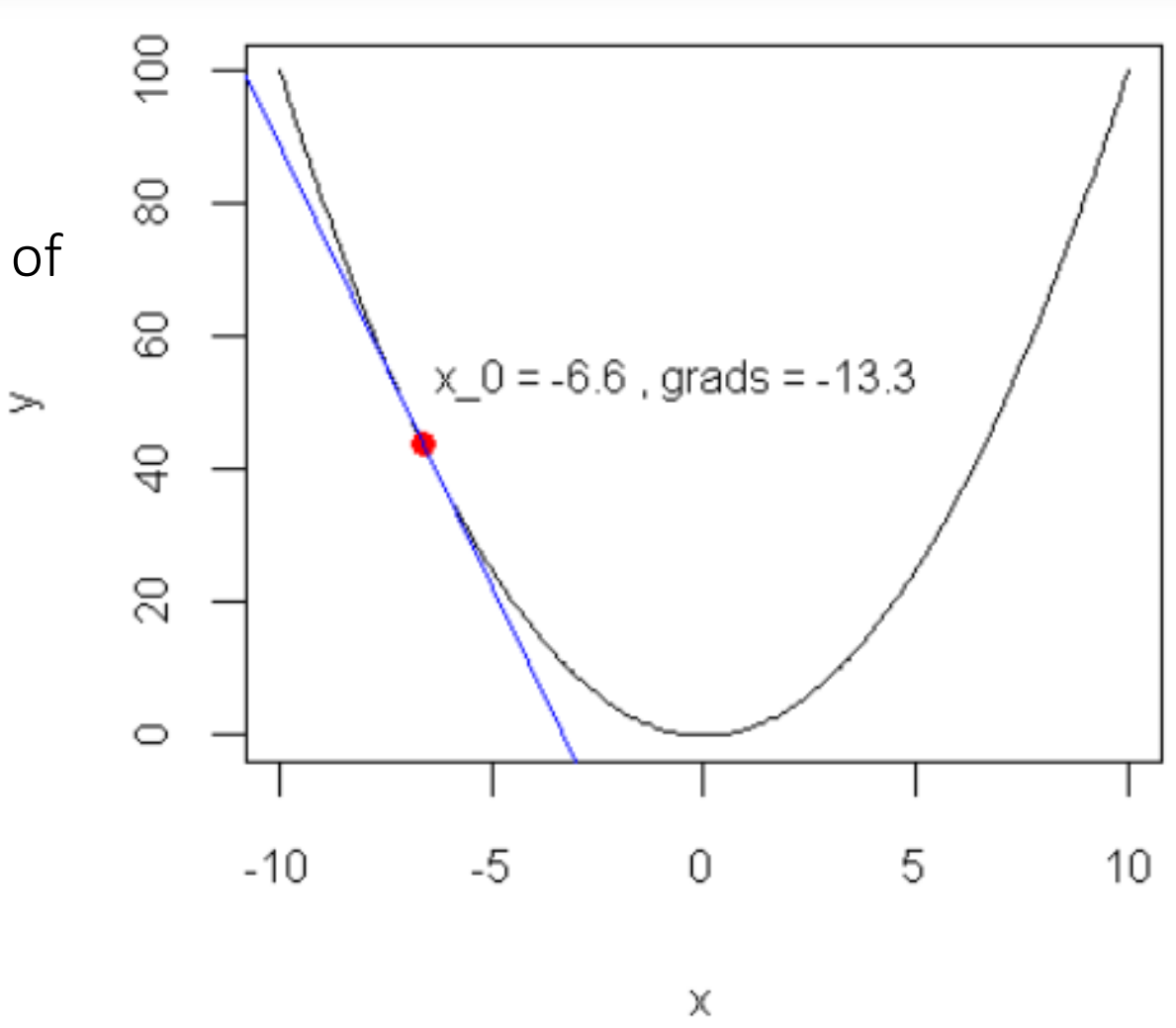
Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

**Example 3:** Given function:

$$y = f(x) = x^2$$

using Gradient Descent method to find the value of x where y = f(x) is minimum.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

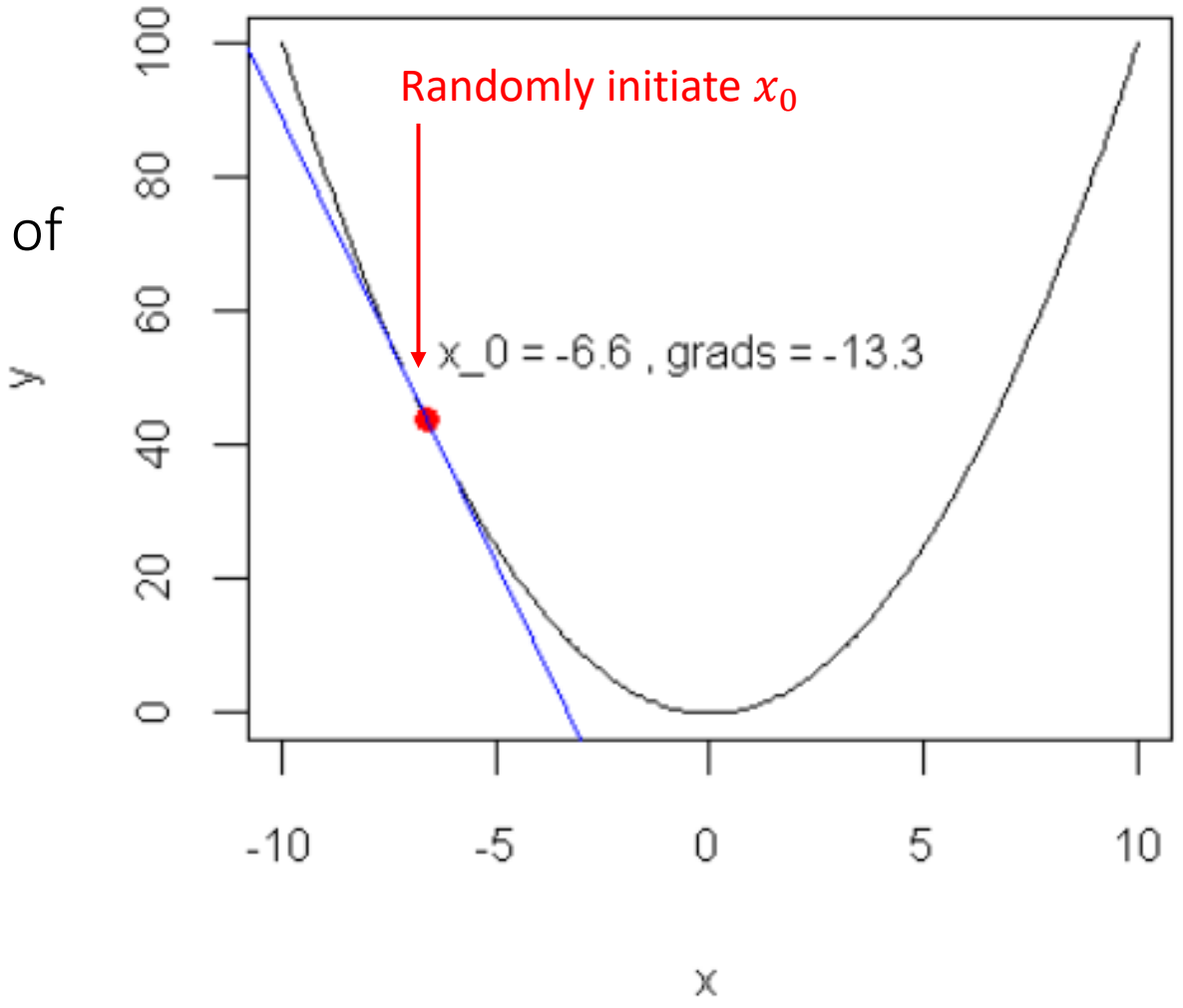
# II. Neural Networks | Calculation | Gradient descent

**Example 3:** Given function:

$$y = f(x) = x^2$$

using Gradient Descent method to find the value of x where y = f(x) is minimum.

- $J(\theta)$ : the target function to be minimized
  - $y = f(x) = x^2$
- $\theta_j$ : the input of the target function
  - x
- $\frac{\partial}{\partial \theta_j} J(\theta)$ : partial derivative of the target function respect to $\theta_j$
  - $\frac{df(x)}{dx} = 2x$




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

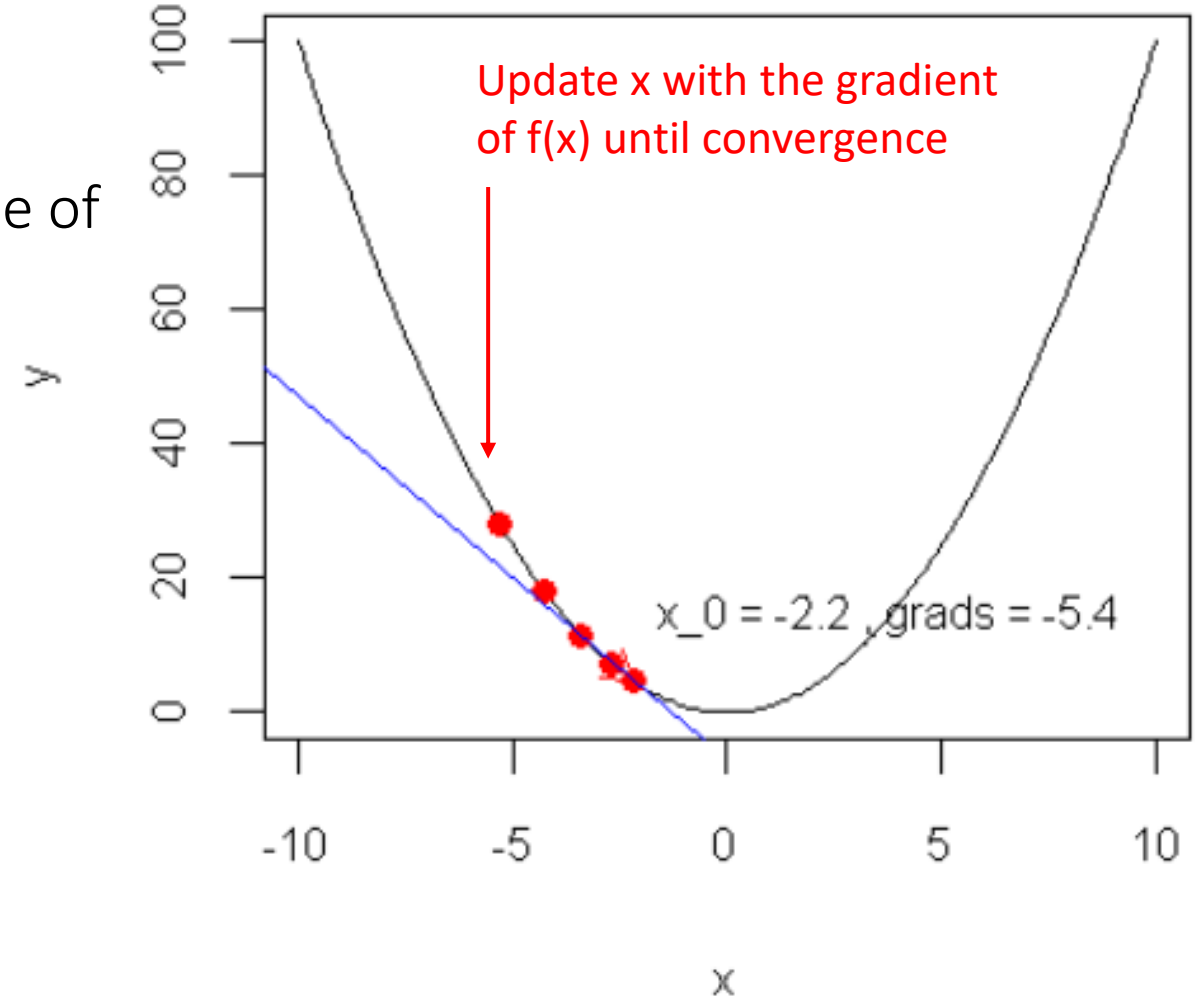
# II. Neural Networks | Calculation | Gradient descent

**Example 3:** Given function:

$$y = f(x) = x^2$$

using Gradient Descent method to find the value of x where y = f(x) is minimum.

- $J(\theta)$ : the target function to be minimized
  - $y = f(x) = x^2$
- $\theta_j$ : the input of the target function
  - x
- $\frac{\partial}{\partial \theta_j} J(\theta)$ : partial derivative of the target function respect to $\theta_j$
  - $\frac{df(x)}{dx} = 2x$
- Update until convergence:
  - $x \leftarrow x - \alpha \frac{df(x)}{dx}$



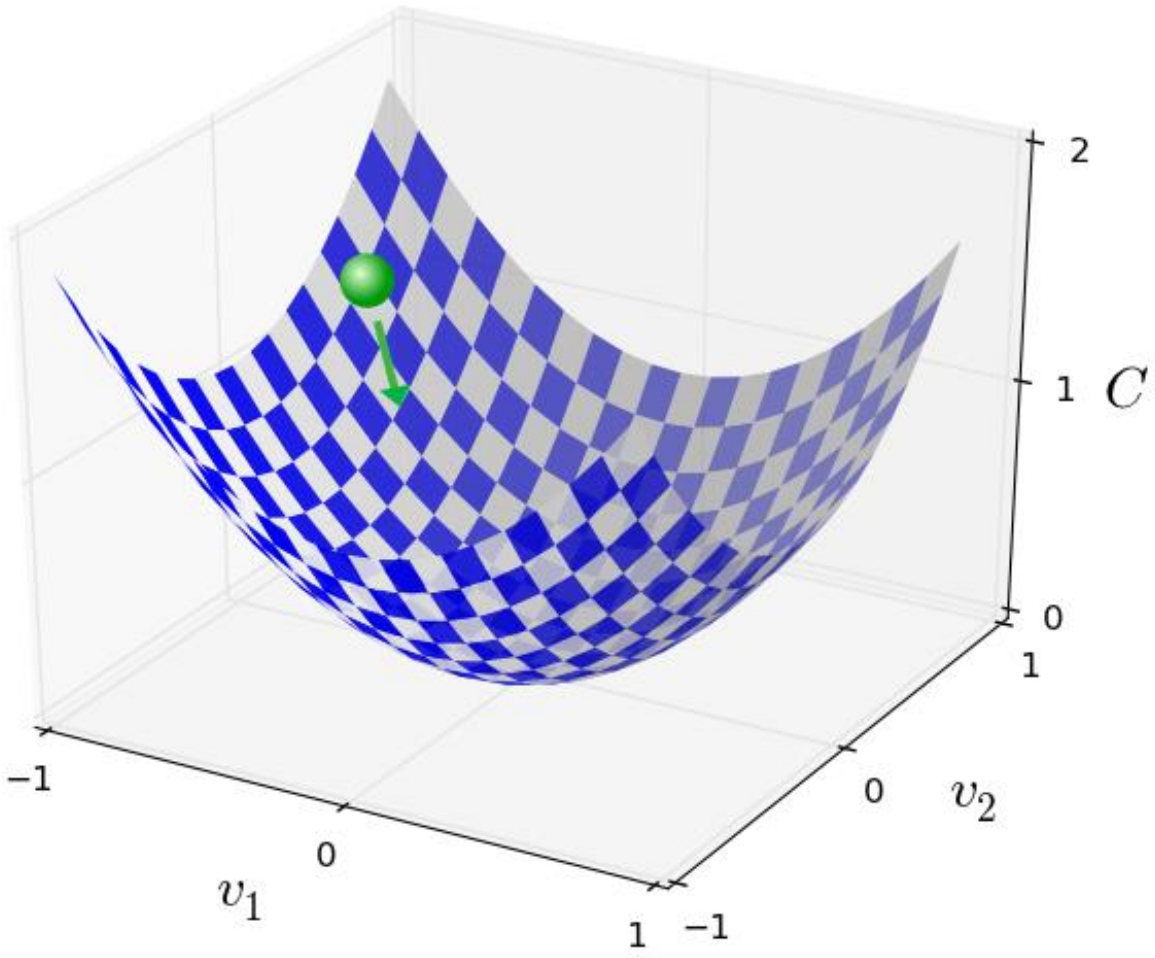

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

- Gradient descent with multiple variables.
- The cost/target function: $C(v_1, v_2)$
- Partial derivative respected to each variable.

$$v \rightarrow v' = v - \eta \nabla C$$

- Repeat until reaching to global minimum.
- Simulate the motion of the ball rolling down the valley.



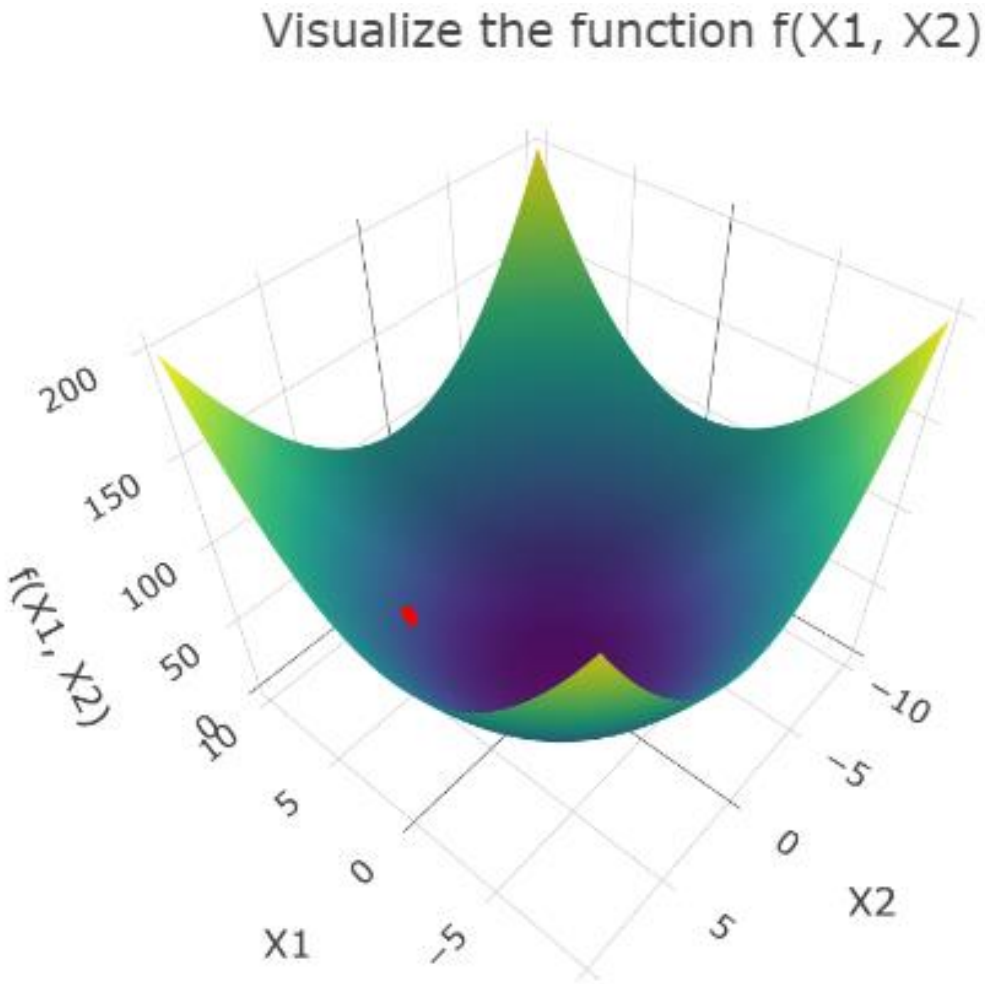

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

**Example 4:** Given function:

$$y = f(x_1, x_2) = x_1^2 + x_2^2$$

using Gradient Descent method to find the value of $x_1, x_2$ where $y = f(x_1, x_2)$ is minimum.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

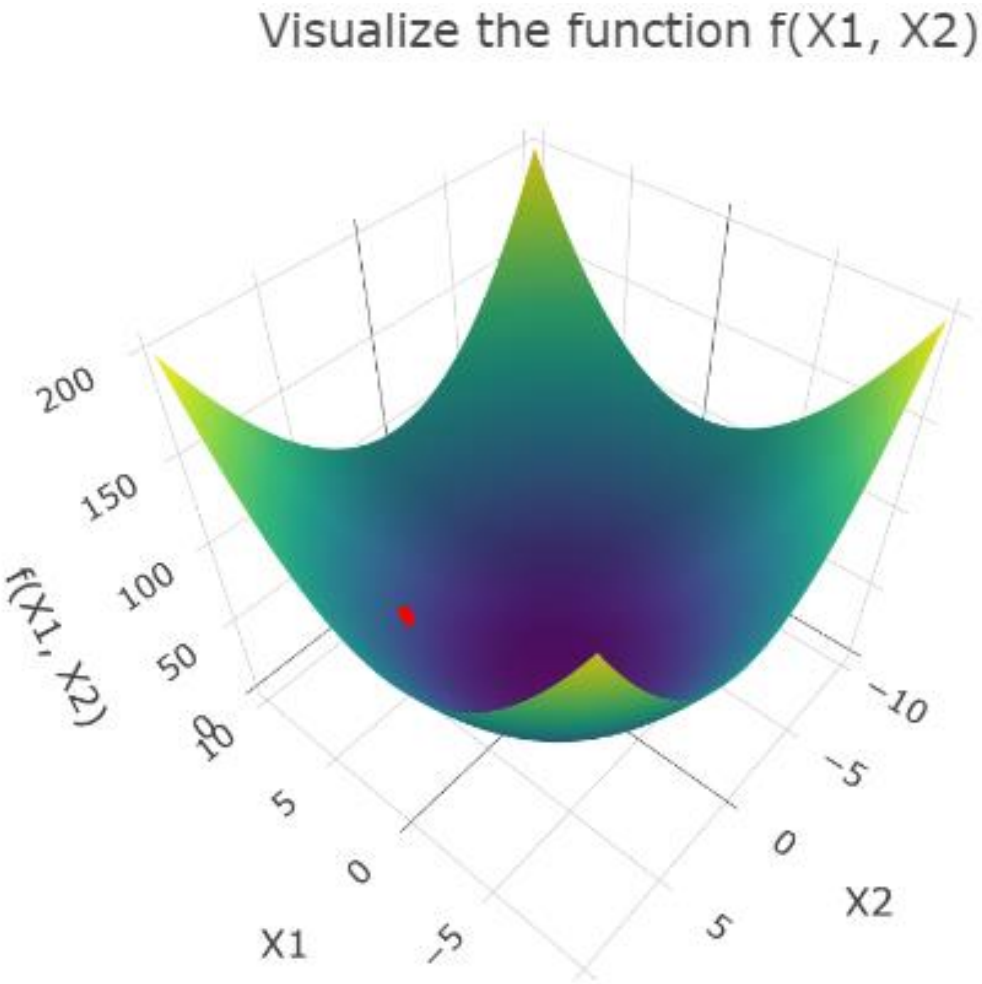
# II. Neural Networks | Calculation | Gradient descent

**Example 4:** Given function:

$$y = f(x_1, x_2) = x_1^2 + x_2^2$$

using Gradient Descent method to find the value of $x_1, x_2$ where $y = f(x_1, x_2)$ is minimum.

- $J(\theta)$ : the target function to be minimized
  - $y = f(x_1, x_2) = x_1^2 + x_2^2$
- $\theta_j$ : the input of the target function
  - $x_1$
  - $x_2$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ : partial derivative of the target function respect to $\theta_j$
  - $\frac{df(x)}{dx_1} = 2x_1$
  - $\frac{df(x)}{dx_2} = 2x_2$




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

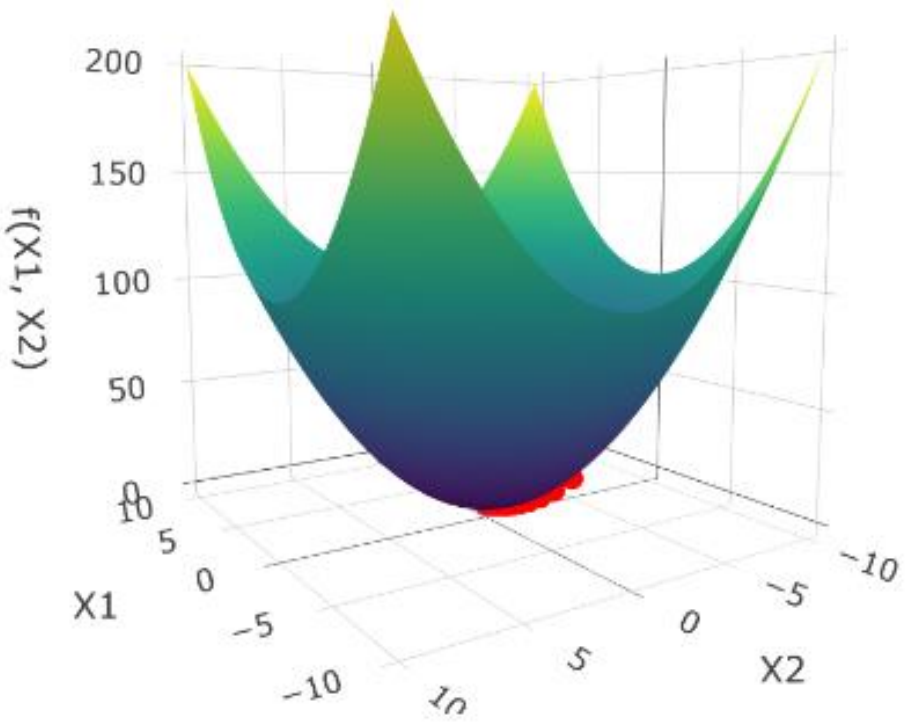
**Example 4:** Given function:

$$y = f(x_1, x_2) = x_1^2 + x_2^2$$

using Gradient Descent method to find the value of $x_1, x_2$ where $y = f(x_1, x_2)$ is minimum.

- Update until convergence:
  - $x_1 \leftarrow x_1 - \alpha \frac{df(x)}{dx_1}$
  - $x_2 \leftarrow x_2 - \alpha \frac{df(x)}{dx_2}$

Visualize the graident descent process for function f(X1, X2)




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

- Gradient Descent with Logistic Regression

- Cost/target function:
  - Log-likelihood function (Chapter 3, Section 2, slide 22):

$$\ell(\beta) = \log \mathrm{L}(\beta) = \sum_{i=1}^{N} \{y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta))\}$$

  - Log-likelihood function (new annotation):

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[ -y^{(i)} \, log \left( h_\theta \left( x^{(i)} \right) \right) - \left( 1 - y^{(i)} \right) log \left( 1 - h_\theta \left( x^{(i)} \right) \right) \right]$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

- Gradient Descent with Logistic Regression

- Cost/target function (the log-likelihood):

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\, log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right) log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]$$

- Partial derivatives respected to $\theta_j$:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

- Update $\theta_j$, repeat until convergence:

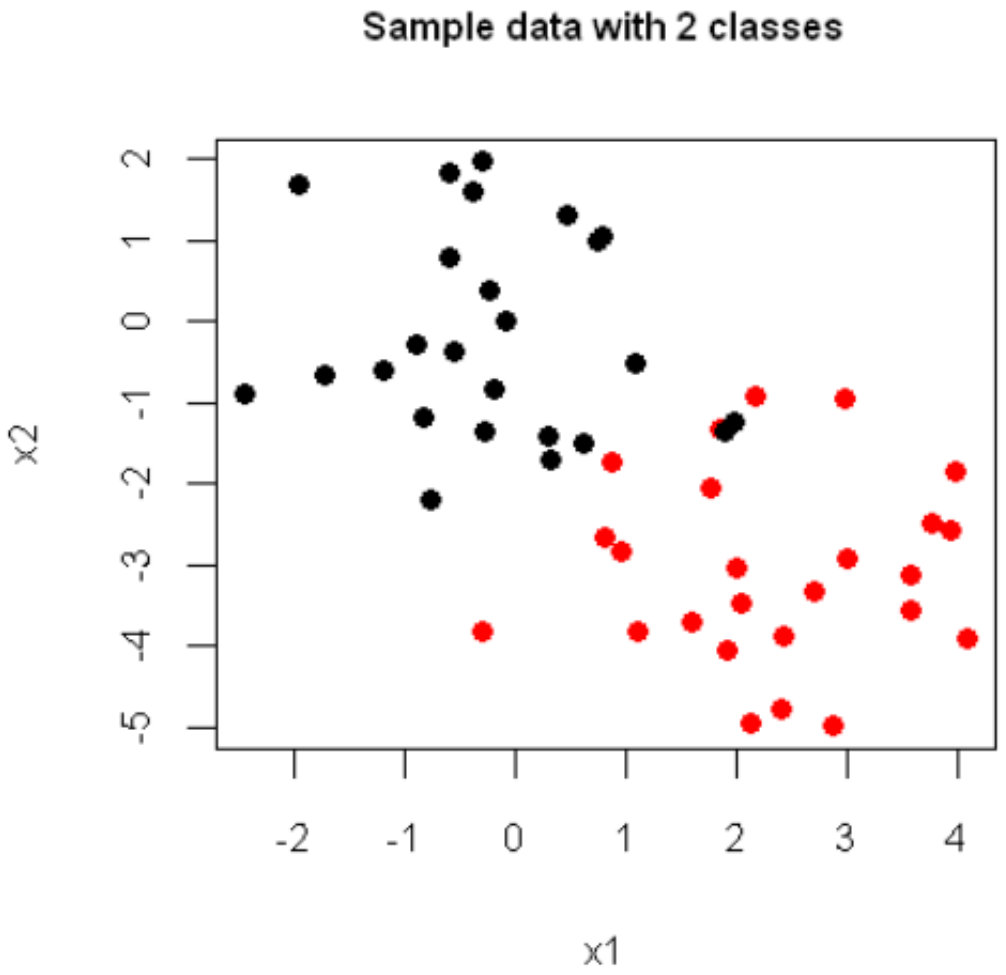
$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

**Example 5:** Using Gradient Descent to estimate the intercept and coefficient of logistic regression model.

| X1 | X2 | Y |
|---|---|---|
| 1.1030855 | -3.8382871 | 1 |
| 2.1848492 | -0.9336986 | 1 |
| 3.5878453 | -3.5622471 | 1 |
| 0.8696243 | -1.7242845 | 1 |
| 1.9197482 | -4.0475726 | 1 |
| 2.1324203 | -4.9658782 | 1 |




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

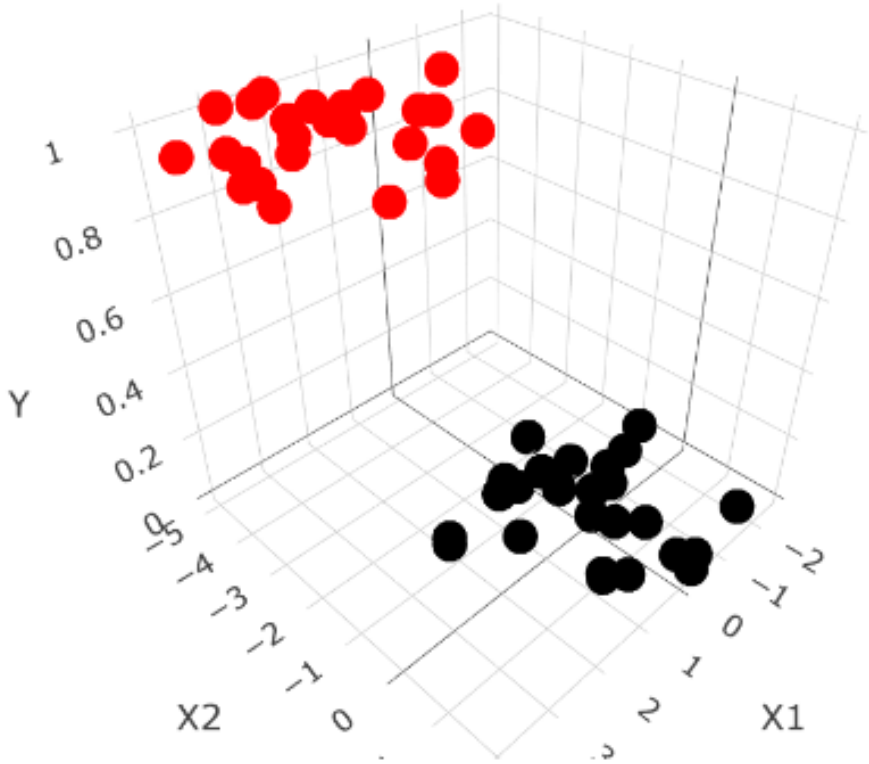
**Example 5:** Using Gradient Descent to estimate the intercept and coefficient of logistic regression model.

Results from a standard R package:

```
Call:  glm(formula = y ~ x, family = "binomial")

Coefficients:
(Intercept)           x1           x2
    -10.631        2.868       -4.460

Degrees of Freedom: 49 Total (i.e. Null);  47 Residual
Null Deviance:      69.31
Residual Deviance: 9.924        AIC: 15.92
```

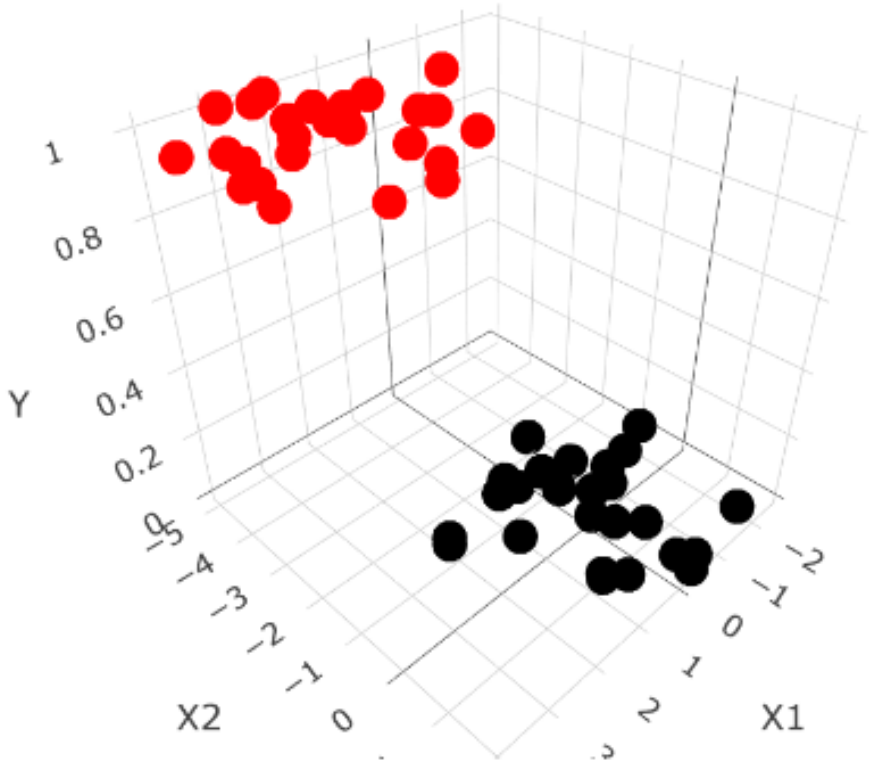



Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

**Example 5:** Using Gradient Descent to estimate the intercept and coefficient of logistic regression model.

- $J(\theta)$ : the target function to be minimized
  - ?
- $\theta_j$ : the input of the target function
  - ?
- $\frac{\partial}{\partial \theta_j} J(\theta)$ : partial derivative of the target function respect to $\theta_j$
  - ?
- Update until convergence:
  - ?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

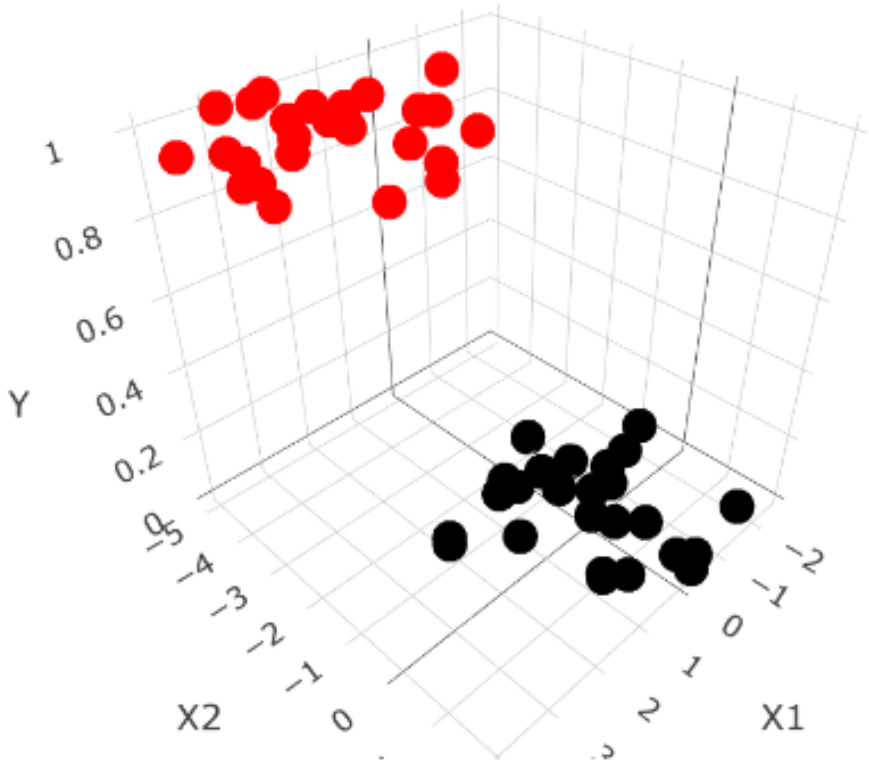
**Example 5:** Using Gradient Descent to estimate the intercept and coefficient of logistic regression model.

- $J(\theta)$ : the target function to be minimized
  - Log-Likelihood

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\, log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right) log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]$$

- $\theta_j$ : the input of the target function
  - $\beta_j$
- $\frac{\partial}{\partial\theta_j}J(\theta)$ : partial derivative of the target function respect to $\theta_j$
  - Partial derivative per $\beta_j$

$$\frac{\partial J(\theta)}{\partial\theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$



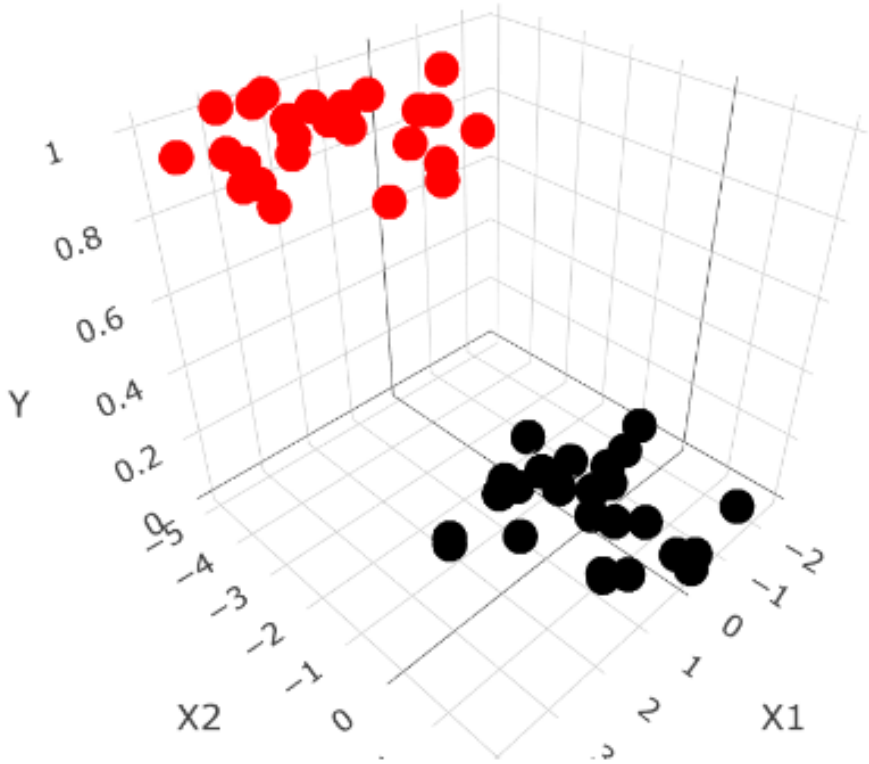

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

**Example 5:** Using Gradient Descent to estimate the intercept and coefficient of logistic regression model.

- Update until convergence:
  - Update $\beta_j$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

**Example 5:** Using Gradient Descent to estimate the intercept and coefficient of logistic regression model.

Results from manual gradient descent:

- Number of iterations until convergence: 5000
- Logistic Regression betas:

```
[1] "Intercept: -10.631"
[1] "Coefficient: 2.868, -4.46"
[1] "Latest cost: 0.09924"
```


Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

Input of target function
e.g. beta, theta

The gradient of the target function (partial derivative, sloop)

Repeat until convergence {

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

The learning rate

The target function to be minimized

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Gradient descent

- Gradient descent for LR model:

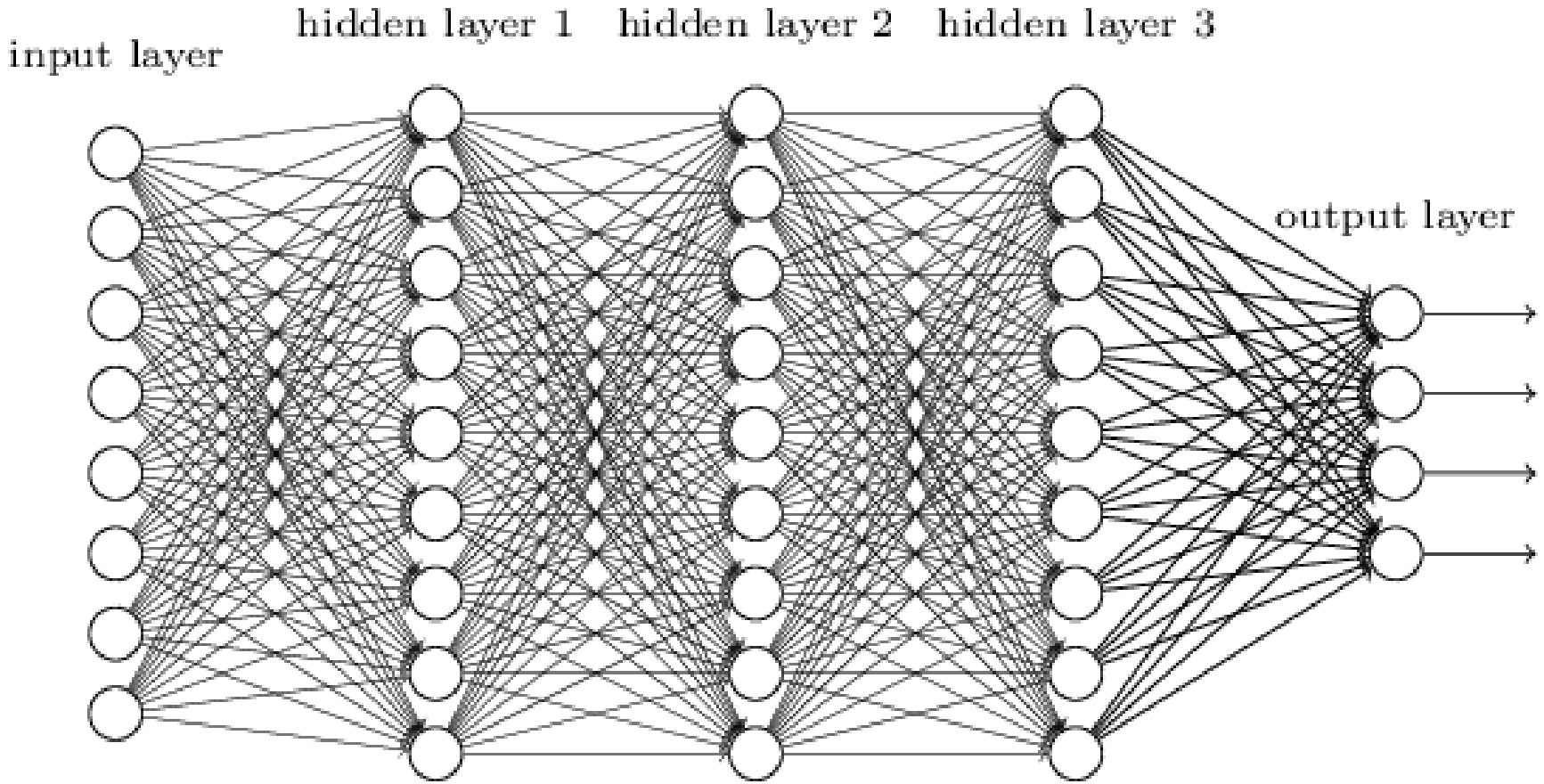
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left( h_\theta \left( x^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

- Gradient descent for NN model ?

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

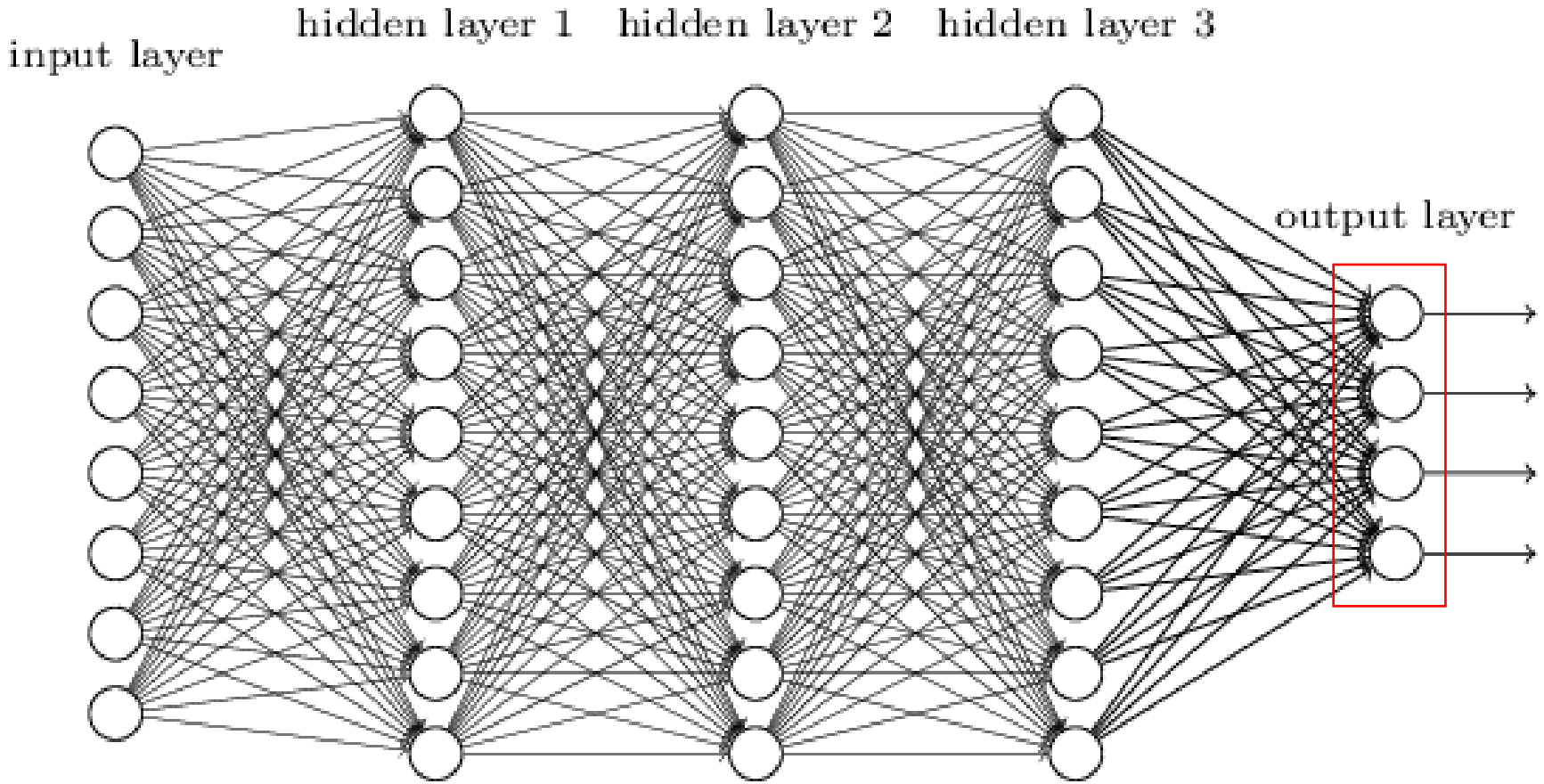
- **Problem:** How to calculate the gradient of the weights $w_j$ and bias $b_j$ in every layer of NN?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

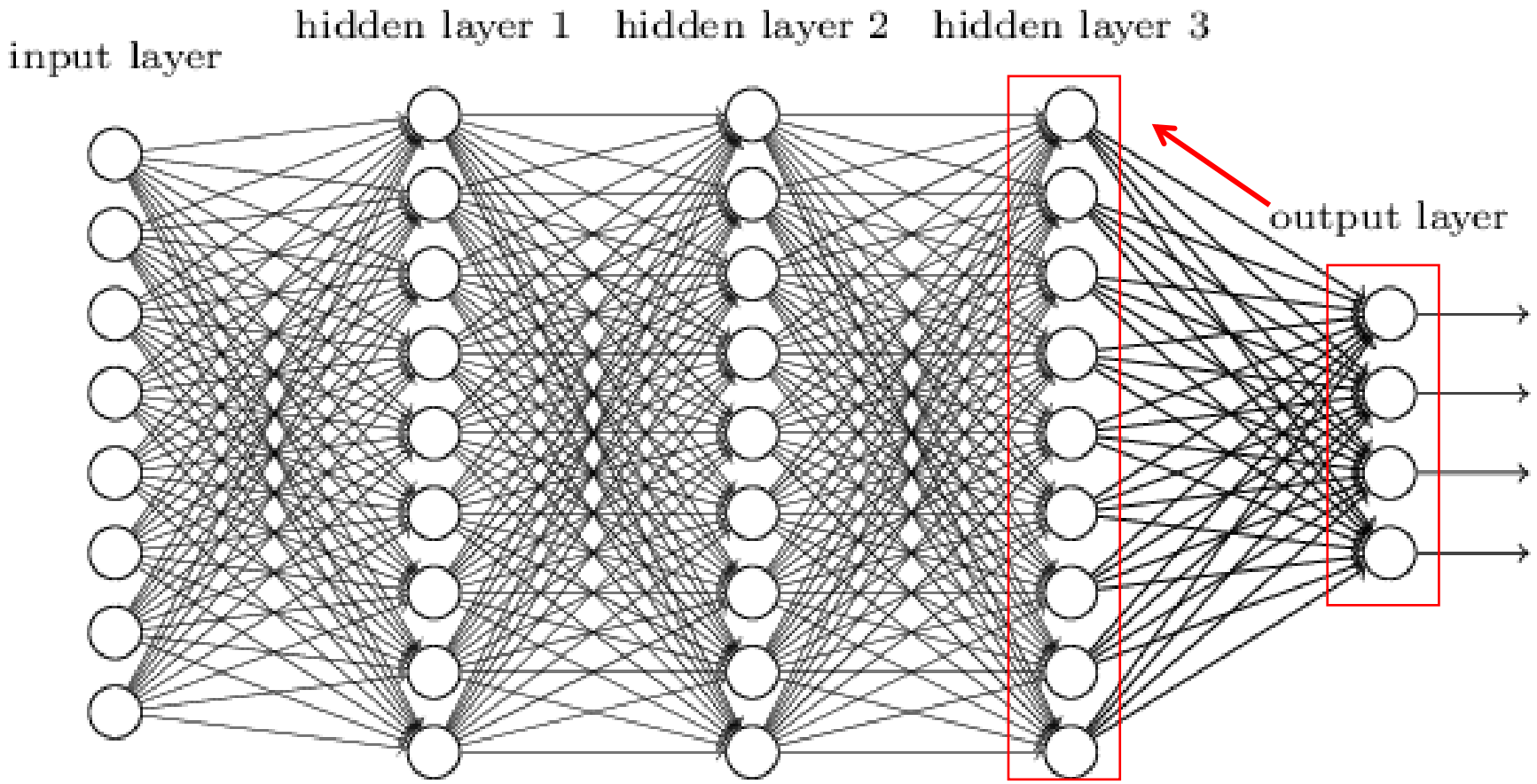
- **Problem:** How to calculate the gradient of the weights $w_j$ and bias $b_j$ in every layer of NN?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

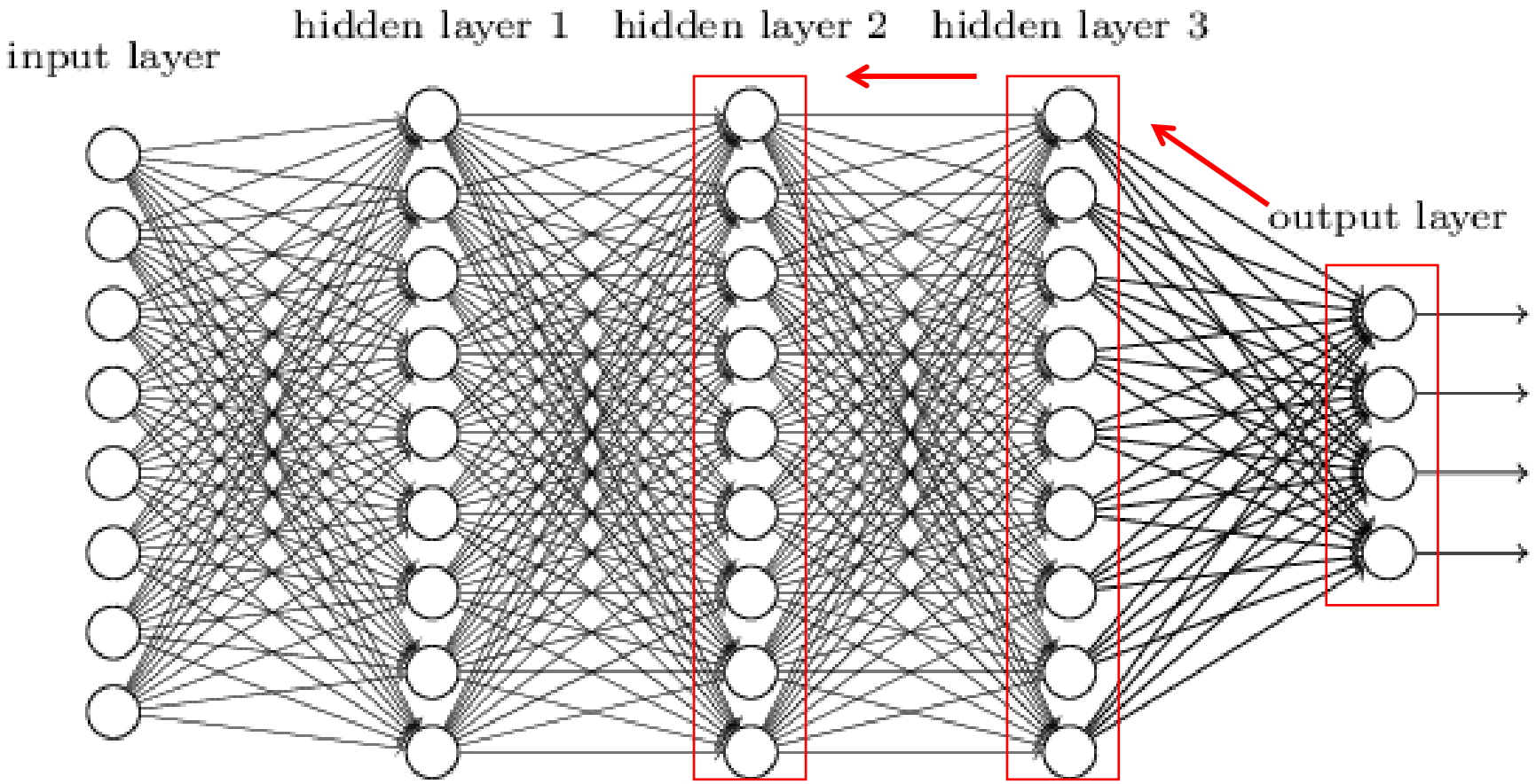
- **Problem:** How to calculate the gradient of the weights $w_j$ and bias $b_j$ in every layer of NN?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

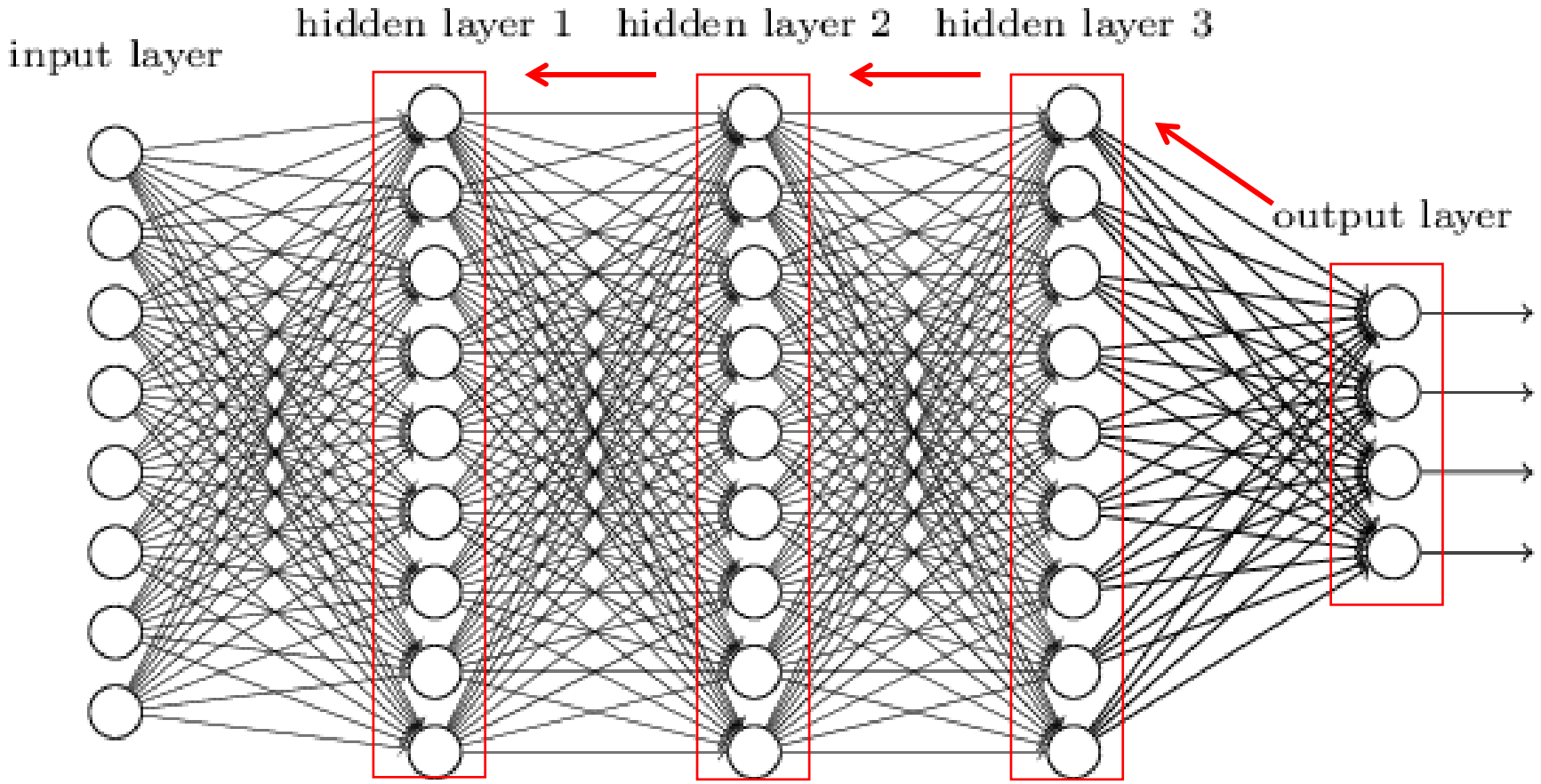
- **Problem:** How to calculate the gradient of the weights $w_j$ and bias $b_j$ in every layer of NN?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

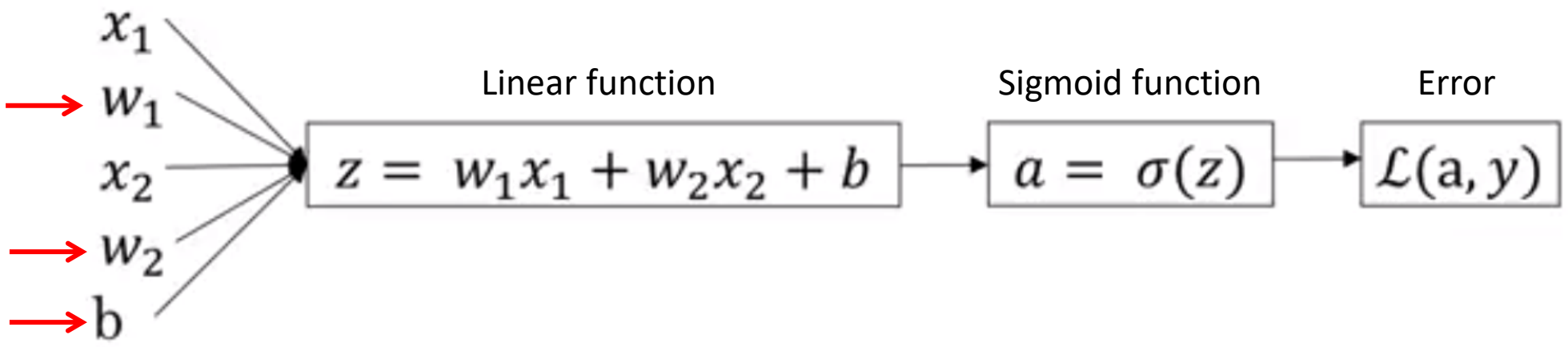
- **Problem:** How to calculate the gradient of the weights $w_j$ and bias $b_j$ in every layer of NN?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- Represent Logistic Regression with computation graph (chain rule):



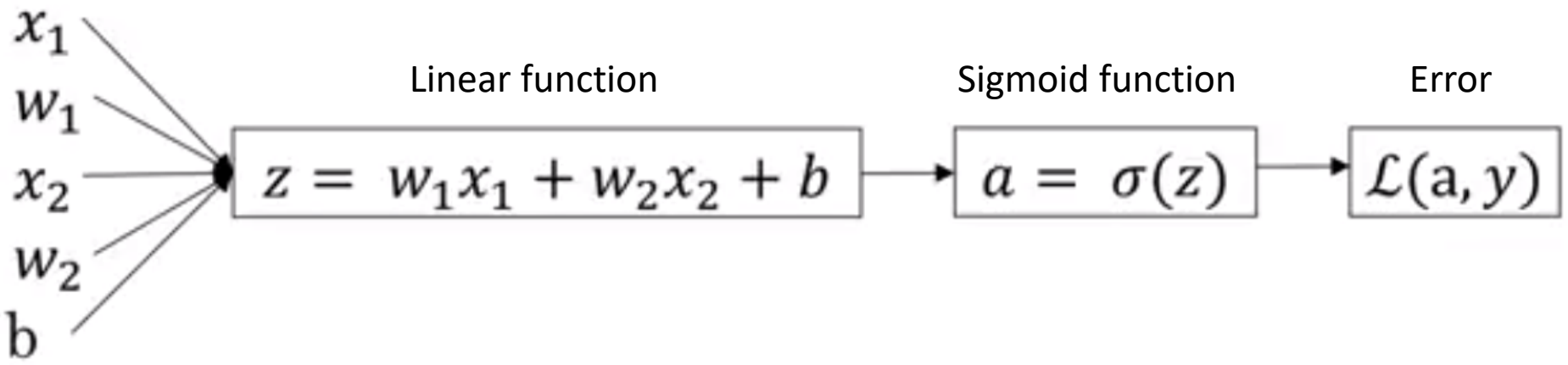

In which:

- $w_1, w_2$ : coefficient of LR model (beta)
- $b$ : intercept of the LR model (beta0 or bias)
- $\sigma(z)$ : sigmoid function of LR model
- $L(a, y)$ : lost function of LR model, sum of all lost functions is the target cost function J

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

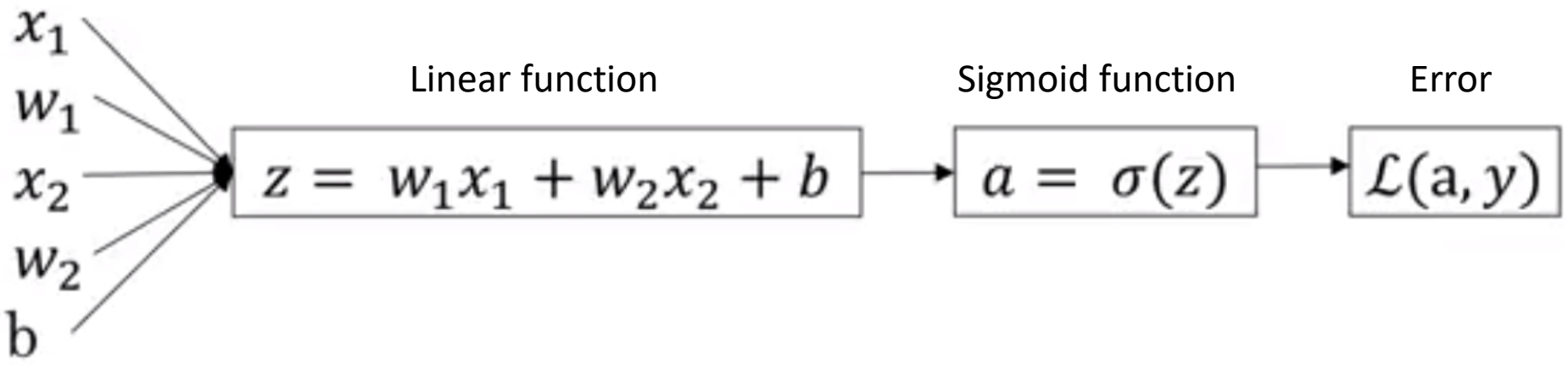
- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?

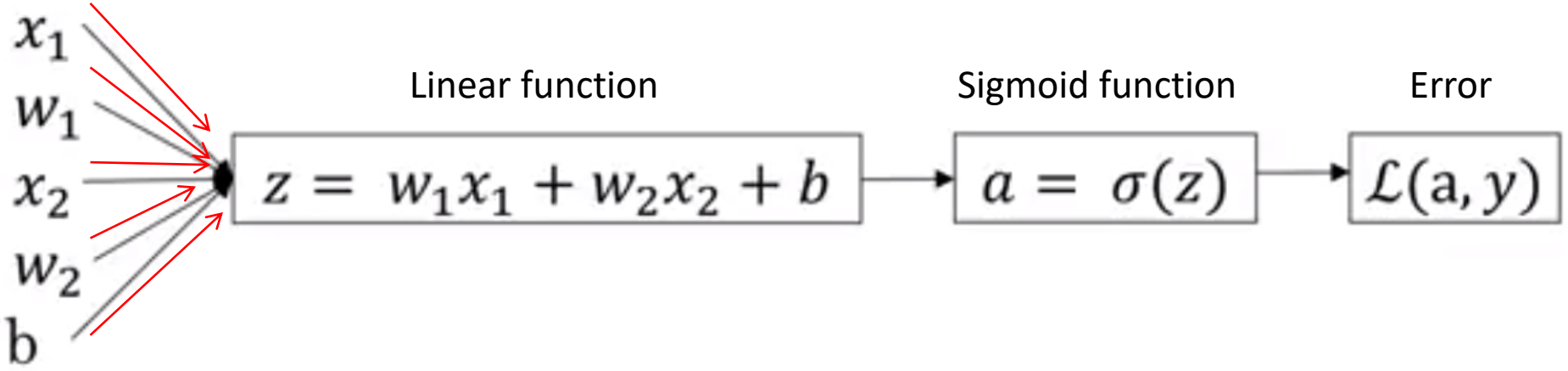



- Step 1: Forward propagation

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?

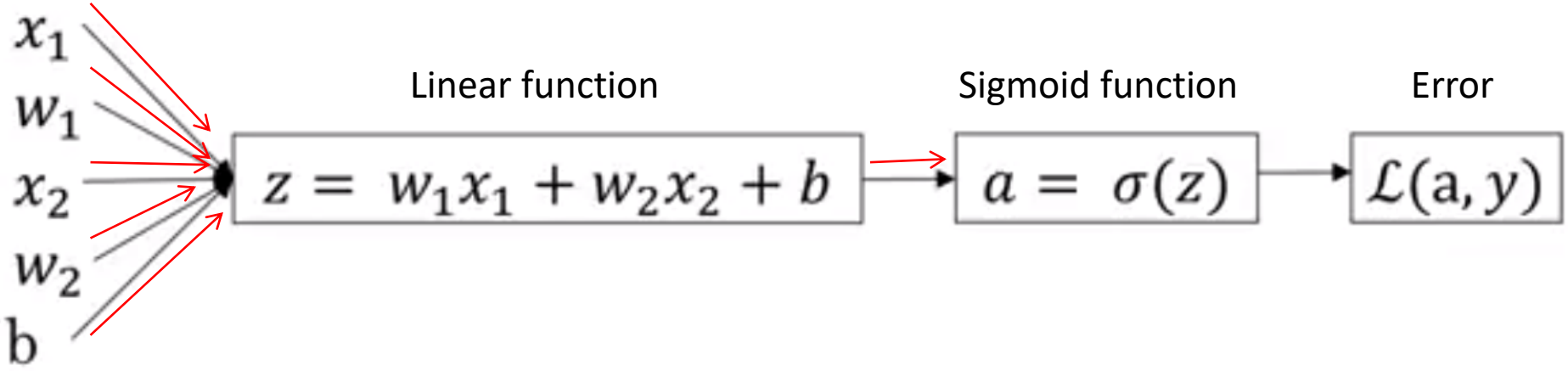



- Step 1: Forward propagation

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?

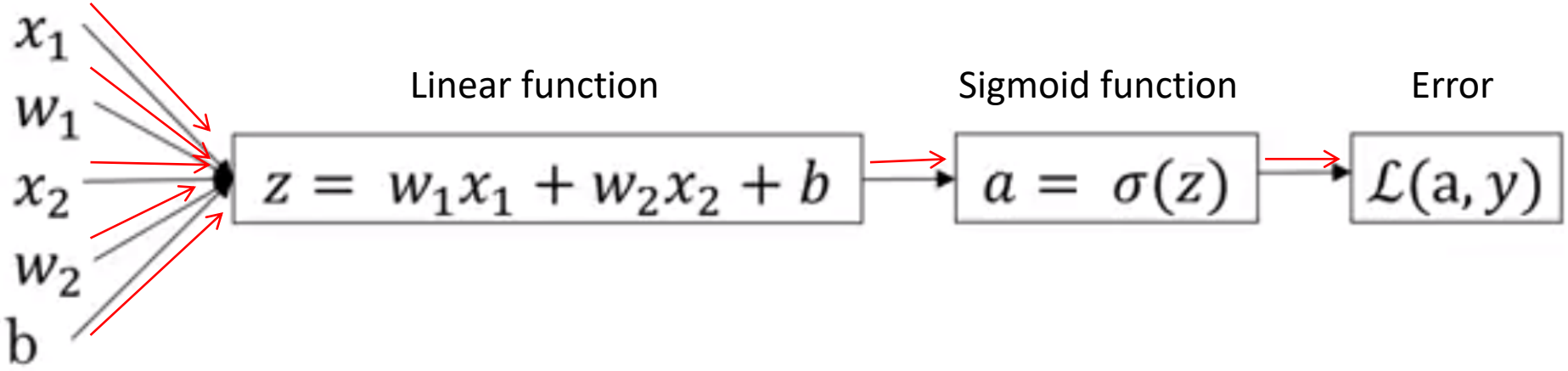



- Step 1: Forward propagation

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?

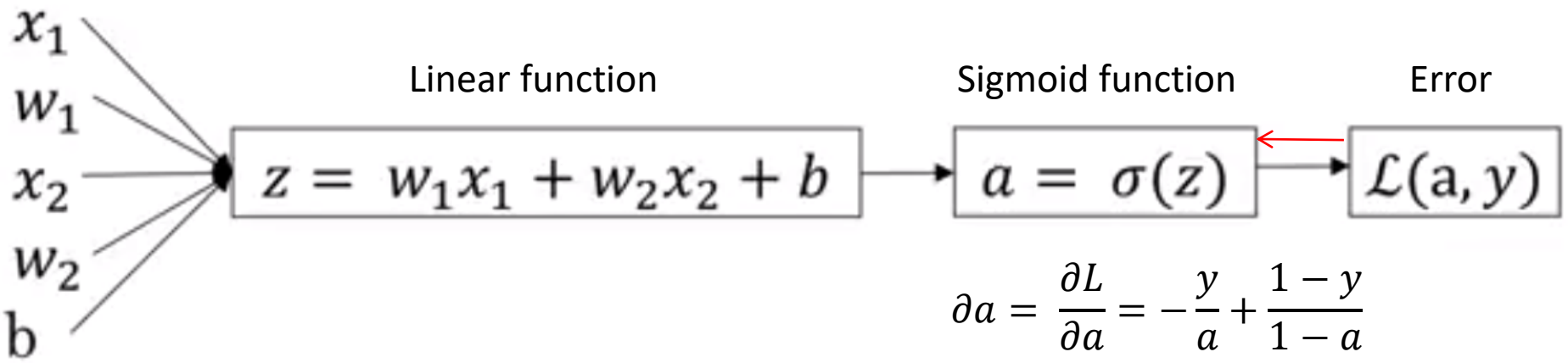



- Step 1: Forward propagation

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?

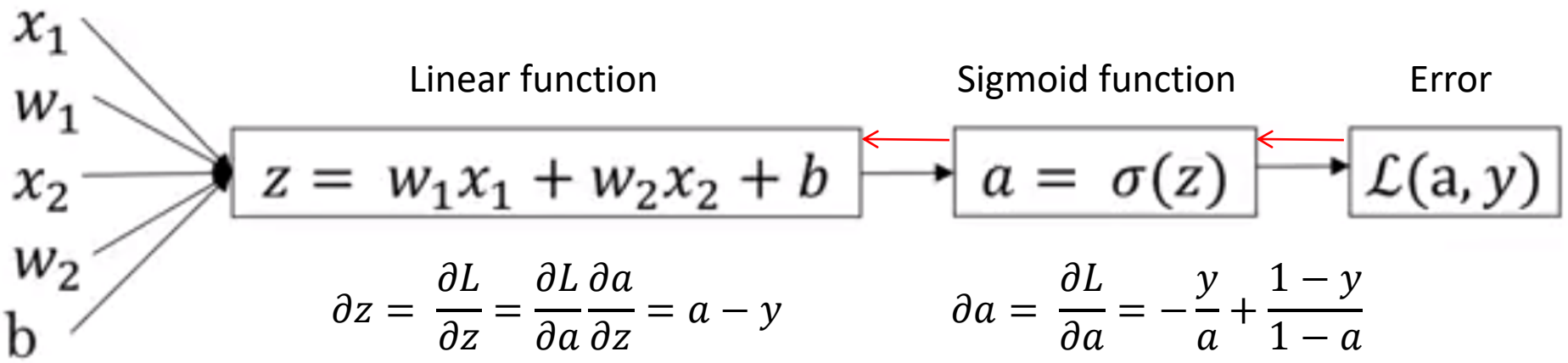



- Step 2: Backward propagation

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?



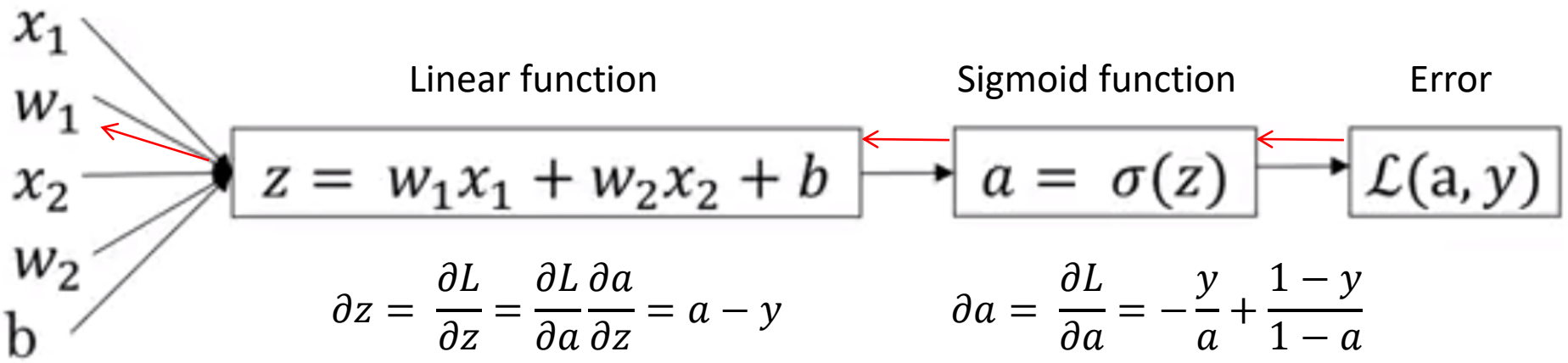

$$\partial z = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z} = a - y \qquad \partial a = \frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$$

- Step 2: Backward propagation

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?




$$\partial z = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z} = a - y \qquad \partial a = \frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$$

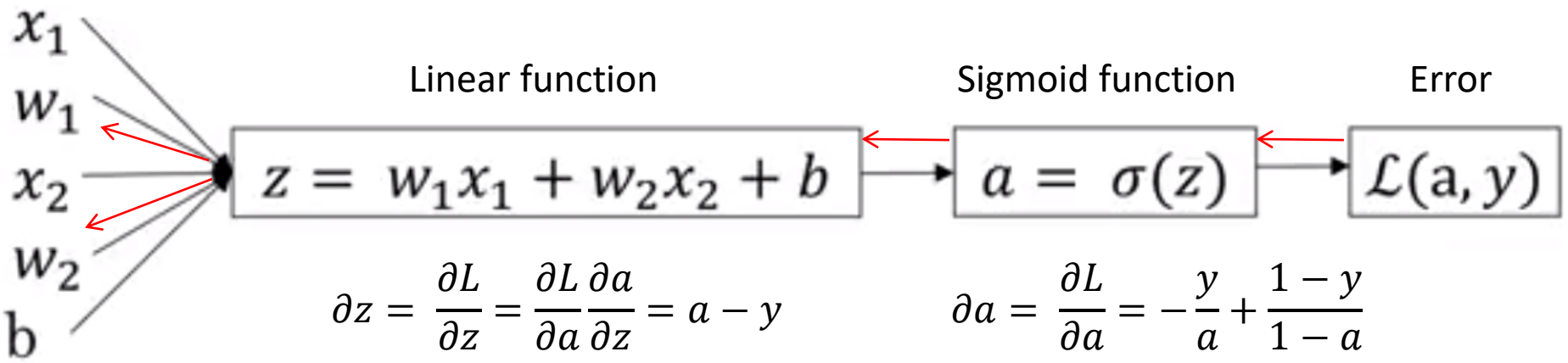
- Step 2: Backward propagation

$$\partial w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_1} = x_1 . \partial z$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?




$$\partial z = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z} = a - y \qquad \partial a = \frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$$

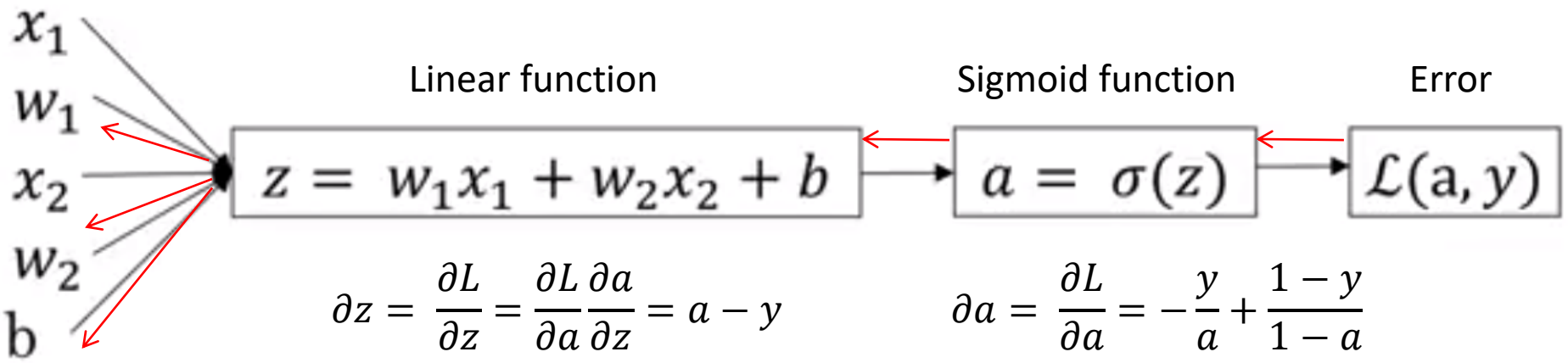
- Step 2: Backward propagation

$$\partial w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_1} = x_1.\partial z \qquad \partial w_2 = \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_2} = x_2.\partial z$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?




$$\partial z = \frac{\partial L}{\partial z} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z} = a - y \qquad \partial a = \frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$$

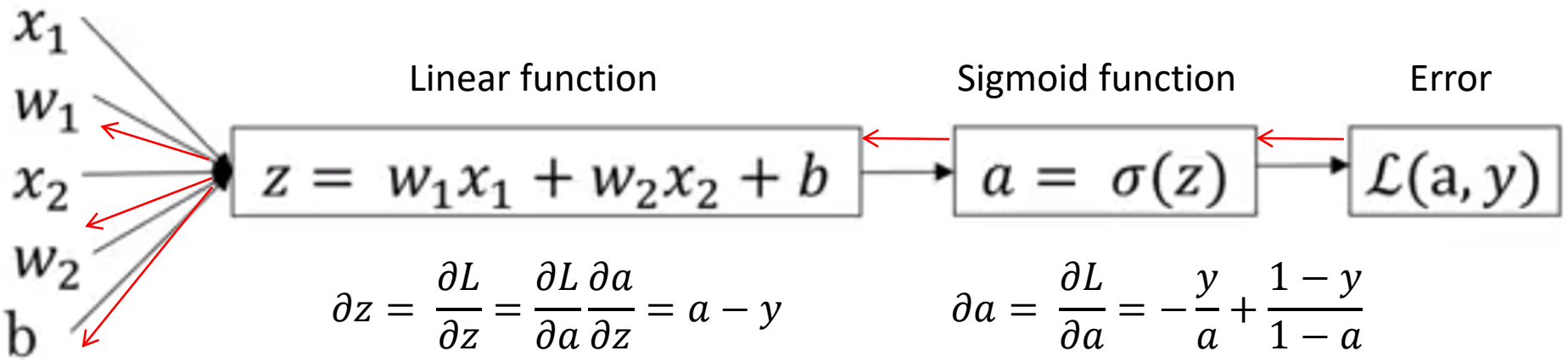
- Step 2: Backward propagation

$$\partial w_1 = \frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_1} = x_1.\partial z \qquad \partial w_2 = \frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_2} = x_2.\partial z \qquad \partial b = \frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial b} = \partial z$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

- **Question:** In order to user gradient descent to optimize the lost function $L(a, y)$, how can we calculate the partial derivative respected to $w_1, w_2, b$ using this computation graph?

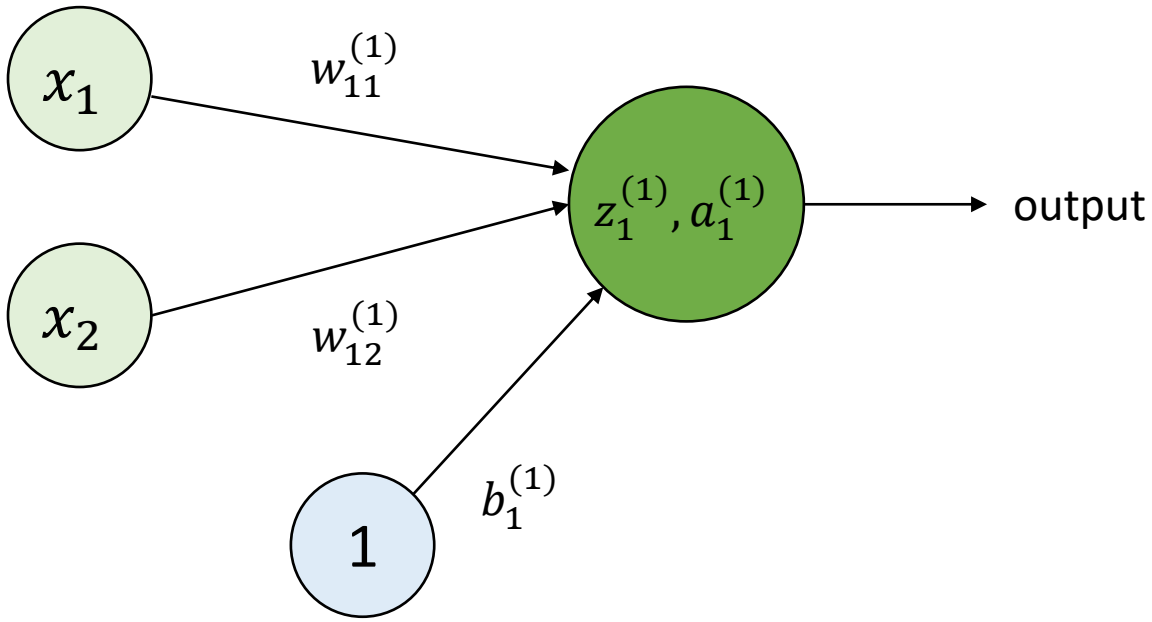



- Step 3: Repeat Forward propagation and Backward propagation until convergence

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

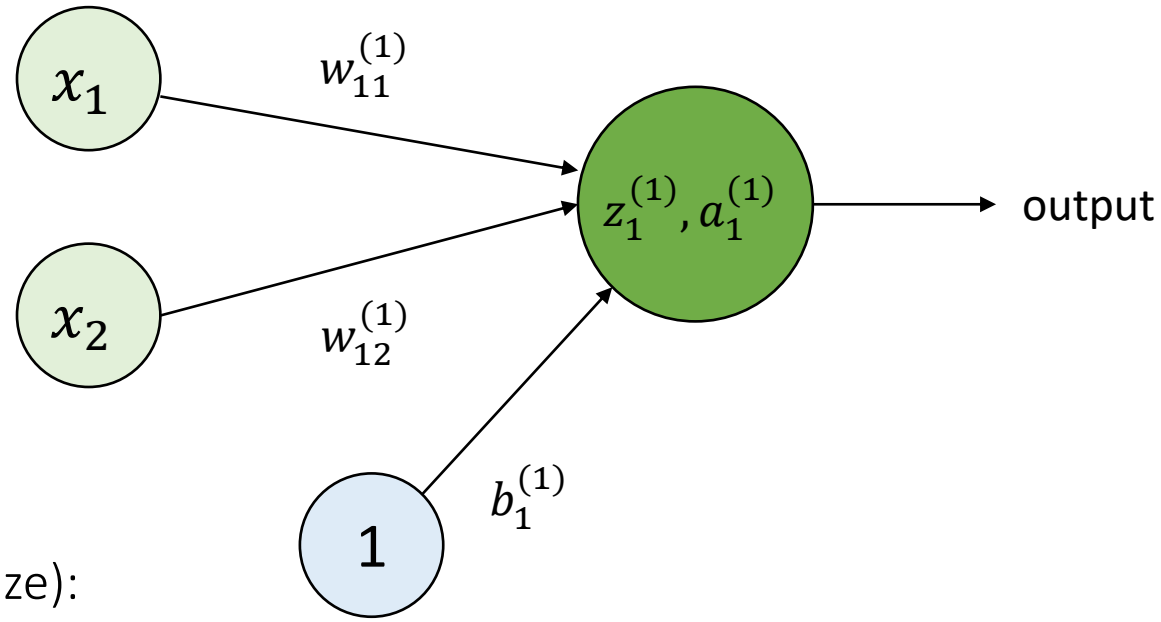



In which:

- $x_1, x_2$ : input neurons of 2 variables $x_1, x_2$
- $w_{11}^{(1)}, w_{12}^{(1)}$ : weights of the output neuron in layer 1
- $b_1^{(1)}$ : bias of the output neuron in layer 1
- $z_1^{(1)} = w_{11}^{(1)} * x_1 + w_{12}^{(1)} * x_2 + b_1^{(1)}$ : value of the neuron in layer 1
- $a_1^{(1)} = \sigma\left(z_1^{(1)}\right)$ : activate value of the neuron in layer 1, using the sigmoid activation function $\sigma$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

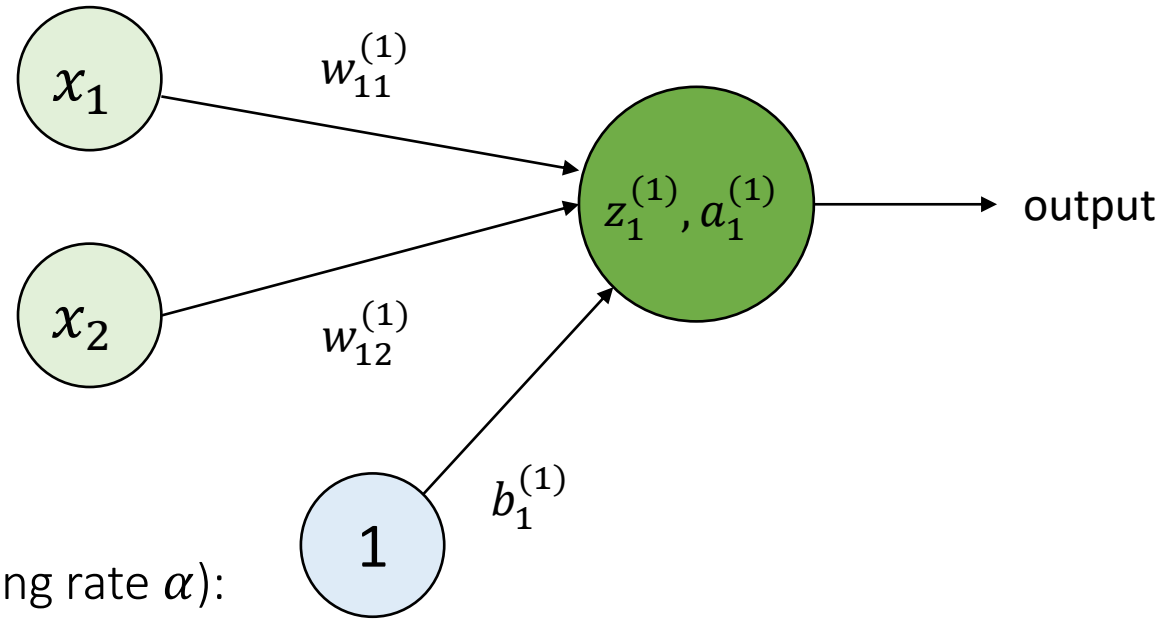



Define the loss, cost function (target function to minimize):

- $L(a, y) = \frac{1}{2} * (a_1^{(1)} - y)^2$
- $C = \sum_{j=1}^{N} L(a_j, y_j)$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

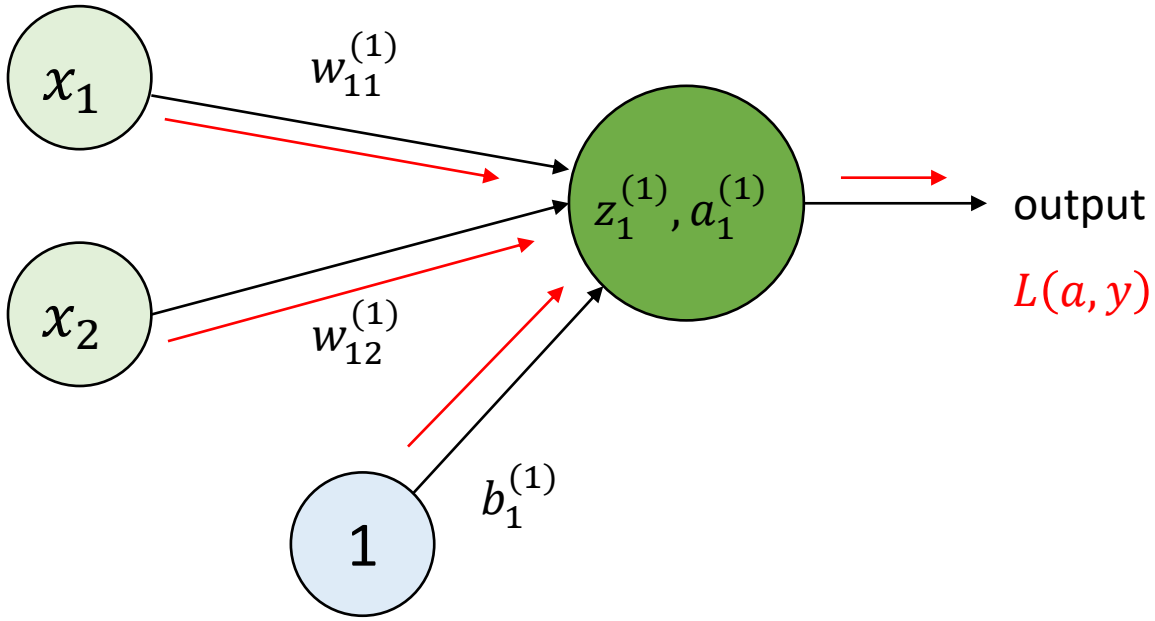



According to Gradient descent, for a single input (learning rate $\alpha$):

- $w_{11}^{(1)} := w_{11}^{(1)} - \alpha * \frac{\partial L}{\partial w_{11}^{(1)}}$
- $w_{12}^{(1)} := w_{12}^{(1)} - \alpha * \frac{\partial L}{\partial w_{12}^{(1)}}$
- $b_1^{(1)} := b_1^{(1)} - \alpha * \frac{\partial L}{\partial b_1^{(1)}}$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

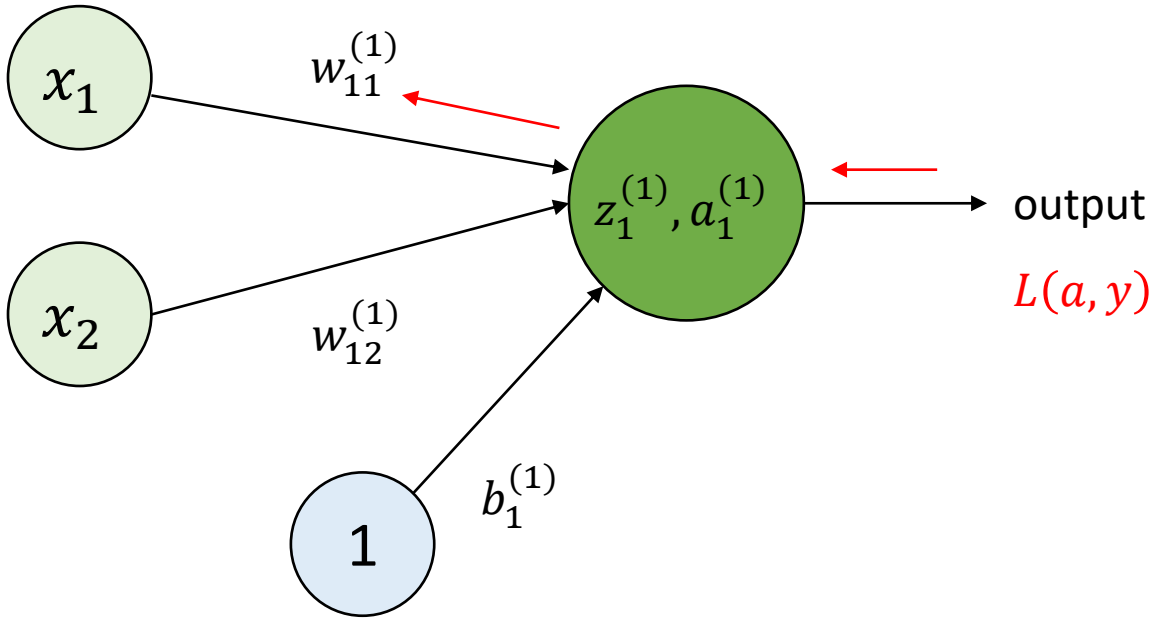



Step 1: Forward propagation

- Randomly initiate $w_{11}^{(1)}$, $w_{12}^{(1)}$, $b_1^{(1)}$
- Calculate $z_1^{(1)}, a_1^{(1)}$
- Make prediction, calculate the loss $L(a, y) = \frac{1}{2} * (a - y)^2$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

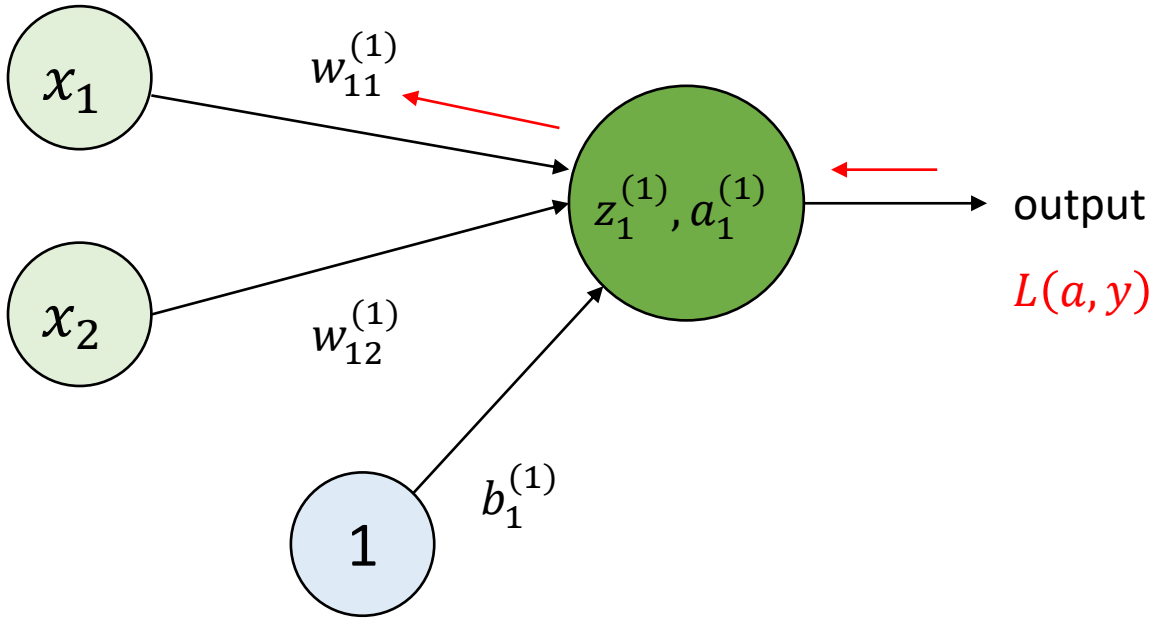



Step 2: Backward propagation

- Using gradient descent to update $w_{11}^{(1)}$
- $w_{11}^{(1)} := w_{11}^{(1)} - \alpha * \frac{\partial L}{\partial w_{11}^{(1)}}$
- According to the chain rule: $\frac{\partial L}{\partial w_{11}^{(1)}} = \frac{\partial L}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} * \frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}}$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

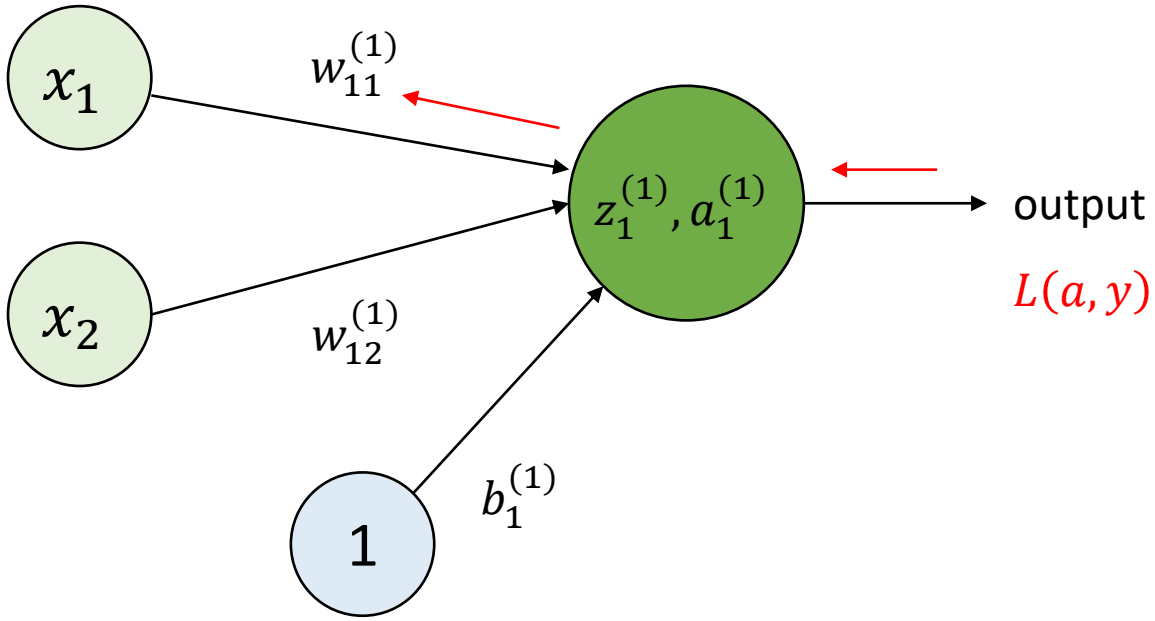



Step 2: Backward propagation

- $\frac{\partial L}{\partial a_1^{(1)}} = a_1^{(1)} - \mathrm{y}$
- $\frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} = a_1^{(1)} * (1 - a_1^{(1)})$
- $\frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}} = x_1$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

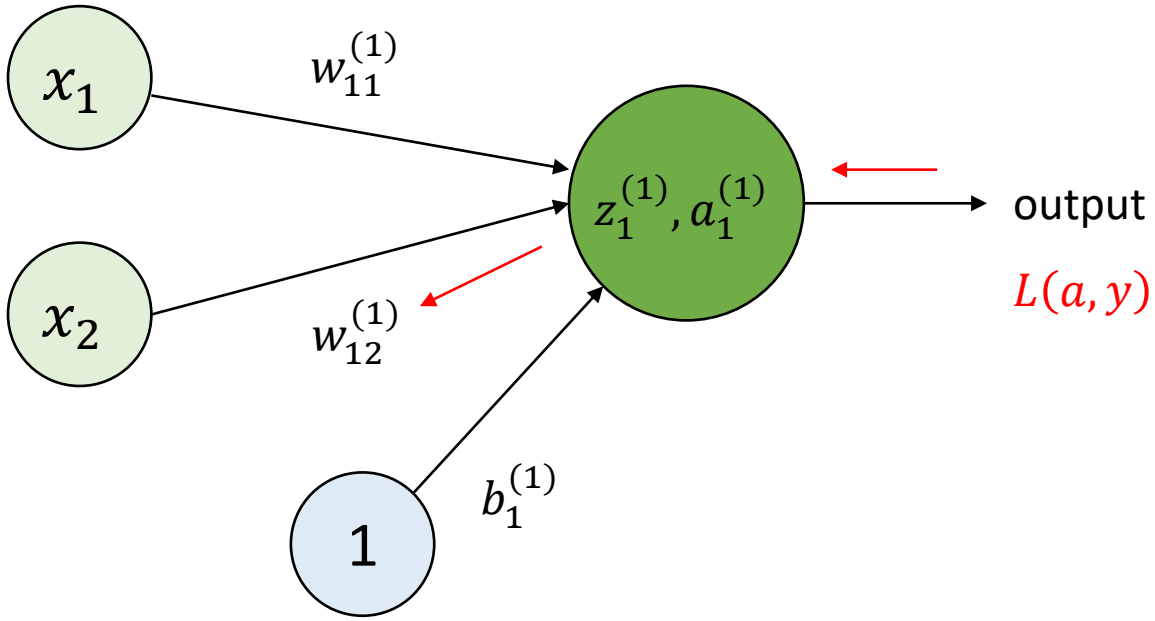



Step 2: Backward propagation

- Using gradient descent to update $w_{11}^{(1)}$
- $w_{11}^{(1)} := w_{11}^{(1)} - \alpha * \frac{\partial L}{\partial w_{11}^{(1)}} = w_{11}^{(1)} - \alpha * \left(a_1^{(1)} - \mathrm{y}\right) * a_1^{(1)} * \left(1 - a_1^{(1)}\right) * x_1$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

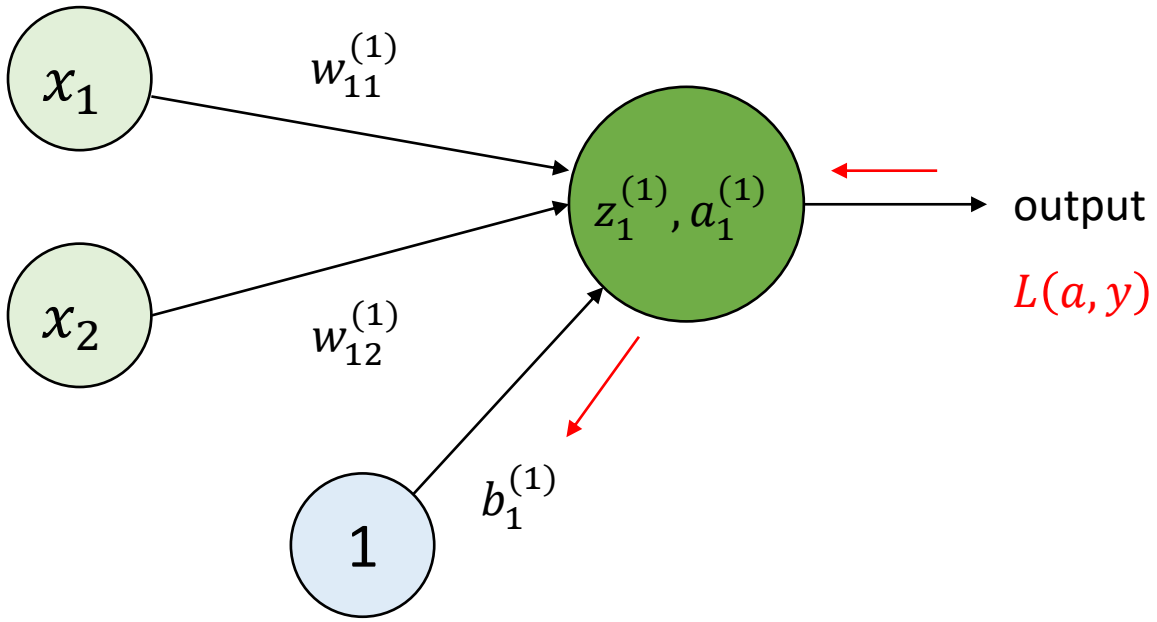



Step 2: Backward propagation

- Using gradient descent to update $w_{12}^{(1)}$
- $w_{12}^{(1)} := w_{12}^{(1)} - \alpha * \frac{\partial L}{\partial w_{12}^{(1)}} = w_{12}^{(1)} - \alpha * \left(a_1^{(1)} - \mathrm{y}\right) * a_1^{(1)} * \left(1 - a_1^{(1)}\right) * x_2$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

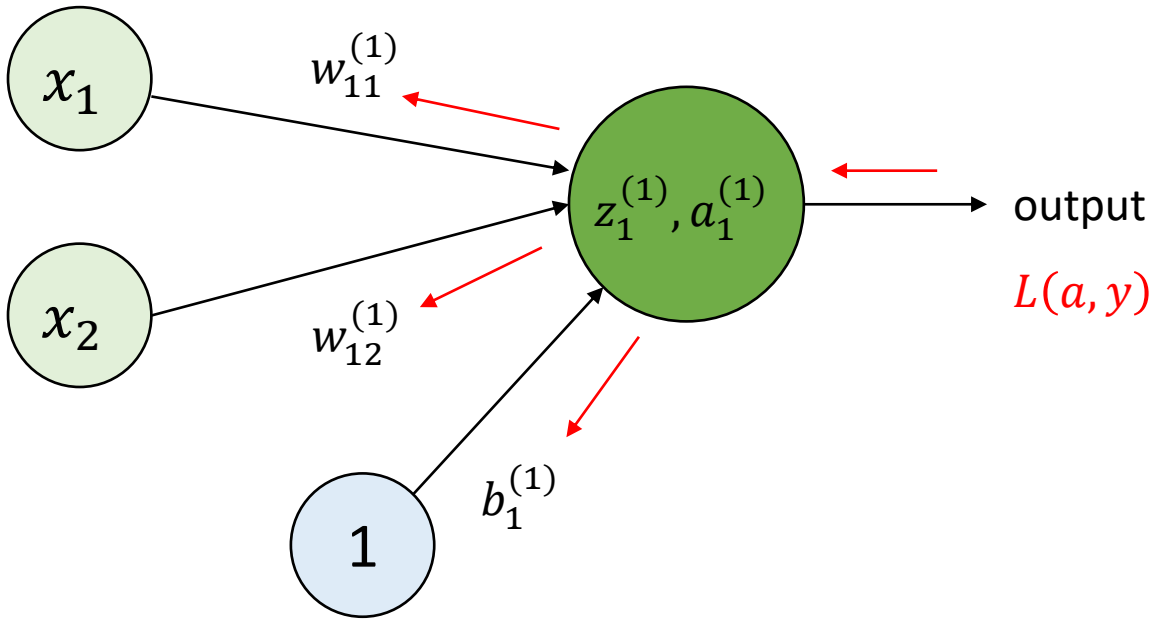



Step 2: Backward propagation

- Using gradient descent to update $b_1^{(1)}$
- $b_1^{(1)} := b_1^{(1)} - \alpha * \frac{\partial L}{\partial b_1^{(1)}} = b_1^{(1)} - \alpha * \left(a_1^{(1)} - \mathrm{y}\right) * a_1^{(1)} * \left(1 - a_1^{(1)}\right)$

# II. Neural Networks | Calculation | Backpropagation

**Example 6:** Backpropagation for a simple NN model.

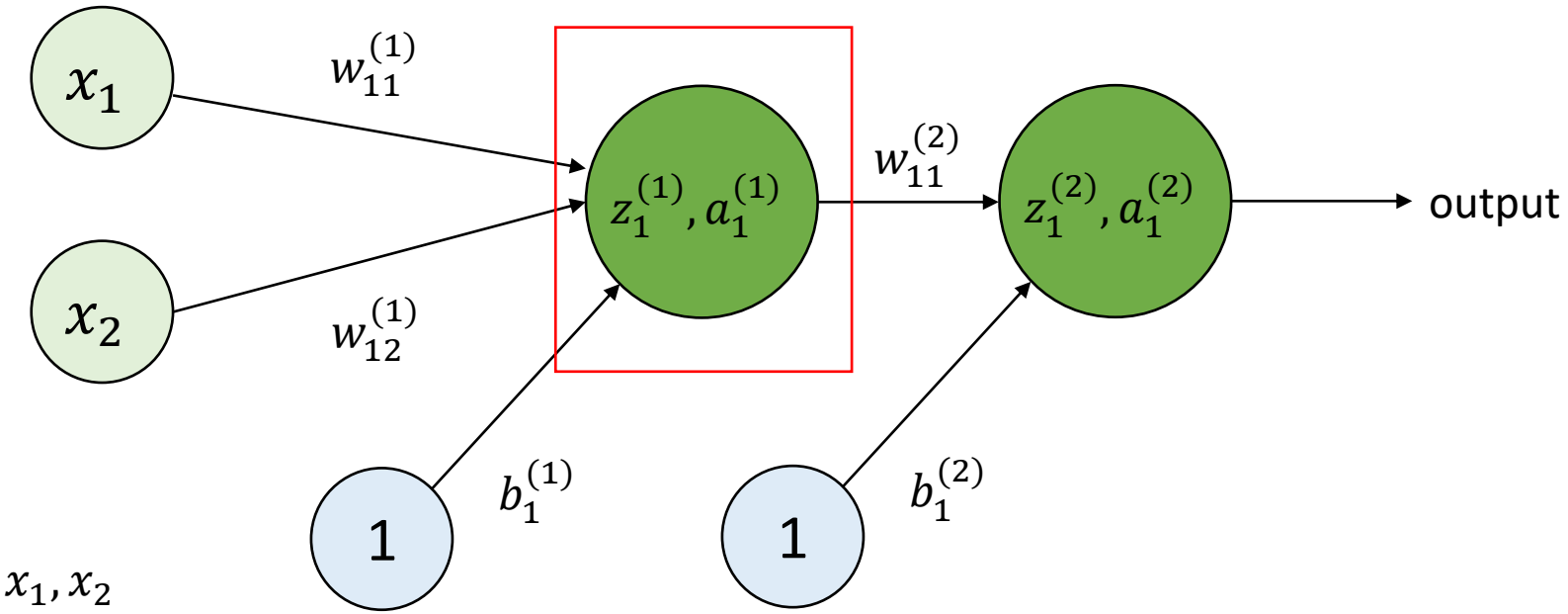



Step 3: Update and repeat

- Actual update the weight and bias
- Repeat Forward then Backward propagation for each single input
- Repeat until converge or hit stopping criteria

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

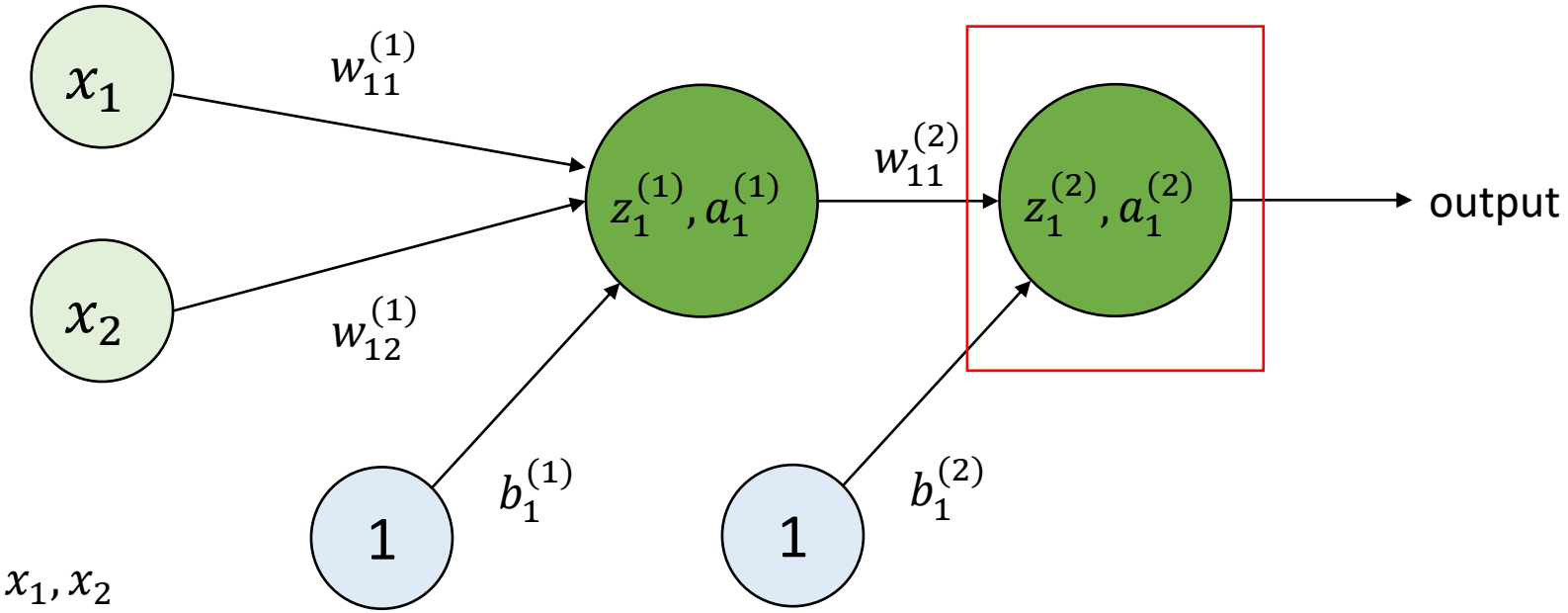



In which:

- $x_1, x_2$ : input neurons of 2 variables $x_1, x_2$
- $w_{11}^{(1)}, w_{12}^{(1)}$ : weights of the hidden neuron in layer 1
- $b_1^{(1)}$ : bias of the hidden neuron in layer 1
- $z_1^{(1)} = w_{11}^{(1)} * x_1 + w_{12}^{(1)} * x_2 + b_1^{(1)}$ : value of the neuron in layer 1
- $a_1^{(1)} = \sigma\left(z_1^{(1)}\right)$ : activate value of the neuron in layer 1, using the sigmoid activation function $\sigma$

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

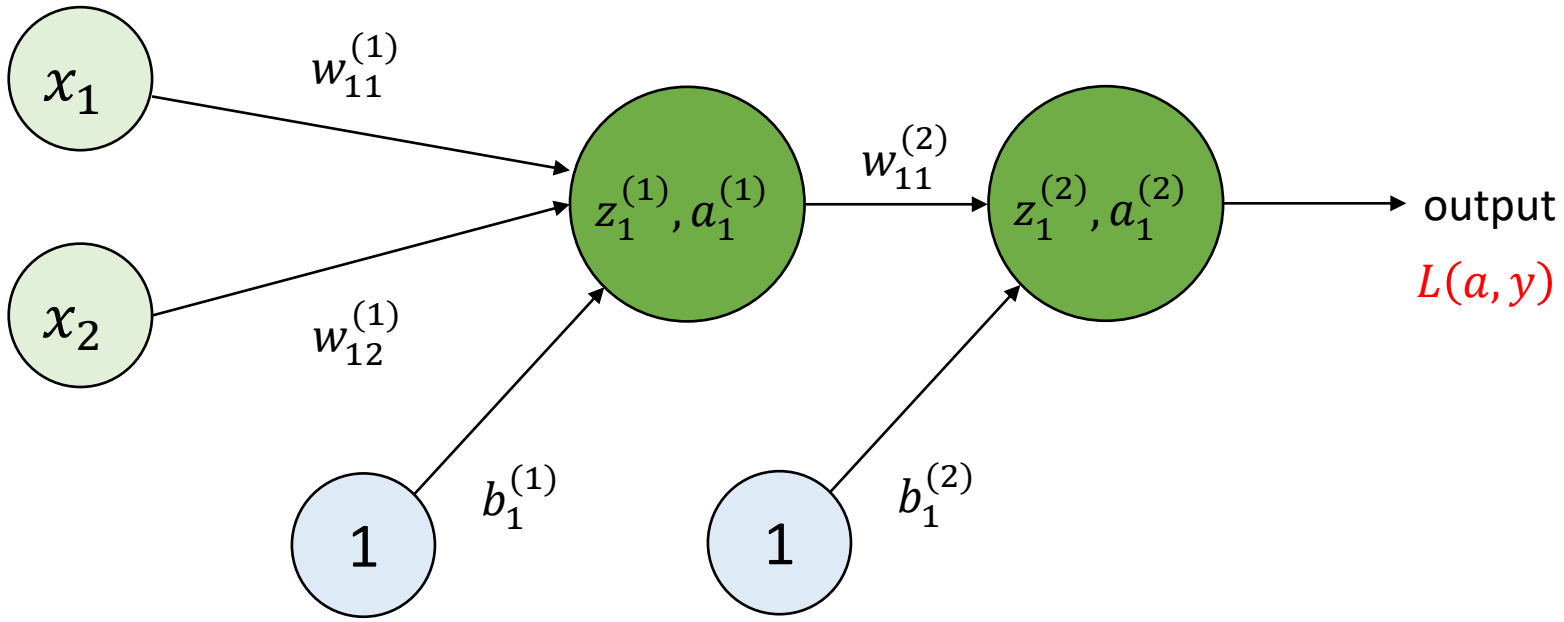



In which:

- $x_1, x_2$ : input neurons of 2 variables $x_1, x_2$
- $w_{11}^{(2)}$,: weights of the output neuron in layer 2
- $b_1^{(2)}$ : bias of the output neuron in layer 2
- $z_1^{(2)} = w_{11}^{(2)} * a_1^{(1)} + b_1^{(2)}$ : value of the neuron in layer 2
- $a_1^{(2)} = \sigma\left(z_1^{(2)}\right)$ : activate value of the neuron in layer 2, using the sigmoid activation function $\sigma$

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

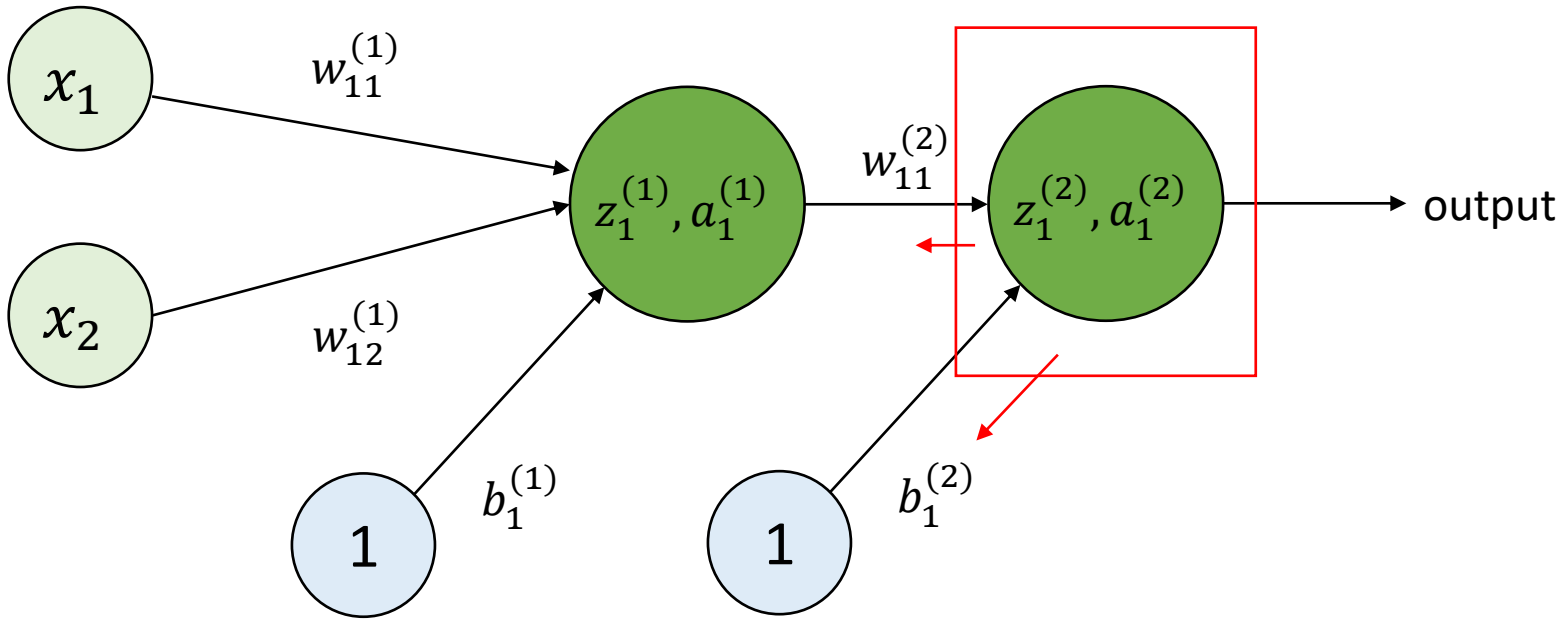



Define the loss, cost function (target function to minimize):

- $L(a, y) = \frac{1}{2} * (a_1^{(2)} - y)^2$
- $C = \sum_{j=1}^{N} L(a_j, y_j)$

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

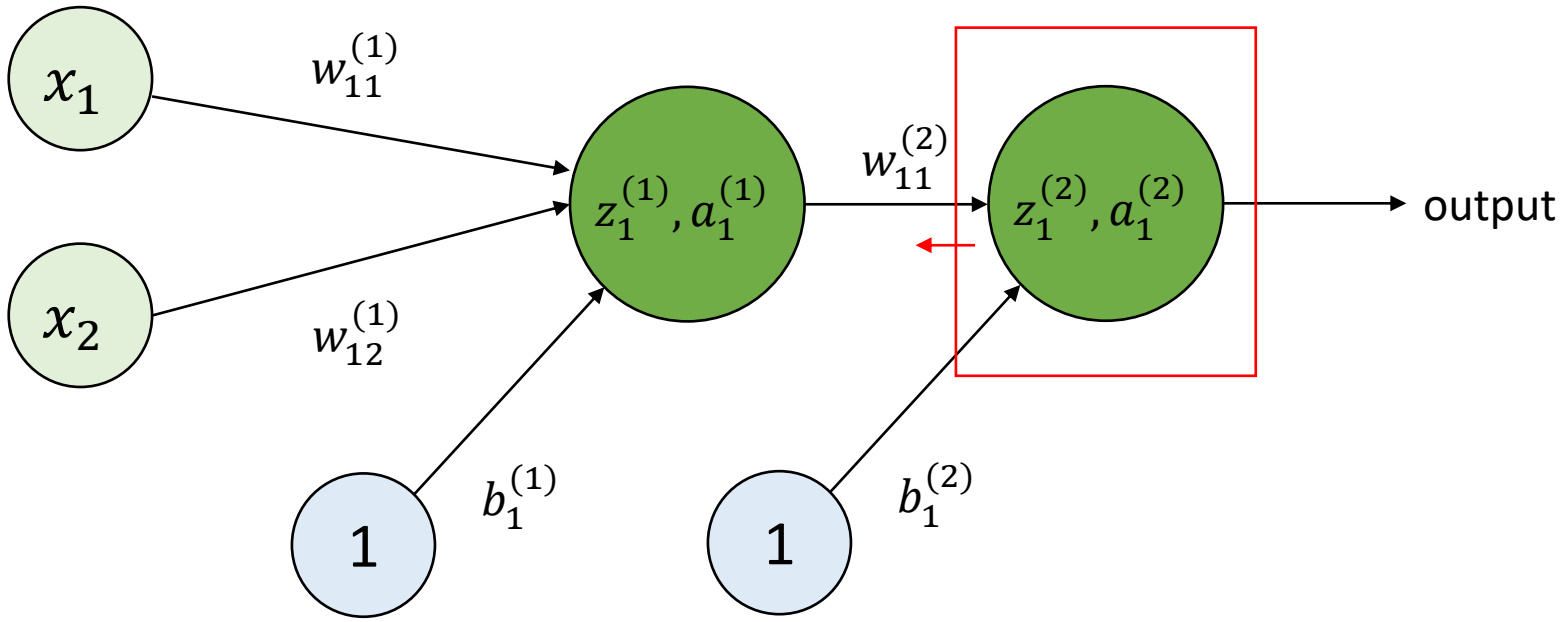



According to Gradient descent, for a single input (learning rate $\alpha$):

- $w_{11}^{(2)} := w_{11}^{(2)} - \alpha * \frac{\partial L}{\partial w_{11}^{(2)}}$
- $b_1^{(2)} := b_1^{(2)} - \alpha * \frac{\partial L}{\partial b_1^{(2)}}$

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

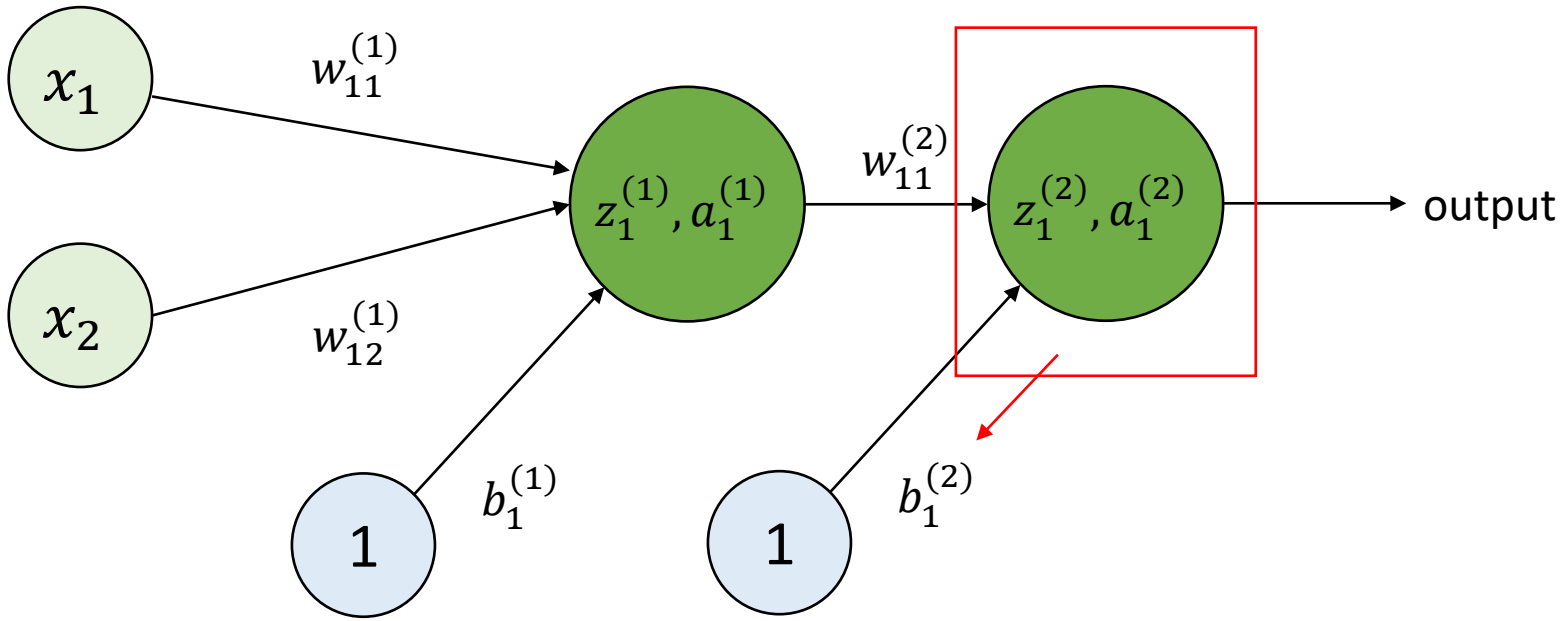



According to Gradient descent, for a single input (learning rate $\alpha$):

- $w_{11}^{(2)} := w_{11}^{(2)} - \alpha * \frac{\partial L}{\partial w_{11}^{(2)}} = w_{11}^{(2)} - \alpha * \frac{\partial L}{\partial a_1^{(2)}} * \frac{\partial a_1^{(2)}}{\partial z_{11}^{(2)}} * \frac{\partial z_{11}^{(2)}}{\partial w_{11}^{(2)}} = w_{11}^{(2)} - \alpha * \left(a_1^{(2)} - \mathrm{y}\right) * a_1^{(2)} * \left(1 - a_1^{(2)}\right) * a_1^{(1)}$

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

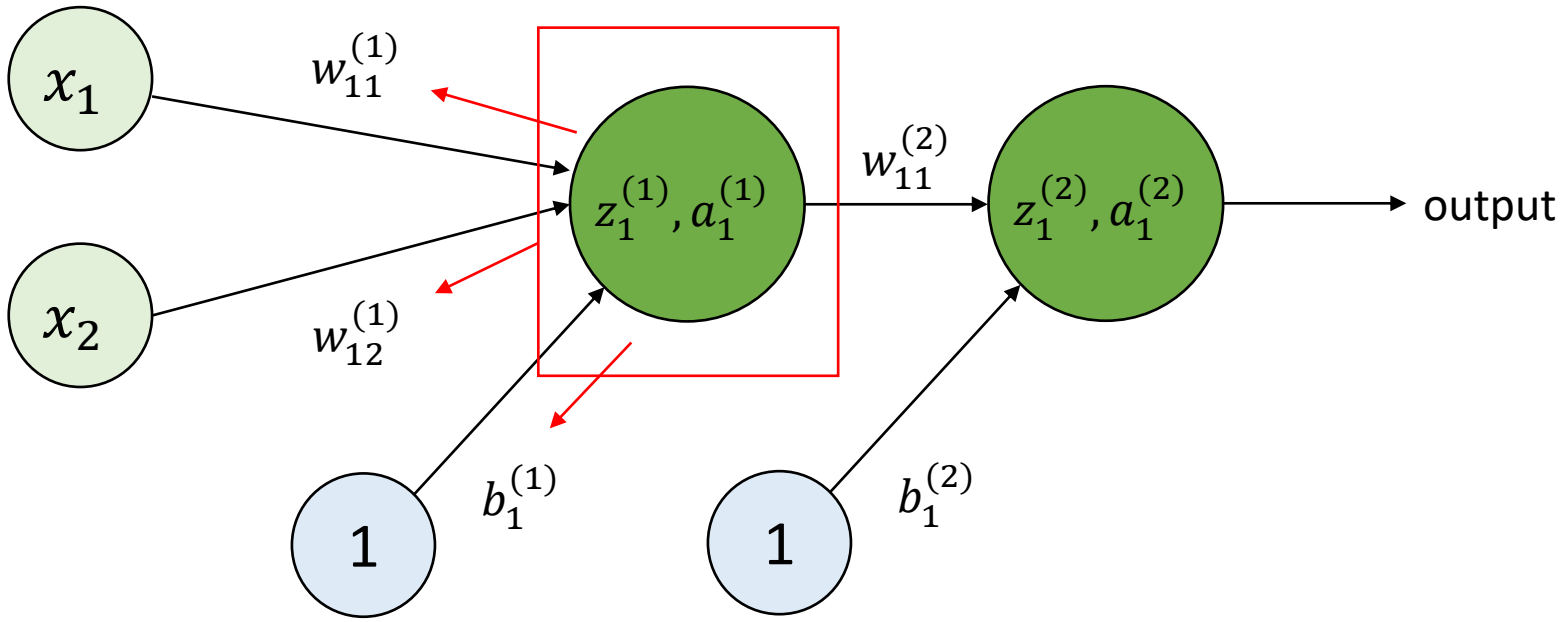



According to Gradient descent, for a single input (learning rate $\alpha$):

- $b_1^{(2)} := b_1^{(2)} - \alpha * \frac{\partial L}{\partial b_1^{(2)}} = w_{11}^{(2)} - \alpha * \frac{\partial L}{\partial a_1^{(2)}} * \frac{\partial a_1^{(2)}}{\partial z_{11}^{(2)}} * \frac{\partial z_{11}^{(2)}}{\partial b_1^{(2)}} = b_1^{(2)} - \alpha * \left(a_1^{(2)} - \mathrm{y}\right) * a_1^{(2)} * \left(1 - a_1^{(2)}\right)$

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

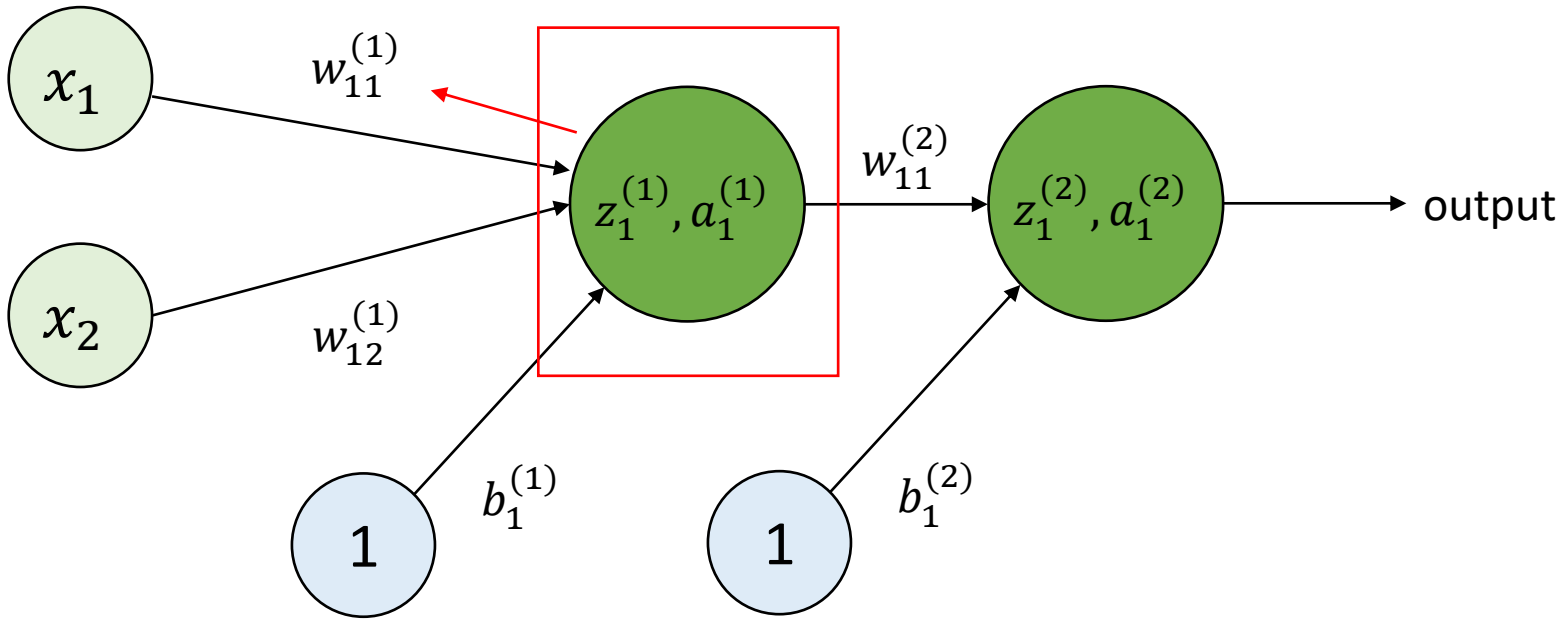



According to Gradient descent, for a single input (learning rate $\alpha$):

- $w_{11}^{(1)} := w_{11}^{(1)} - \alpha * \frac{\partial L}{\partial w_{11}^{(1)}}$
- $w_{12}^{(1)} := w_{12}^{(1)} - \alpha * \frac{\partial L}{\partial w_{12}^{(1)}}$
- $b_1^{(1)} := b_1^{(1)} - \alpha * \frac{\partial L}{\partial b_1^{(1)}}$

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

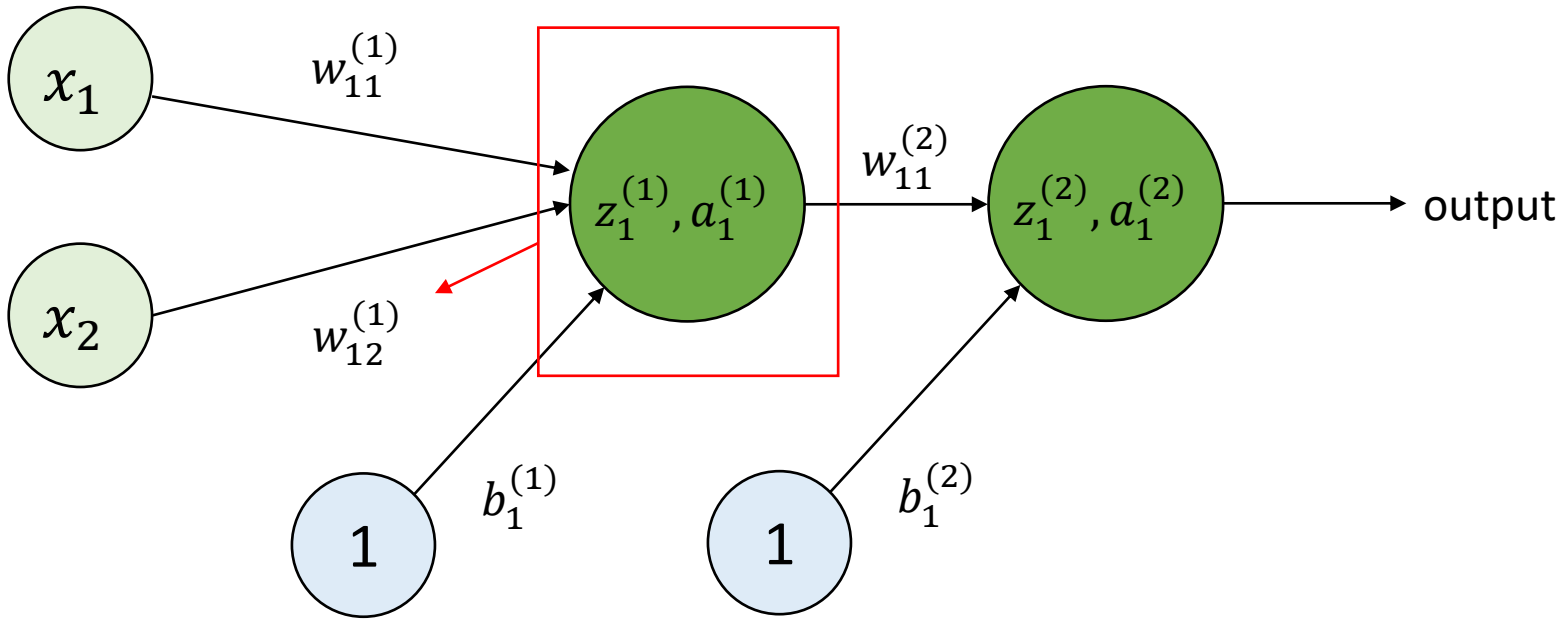



According to Gradient descent, for a single input (learning rate $\alpha$):

- $w_{11}^{(1)} := w_{11}^{(1)} - \alpha * \frac{\partial L}{\partial w_{11}^{(1)}} = w_{11}^{(1)} - \alpha * \frac{\partial L}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} * \frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}} = w_{11}^{(1)} - \alpha * \frac{\partial L}{\partial a_1^{(2)}} * \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} * \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} * \frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}}$

# II. Neural Networks | Calculation | Backpropagation

**Example 7:** Backpropagation for a simple NN model (1 hidden layer).

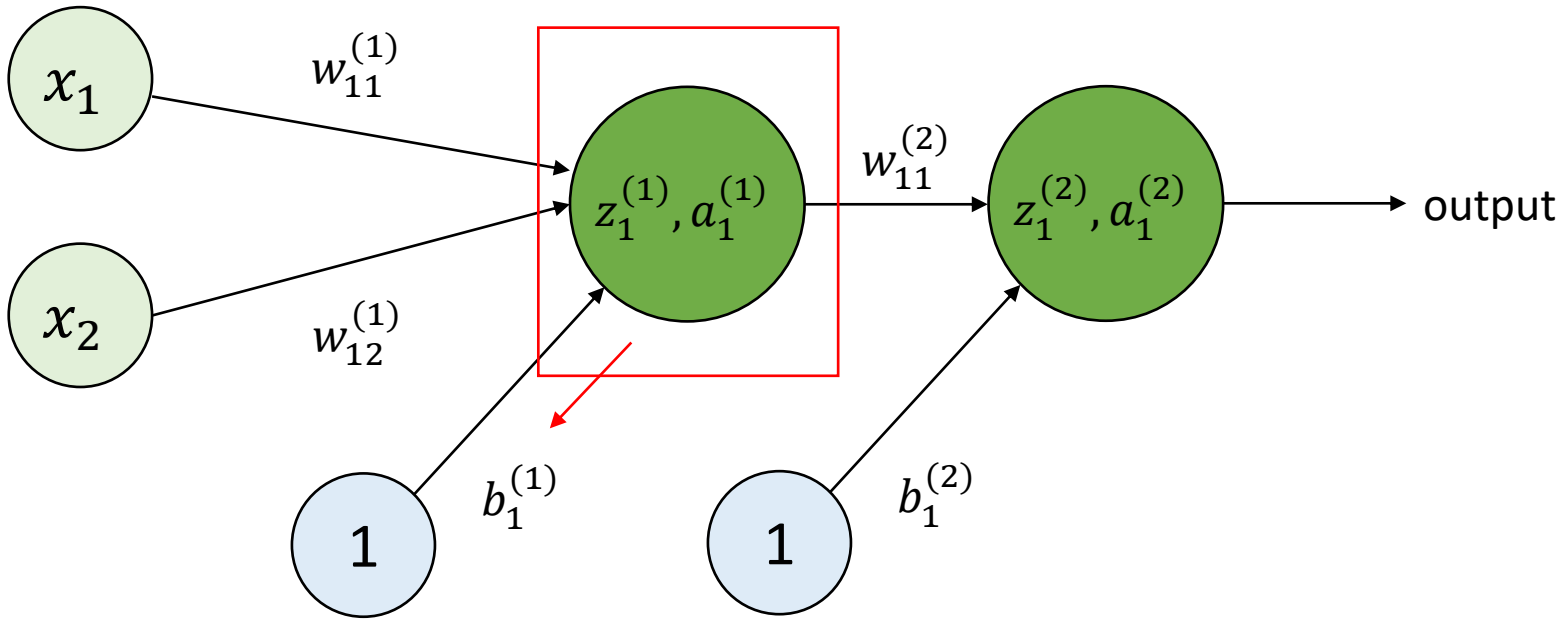



According to Gradient descent, for a single input (learning rate $\alpha$):

- $w_{12}^{(1)} := w_{12}^{(1)} - \alpha * \frac{\partial L}{\partial w_{12}^{(1)}} = w_{12}^{(1)} - \alpha * \frac{\partial L}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} * \frac{\partial z_1^{(1)}}{\partial w_{12}^{(1)}} = w_{12}^{(1)} - \alpha * \frac{\partial L}{\partial a_1^{(2)}} * \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} * \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} * \frac{\partial z_1^{(1)}}{\partial w_{12}^{(1)}}$

# II. Neural Networks | Calculation | Backpropagation

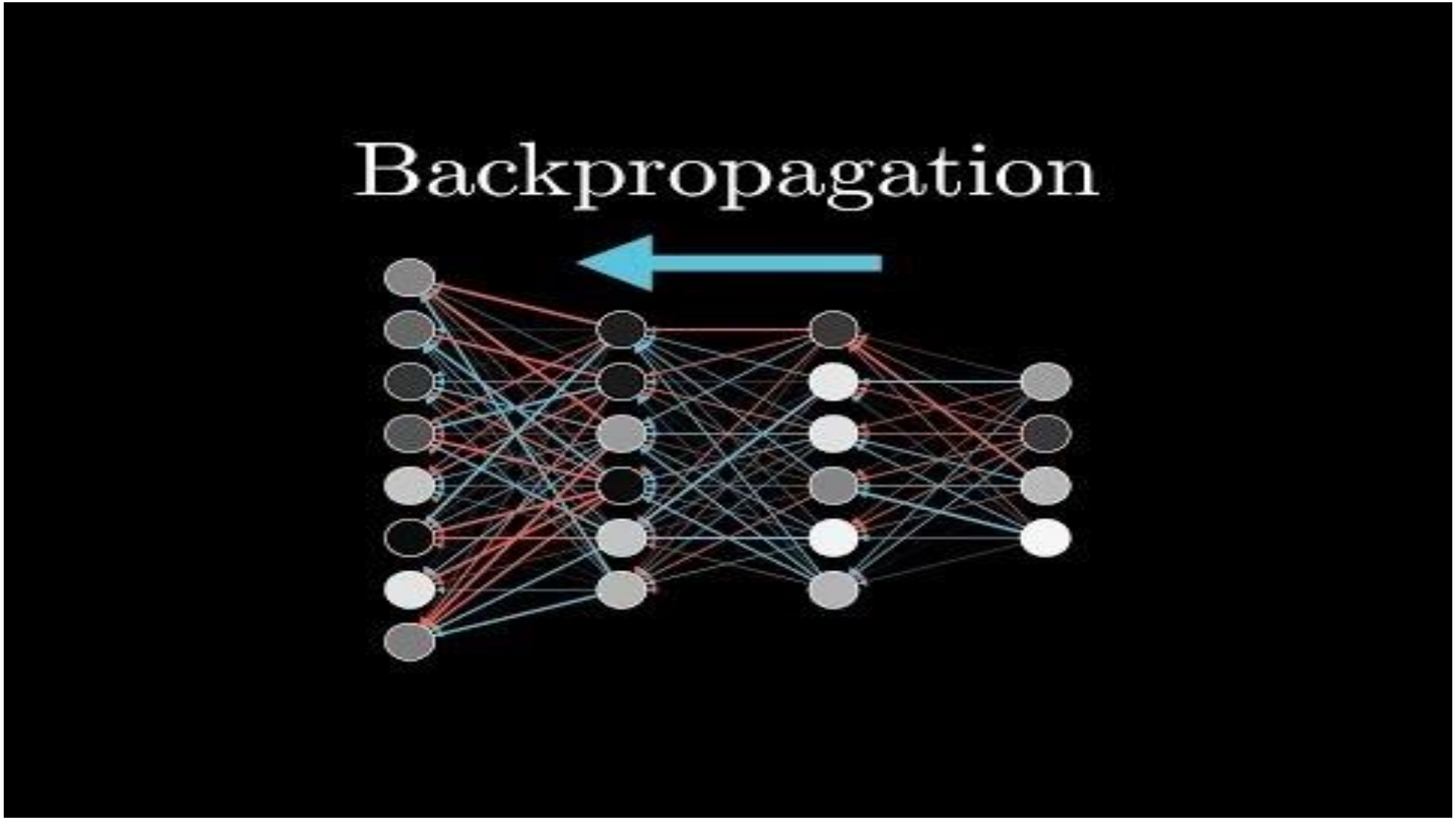
**Example 7:** Backpropagation for a simple NN model (1 hidden layer).




According to Gradient descent, for a single input (learning rate $\alpha$):

- $b_1^{(1)} := b_1^{(1)} - \alpha * \frac{\partial L}{\partial b_1^{(1)}} = b_1^{(1)} - \alpha * \frac{\partial L}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} * \frac{\partial z_1^{(1)}}{\partial b_1^{(1)}} = b_1^{(1)} - \alpha * \frac{\partial L}{\partial a_1^{(2)}} * \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} * \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} * \frac{\partial z_1^{(1)}}{\partial b_1^{(1)}}$

# II. Neural Networks | Backpropagation




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: What is backpropagation really doing? | Chapter 3, deep learning. Link: https://www.youtube.com/watch?v=Ilg3gGewQ5U

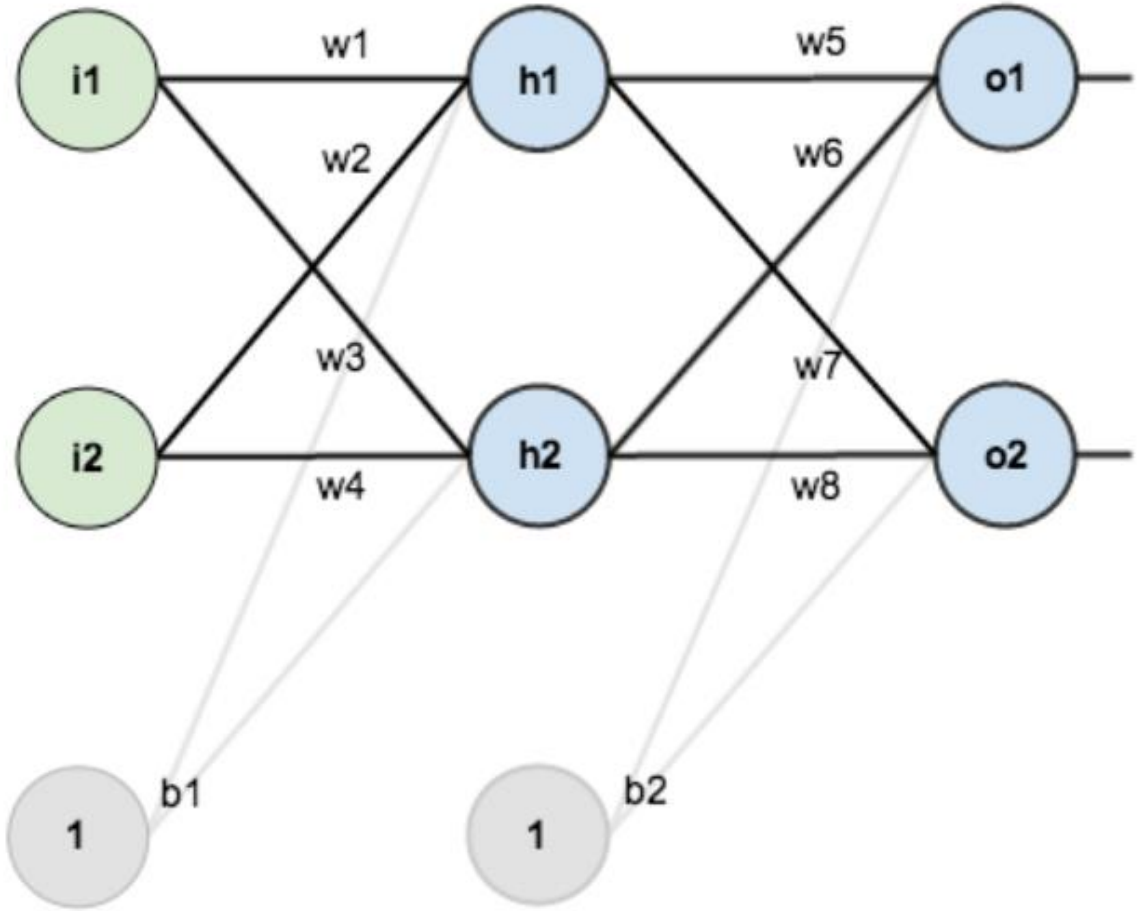
# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$

**Some annotations:**

- $z_{h1}$ : linear combination value of neural h1
- $a_{h1}$ : activation output value of neural h1
- $E_{o1}$ : error at neural output o1
- $E_{total}$ : total output error
- … etc.

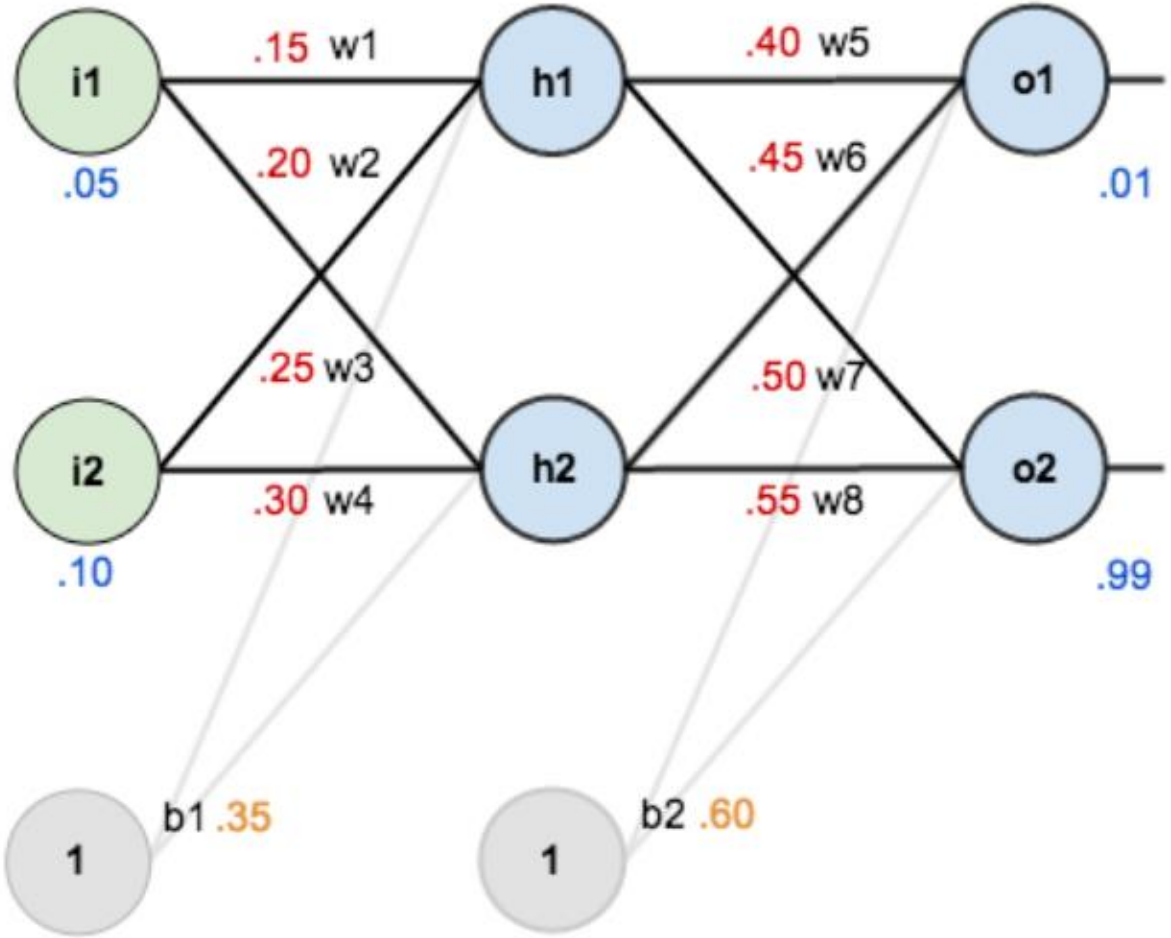



Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 1:

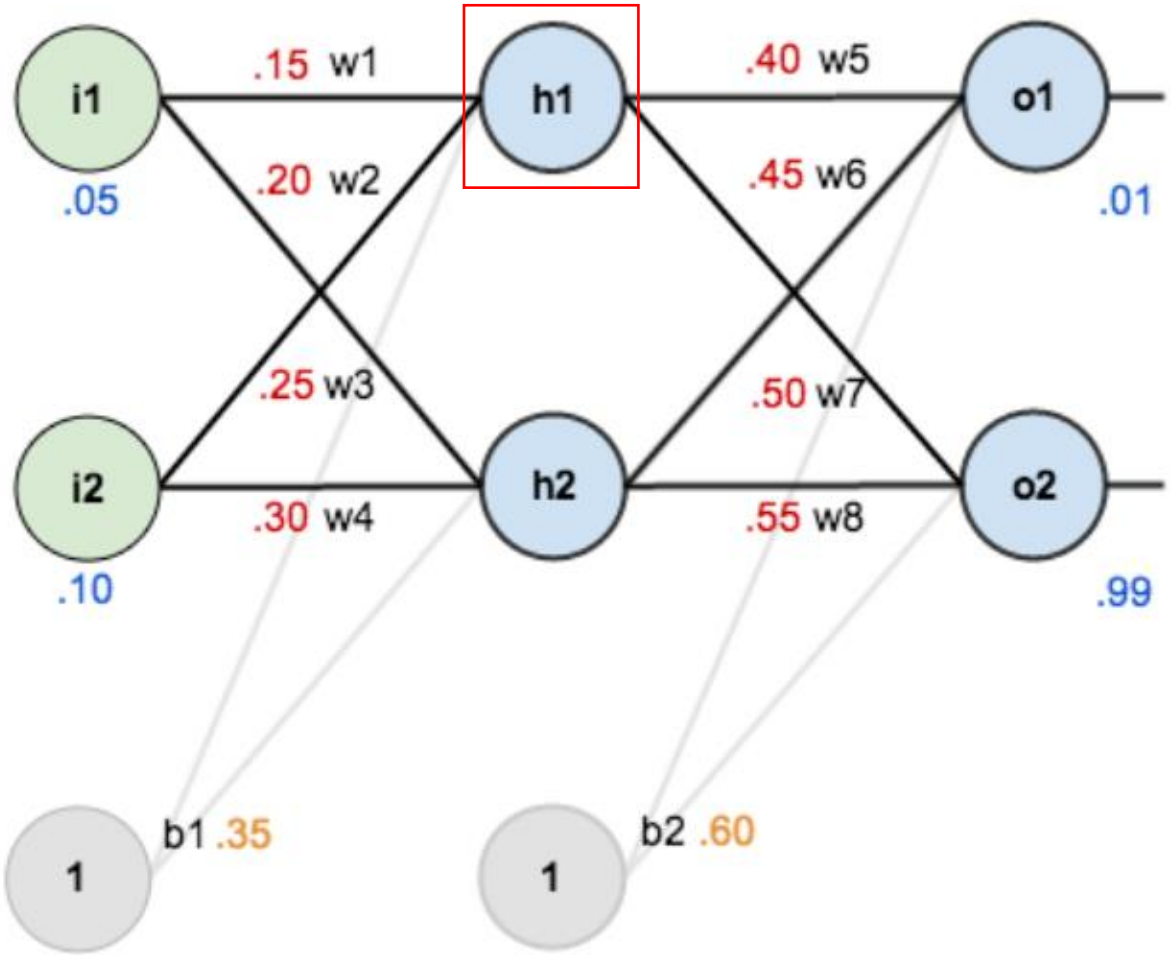
- Single training input
  - $X = (x_1 = 0.05, x_2 = 0.10)$
  - $Y = (y_1 = 0.01, y_2 = 0.99)$
- Random initiate the weight and bias
  - $w1, w2, \dots, w8$
  - $b1, b2$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 2:

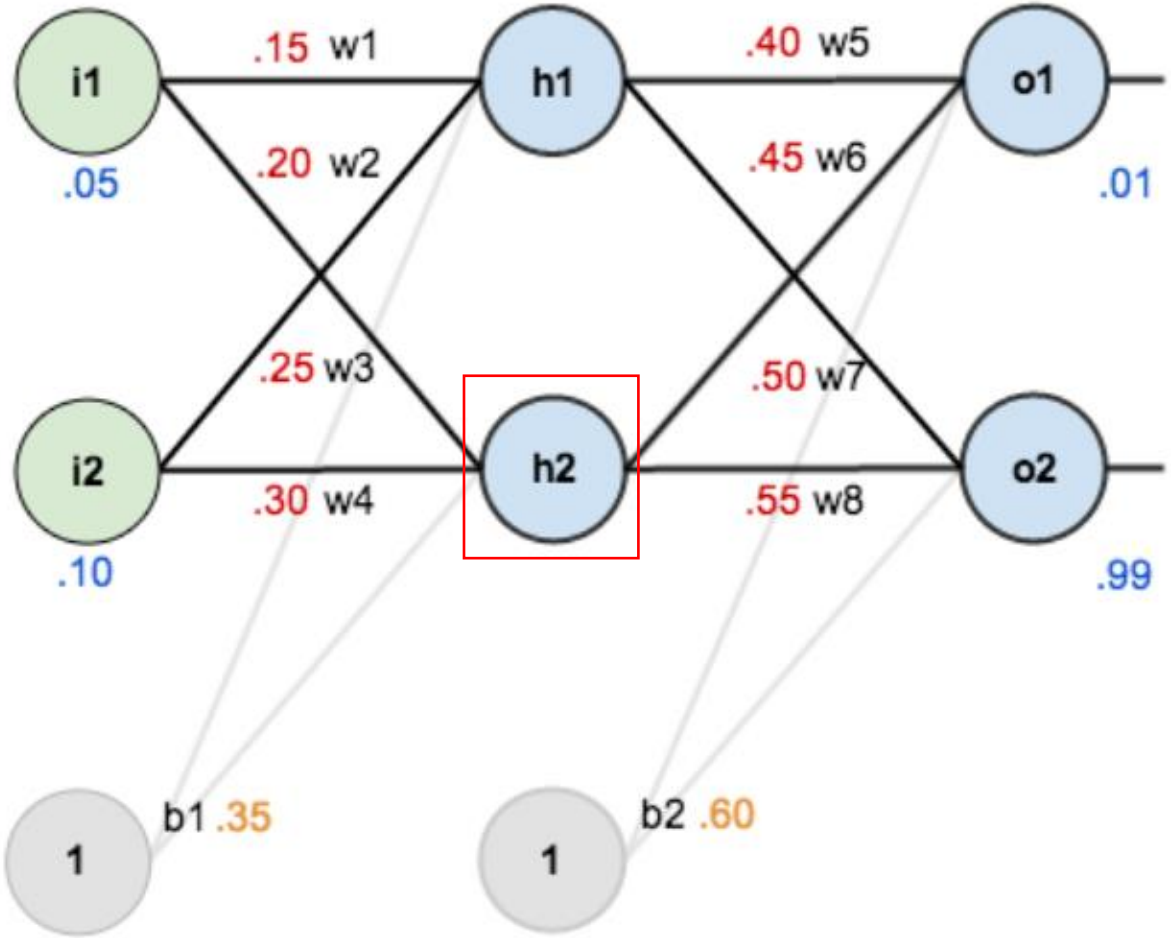
- Forward propagation
  - $z_{h1} = w1 * a_{i1} + w2 * a_{i2} + b1 * 1$
    $= 0.15 * 0.05 + 0.20 * 0.10 + 0.35 * 1$
    $= 0.3775$
  - $a_{h1} = sigmoid(z_{h1}) = 1/(1 + \exp(-z_{h1}))$
    $= 1/(1 + \exp(-0.3775)$
    $= 0.5933$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 2:

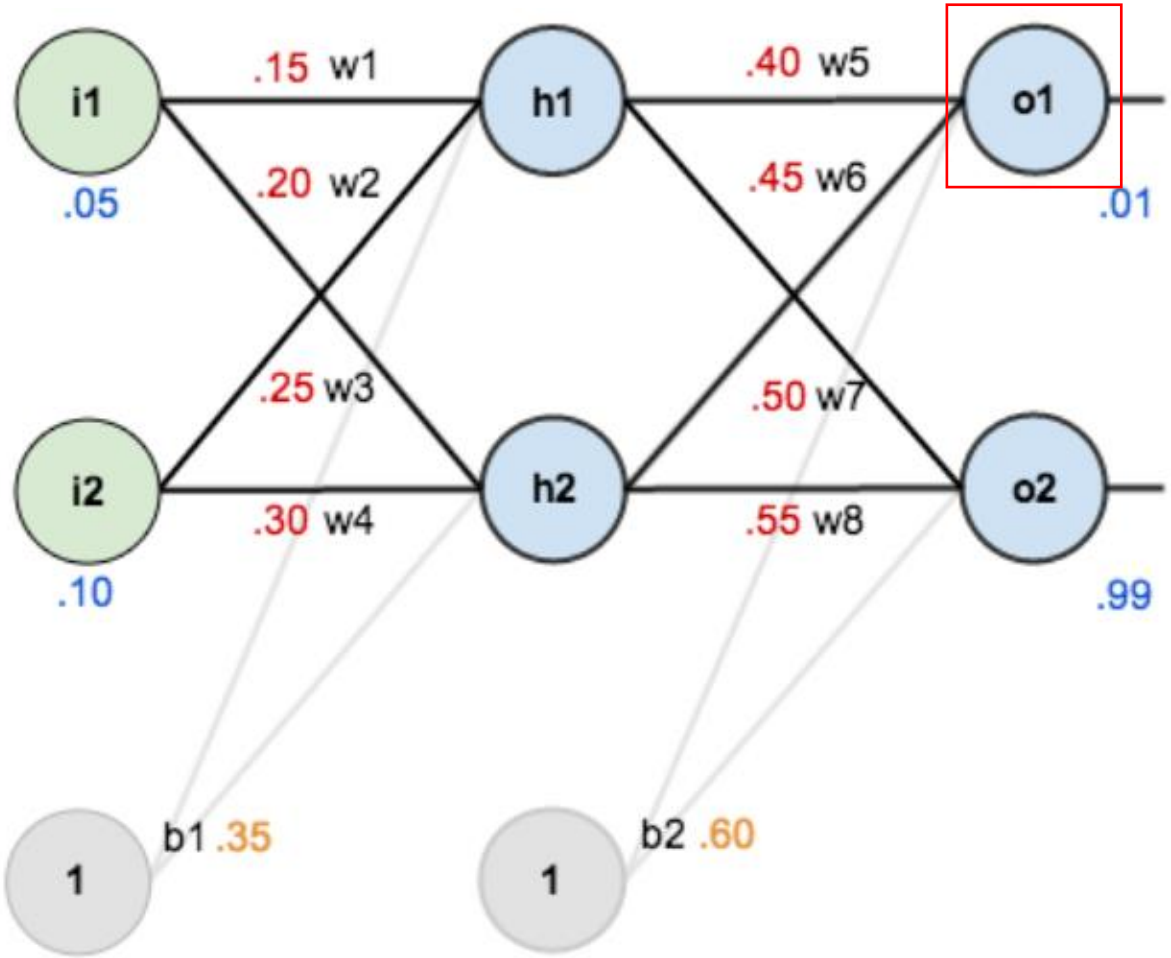
- Forward propagation
  - $z_{h2} = w3 * a_{i1} + w4 * a_{i2} + b1 * 1$
    $= 0.25 * 0.05 + 0.30 * 0.10 + 0.35 * 1$
    $= 0.3925$
  - $a_{h2} = sigmoid(z_{h2}) = 1/(1 + \exp(-z_{h2}))$
    $= 1/(1 + \exp(-0.3925)$
    $= 0.5969$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 2:

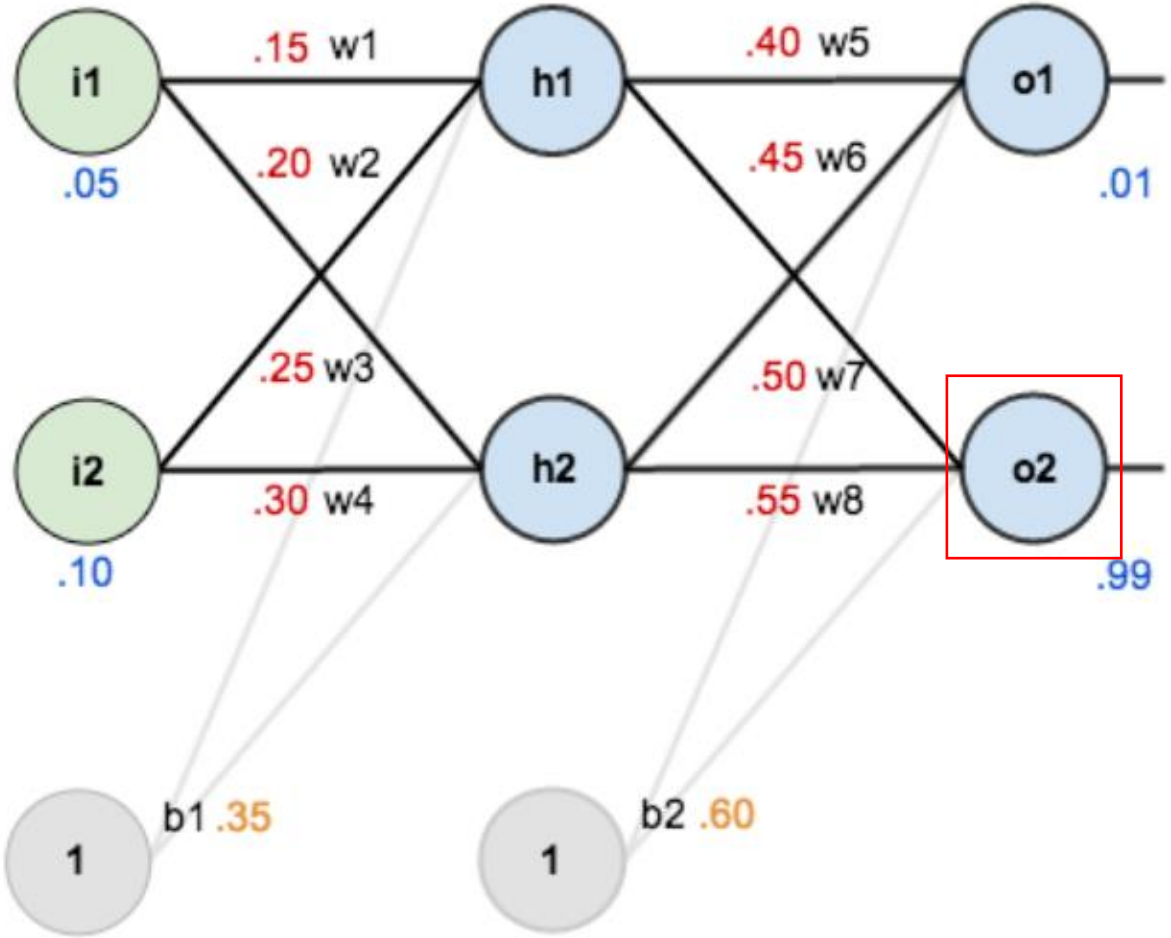
- Forward propagation
  - $z_{o1} = w5 * a_{h1} + w6 * a_{h2} + b2 * 1$
    $= 0.40 * 0.5933 + 0.45 * 0.5969 + 0.60 * 1$
    $= 1.1059$
  - $a_{o1} = sigmoid(z_{o1}) = 1/(1 + \exp(-z_{o1}))$
    $= 1/(1 + \exp(-1.1059)$
    $= 0.7514$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 2:

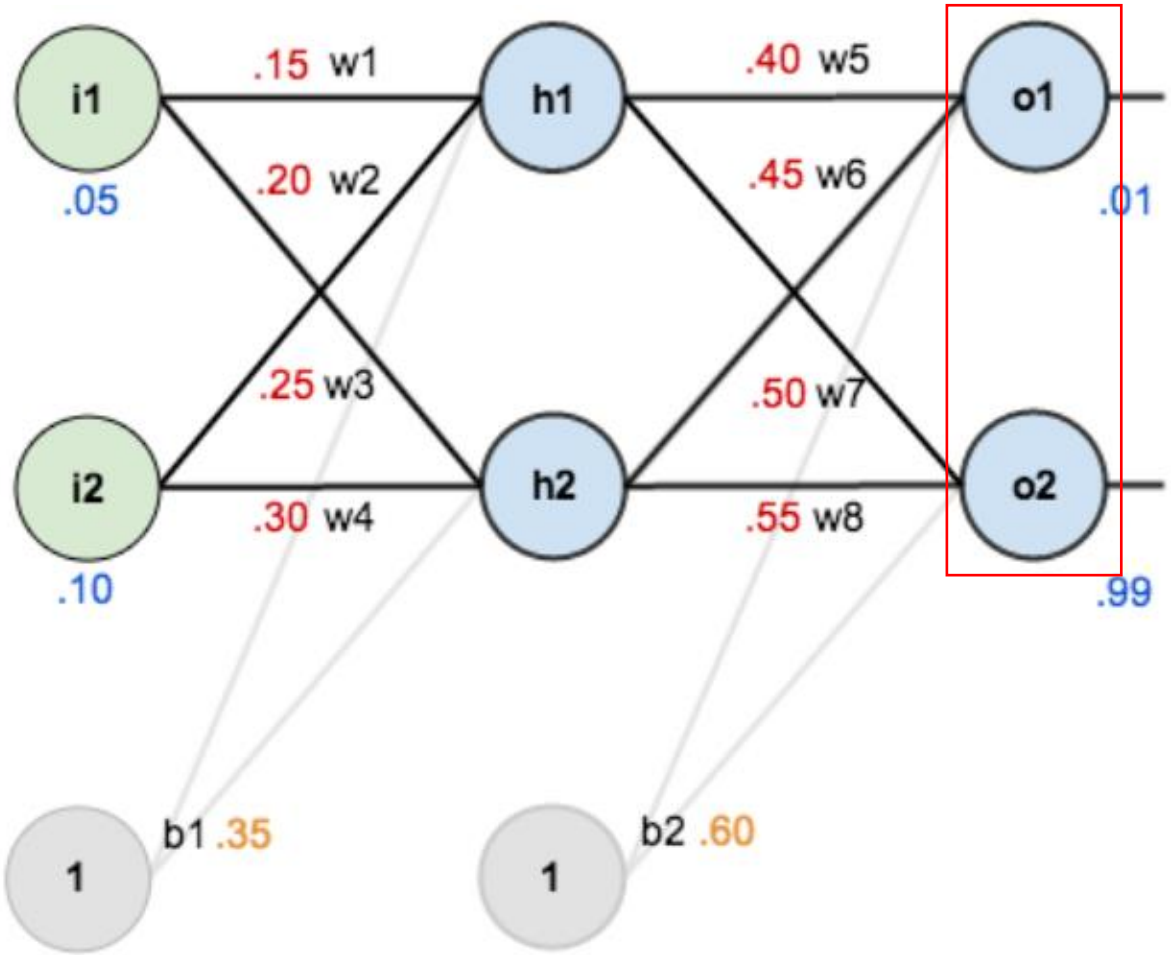
- Forward propagation
  - $z_{o2} = w7 * a_{h1} + w8 * a_{h2} + b2 * 1$
    $= 0.50 * 0.5933 + 0.55 * 0.5969 + 0.60 * 1$
    $= 1.2249$
  - $a_{o2} = sigmoid(z_{o2}) = 1/(1 + \exp(-z_{o2}))$
    $= 1/(1 + \exp(-1.2249)$
    $= 0.7729$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 3:

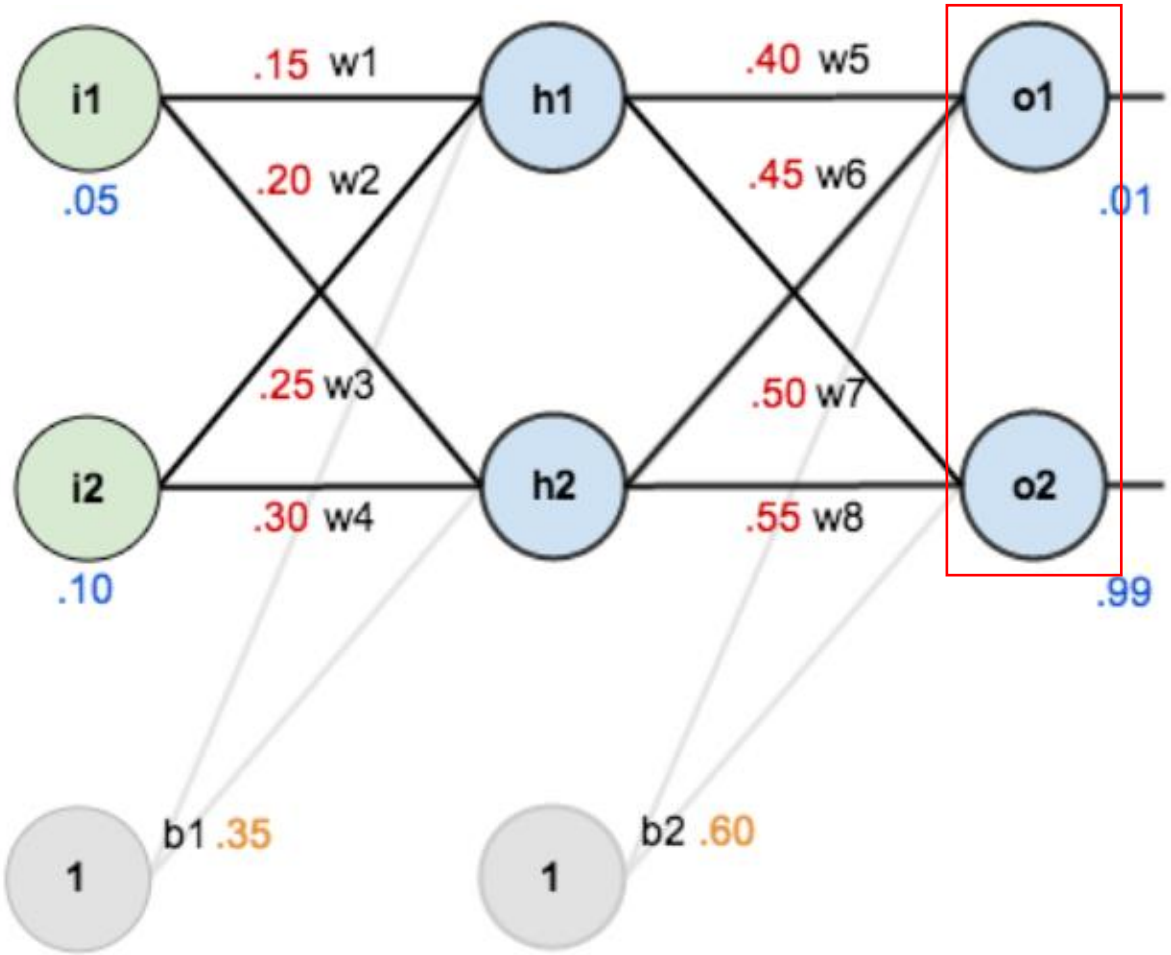
- Total output error (Sum Squared Error)
  - $E_{total} = \frac{1}{2} * \sum(actual - output)^2$
    $= \frac{1}{2} * [\ (0.01 - a_{o1})^2 + (0.99 - a_{o2})^2\ ]$
    $= 0.2984$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 4:

- Backward propagation for output layer
  - Gradient descent update for w5

$$w5^{+} \leftarrow w5 - \alpha * \frac{\partial E_{total}}{\partial w5}$$

  - Chain rule for partial derivative

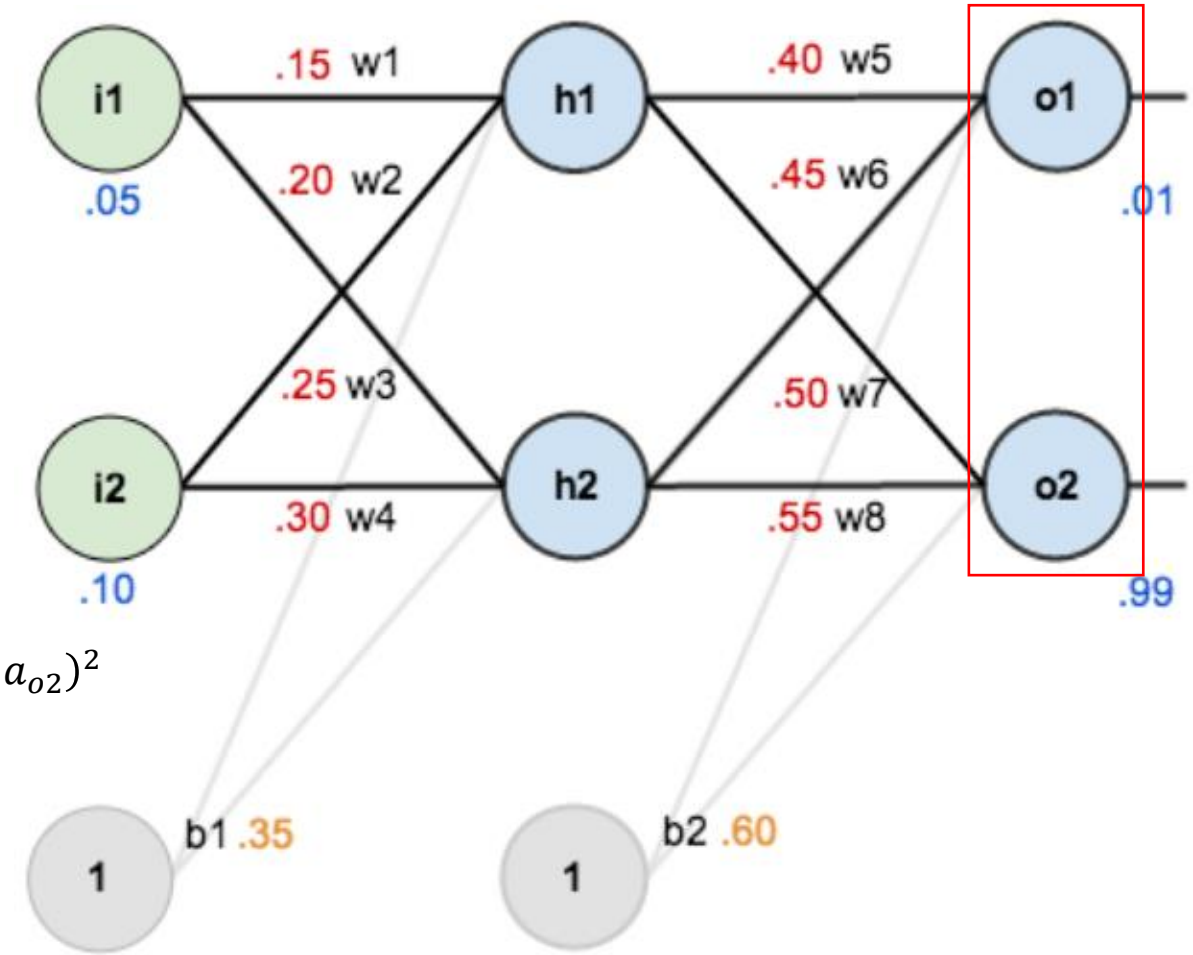
$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial z_{o1}} * \frac{\partial z_{o1}}{\partial w_5}$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 4:

- Backward propagation for output layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial z_{o1}} * \frac{\partial z_{o1}}{\partial w_5}$$

  - Calculate the partial derivative $\frac{\partial E_{total}}{\partial a_{o1}}$
    - $E_{total} = \frac{1}{2} * (actual_{o1} - a_{o1})^2 + \frac{1}{2} * (actual_{o2} - a_{o2})^2$
    - $\frac{\partial E_{total}}{\partial a_{o1}} = \frac{1}{2} * 2 * (actual_{o1} - a_{o1}) * -1 + 0$

$$= -(actual_{o1} - a_{o1})$$

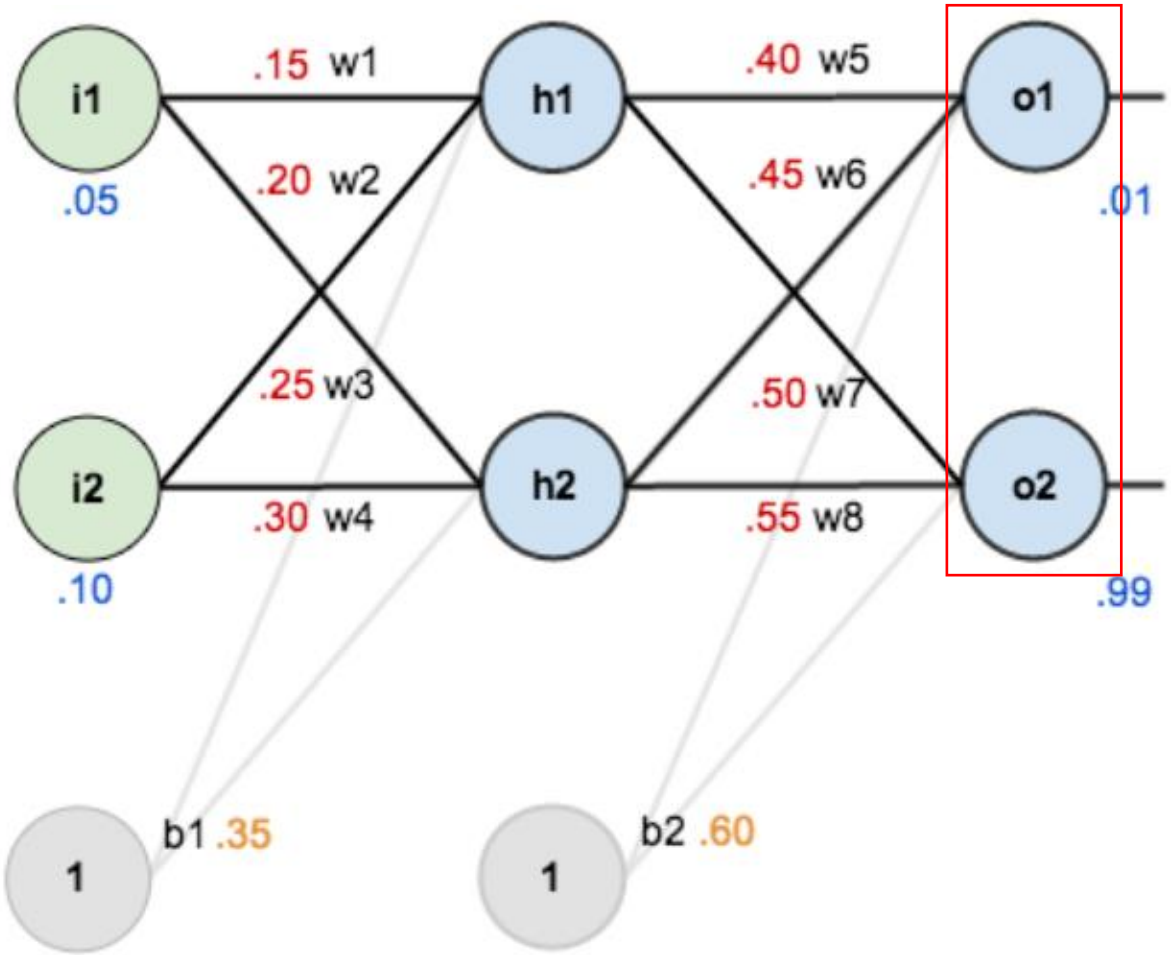
$$= -(0.01 - 0.7514) = 0.7414$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 4:

- Backward propagation for output layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial z_{o1}} * \frac{\partial z_{o1}}{\partial w_5}$$

  - Calculate the partial derivative $\frac{\partial a_{o1}}{\partial z_{o1}}$
    - $a_{o1} = 1/(1 + \exp(-z_{o1})$
    - $\frac{\partial a_{o1}}{\partial z_{o1}} = a_{o1} * (1 - a_{o1})$

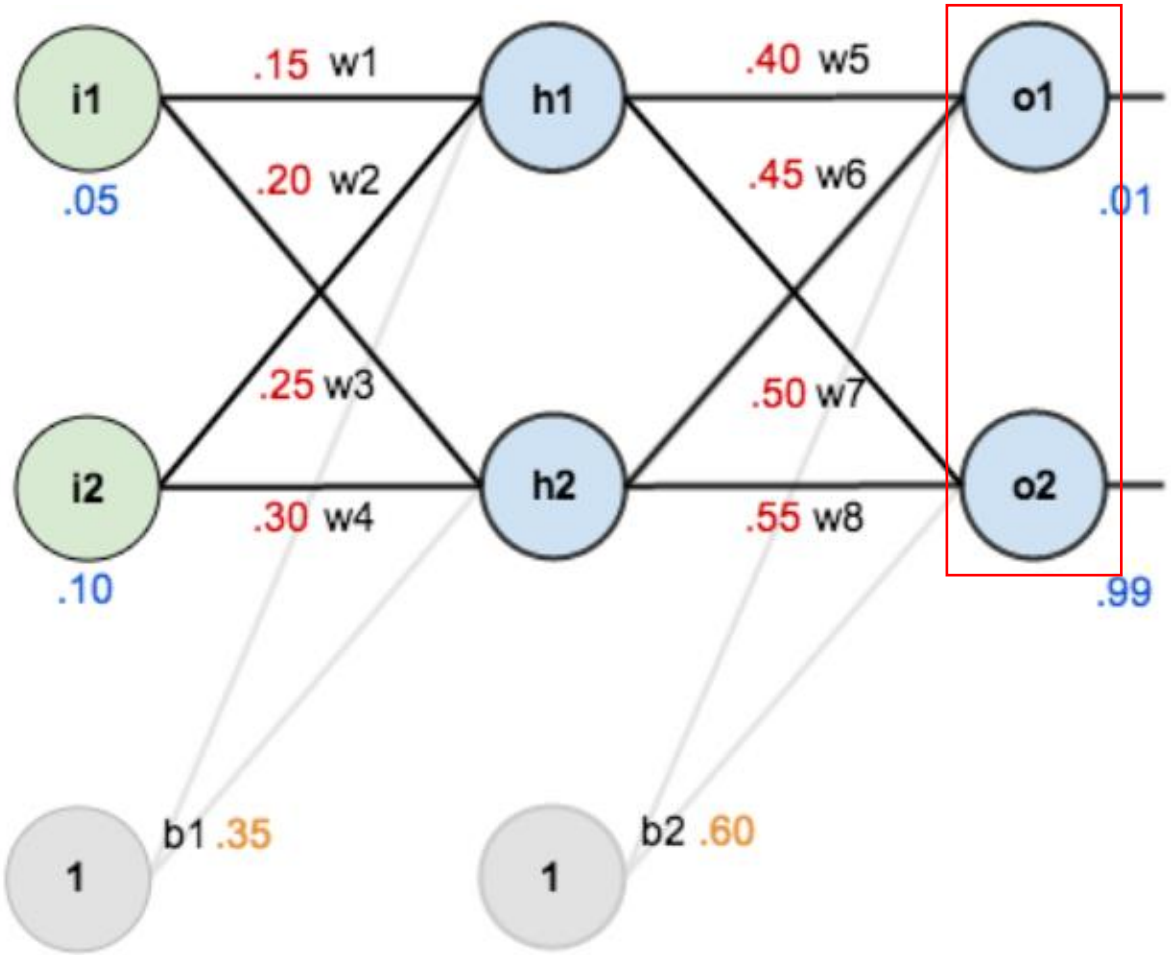
$$= 0.7514 * (1 - 07514) = 0.1868$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 4:

- Backward propagation for output layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial z_{o1}} * \frac{\partial z_{o1}}{\partial w_5}$$

  - Calculate the partial derivative $\frac{\partial z_{o1}}{\partial w_5}$
    - $z_{o1} = w5 * a_{h1} + w6 * a_{h2} + b2 * 1$
    - $\frac{\partial z_{o1}}{\partial w_5} = 1 * a_{h1} + 0 + 0$

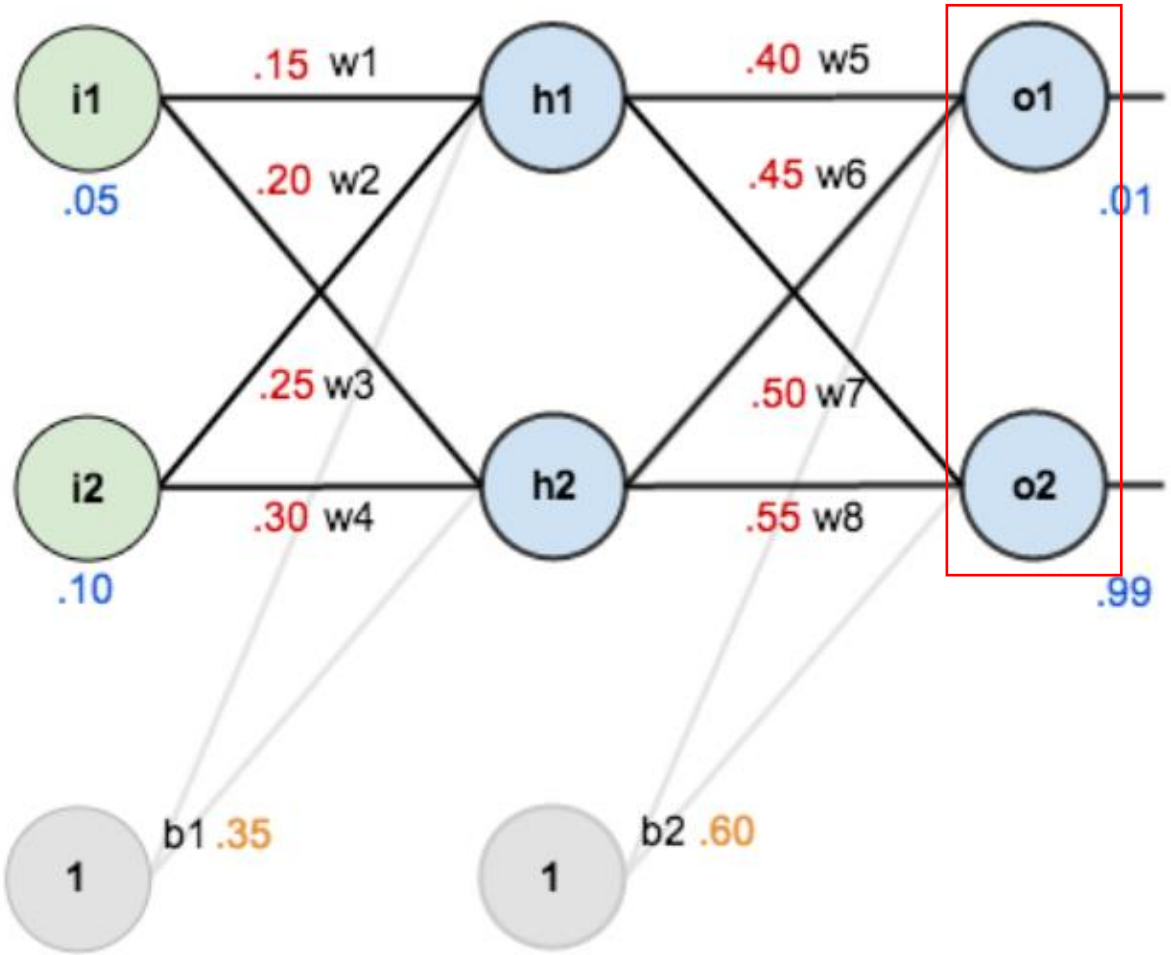
$$= 0.5933$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 4:

- Backward propagation for output layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w5} = \frac{\partial E_{total}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial z_{o1}} * \frac{\partial z_{o1}}{\partial w_5}$$
$$= 0.7414 * 0.1868 * 0.5933$$
$$= .0820$$

  - Gradient descent update for w5

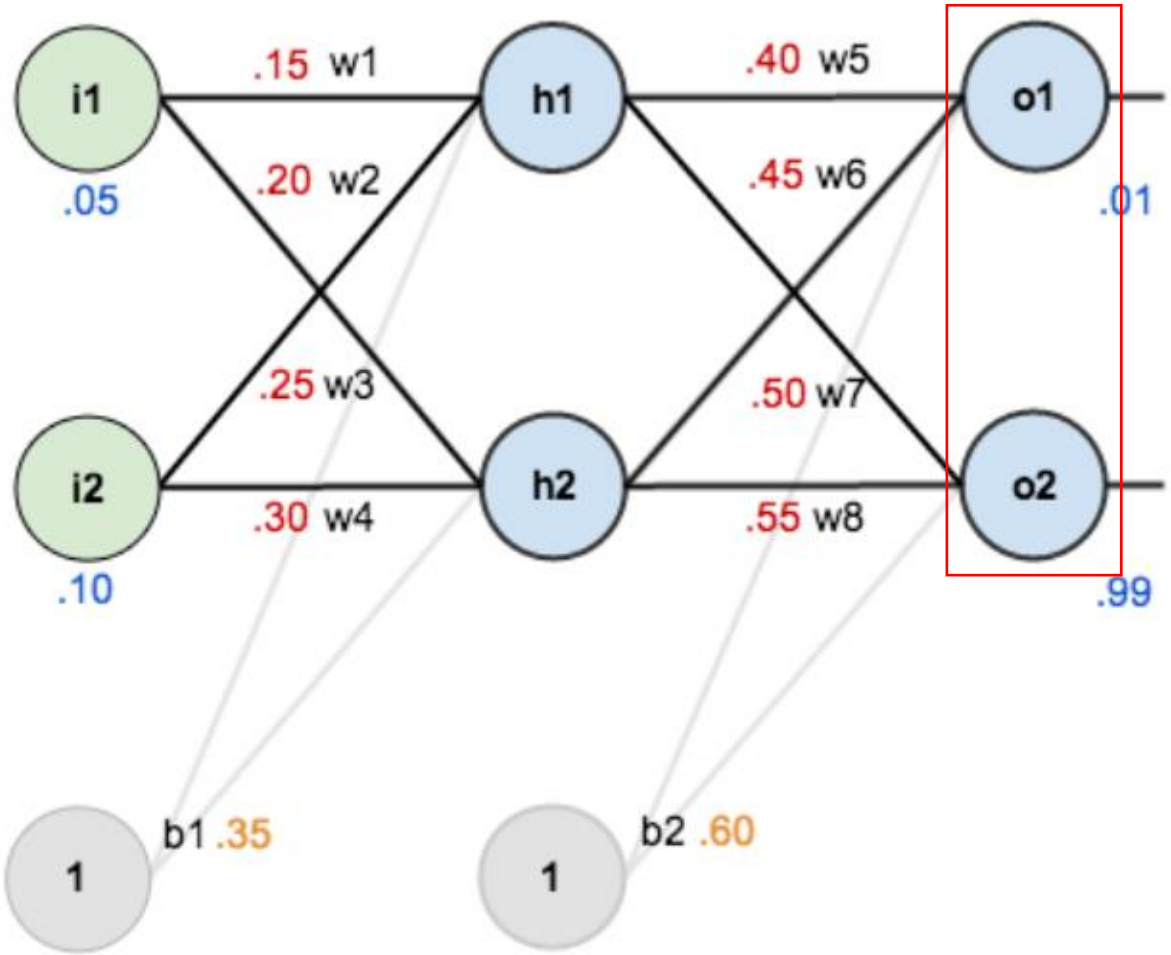
$$w5^{+} \leftarrow w5 - \alpha * \frac{\partial E_{total}}{\partial w5}$$
$$w5^{+} \leftarrow 0.40 - 0.5 * 0.0820 = 0.3590$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 4:

- Backward propagation for output layer
  - Following the same process

$$w5^{+} \leftarrow w5 - \alpha * \frac{\partial E_{total}}{\partial w5} = 0.3590$$

$$w6^{+} \leftarrow w6 - \alpha * \frac{\partial E_{total}}{\partial w6} = 0.4087$$

$$w7^{+} \leftarrow w7 - \alpha * \frac{\partial E_{total}}{\partial w7} = 0.5113$$

$$w8^{+} \leftarrow w8 - \alpha * \frac{\partial E_{total}}{\partial w8} = 0.5617$$

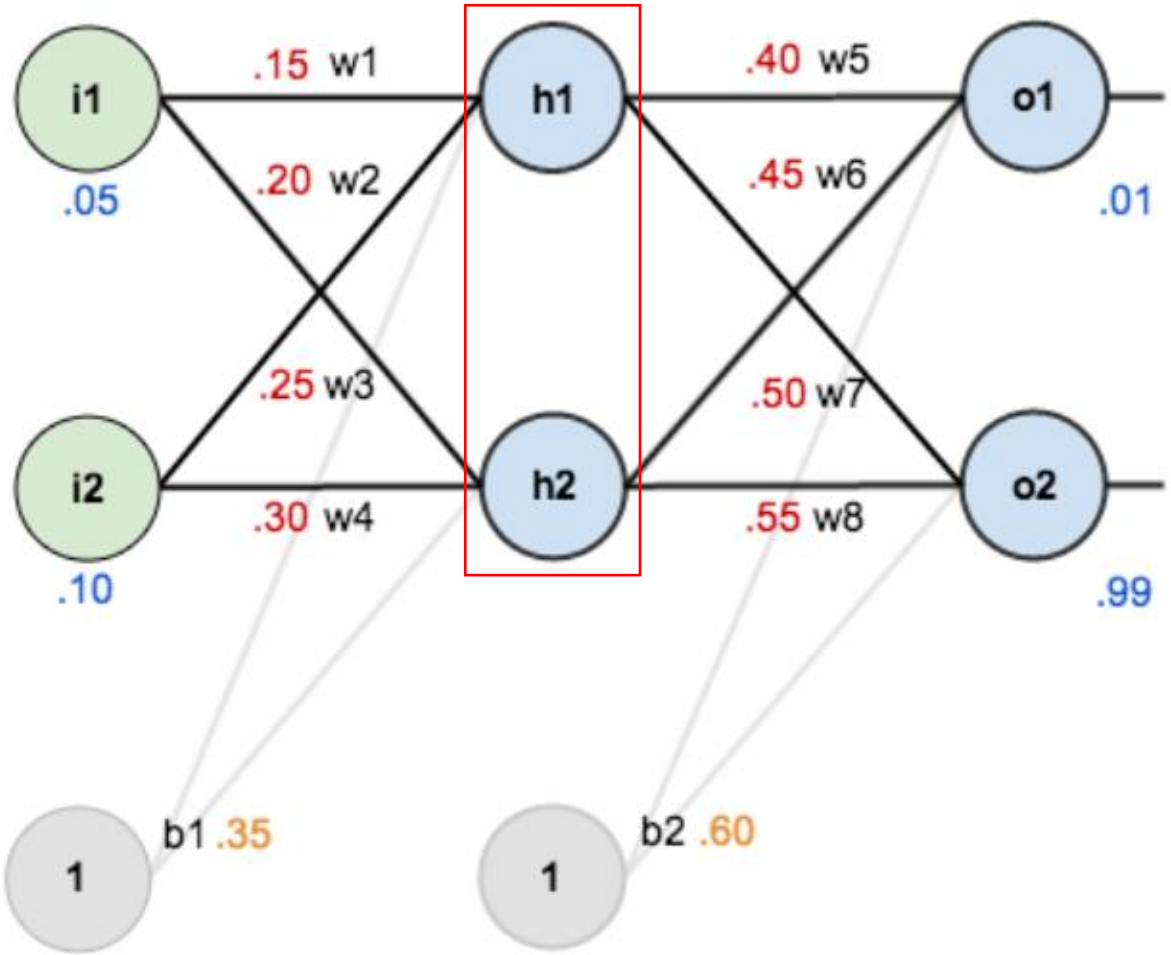
  - Note: the updating of w and b happens actually after the back-propagation step.

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Gradient descent update for w1

$$w1^{+} \leftarrow w1 - \alpha * \frac{\partial E_{total}}{\partial w1}$$

  - Chain rule for partial derivative

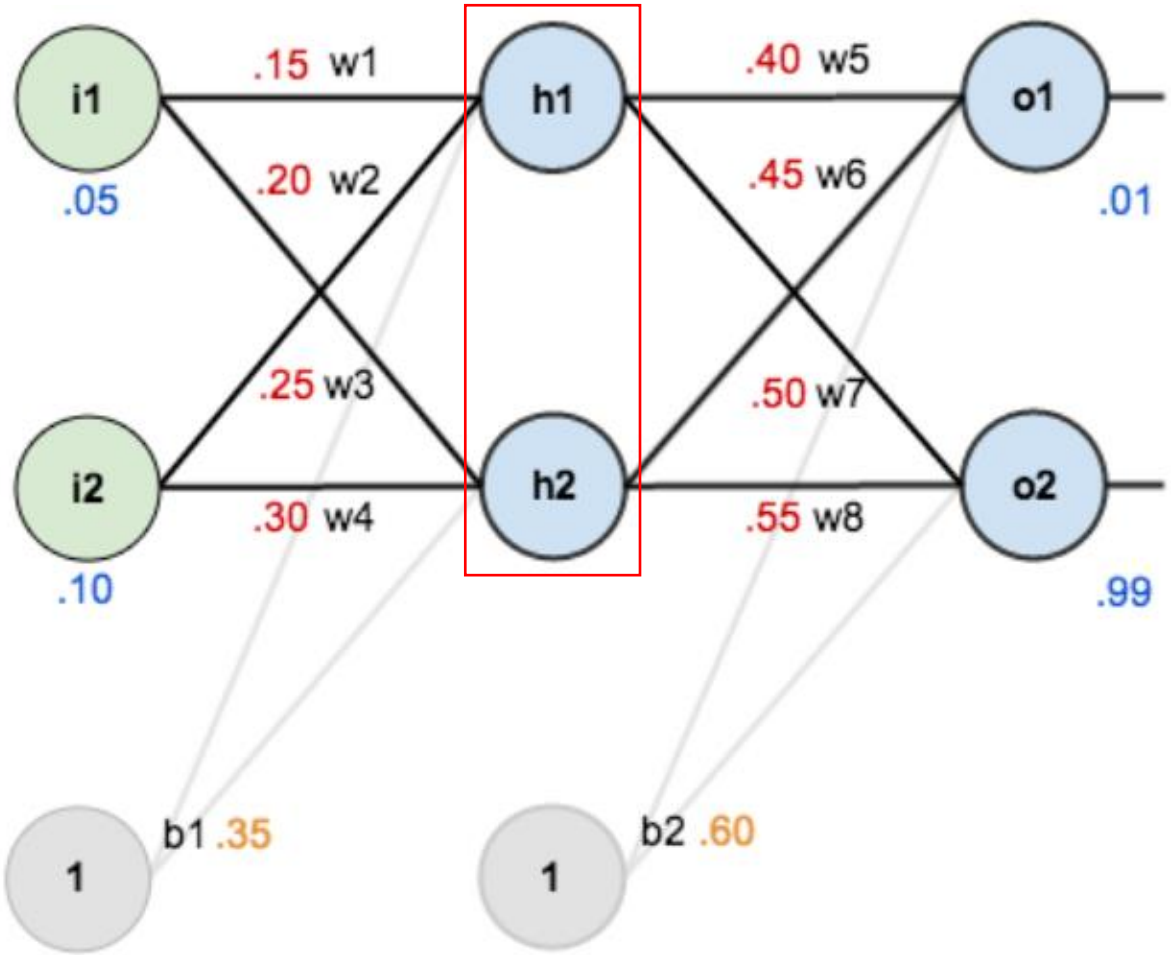
$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial z_{h1}} * \frac{\partial z_{h1}}{\partial w_1}$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial z_{h1}} * \frac{\partial z_{h1}}{\partial w_1}$$

  - Calculate the partial derivative $\frac{\partial E_{total}}{\partial a_{h1}}$

$$E_{total} = E_{o1} + E_{o2}$$

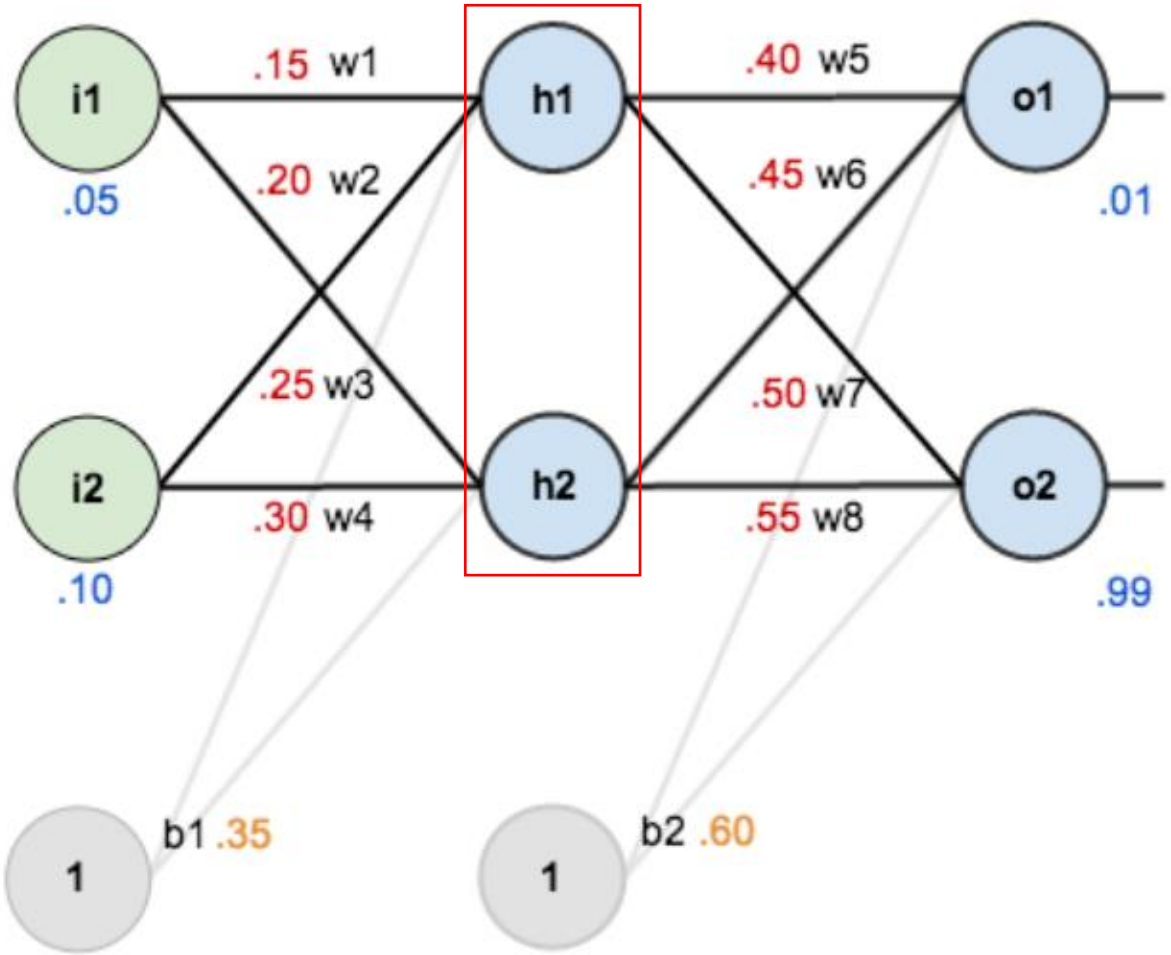
$$\frac{\partial E_{total}}{\partial a_{h1}} = \frac{\partial E_{o1}}{\partial a_{h1}} + \frac{\partial E_{o2}}{\partial a_{h1}}$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Calculate the partial derivative $\frac{\partial E_{total}}{\partial a_{h1}}$

$$\frac{\partial E_{total}}{\partial a_{h1}} = \frac{\partial E_{o1}}{\partial a_{h1}} + \frac{\partial E_{o2}}{\partial a_{h1}}$$

(i) $$\frac{\partial E_{o1}}{\partial a_{h1}} = \frac{\partial E_{o1}}{\partial z_{o1}} * \frac{\partial z_{o1}}{\partial a_{h1}}$$

$$\frac{\partial E_{o1}}{\partial z_{o1}} = \frac{\partial E_{o1}}{\partial a_{o1}} * \frac{\partial a_{o1}}{\partial z_{o1}} \text{ (calculated in w5)}$$
$$= 0.7414 * 0.1868 = 0.1385$$

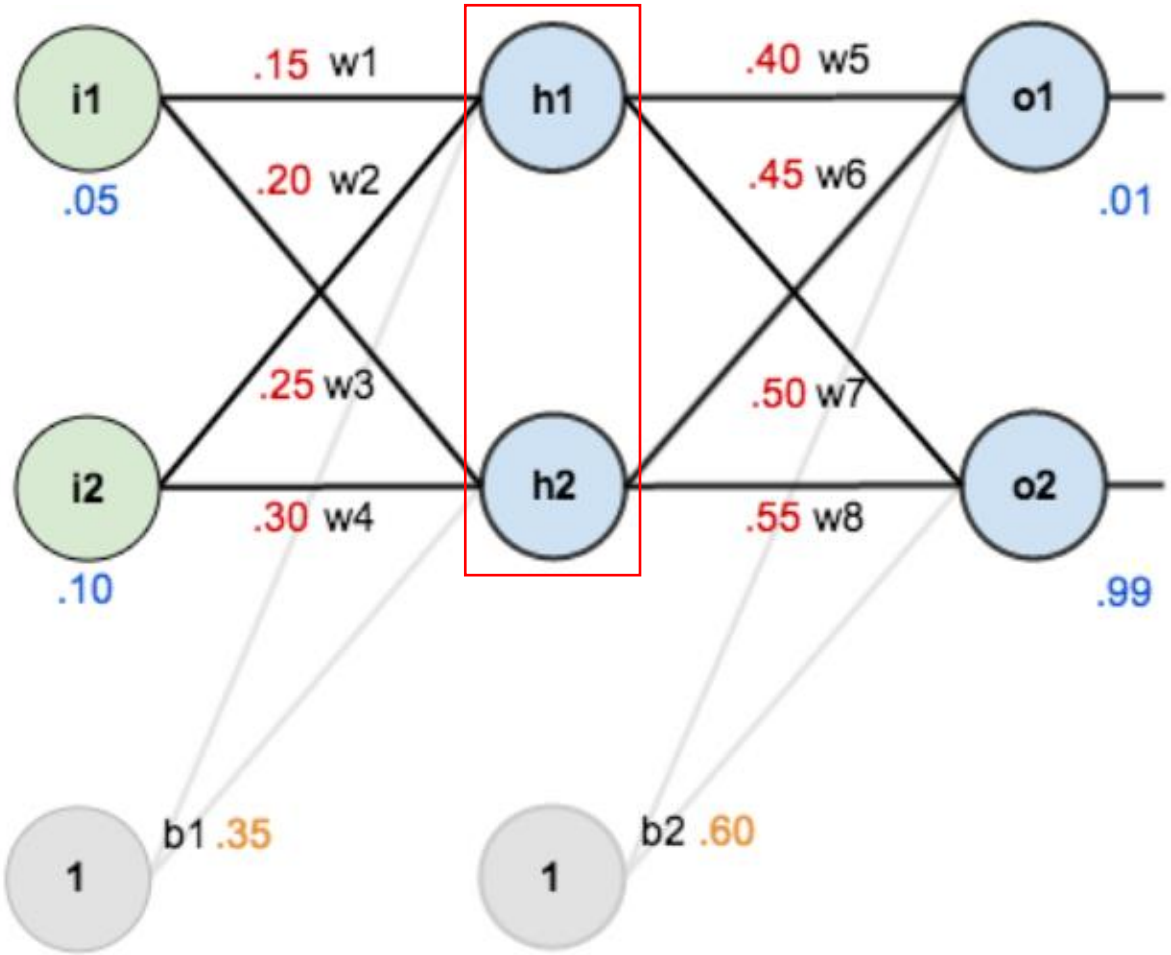
$$\frac{\partial z_{o1}}{\partial a_{h1}} = \frac{\partial (w5 * a_{h1} + w6 * a_{h2} + b2 * 1)}{\partial a_{h1}} = w5 = 0.40$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Calculate the partial derivative $\frac{\partial E_{total}}{\partial a_{h1}}$

$$\frac{\partial E_{total}}{\partial a_{h1}} = \frac{\partial E_{o1}}{\partial a_{h1}} + \frac{\partial E_{o2}}{\partial a_{h1}}$$

(i) $$\frac{\partial E_{o1}}{\partial a_{h1}} = \frac{\partial E_{o1}}{\partial z_{o1}} * \frac{\partial z_{o1}}{\partial a_{h1}}$$

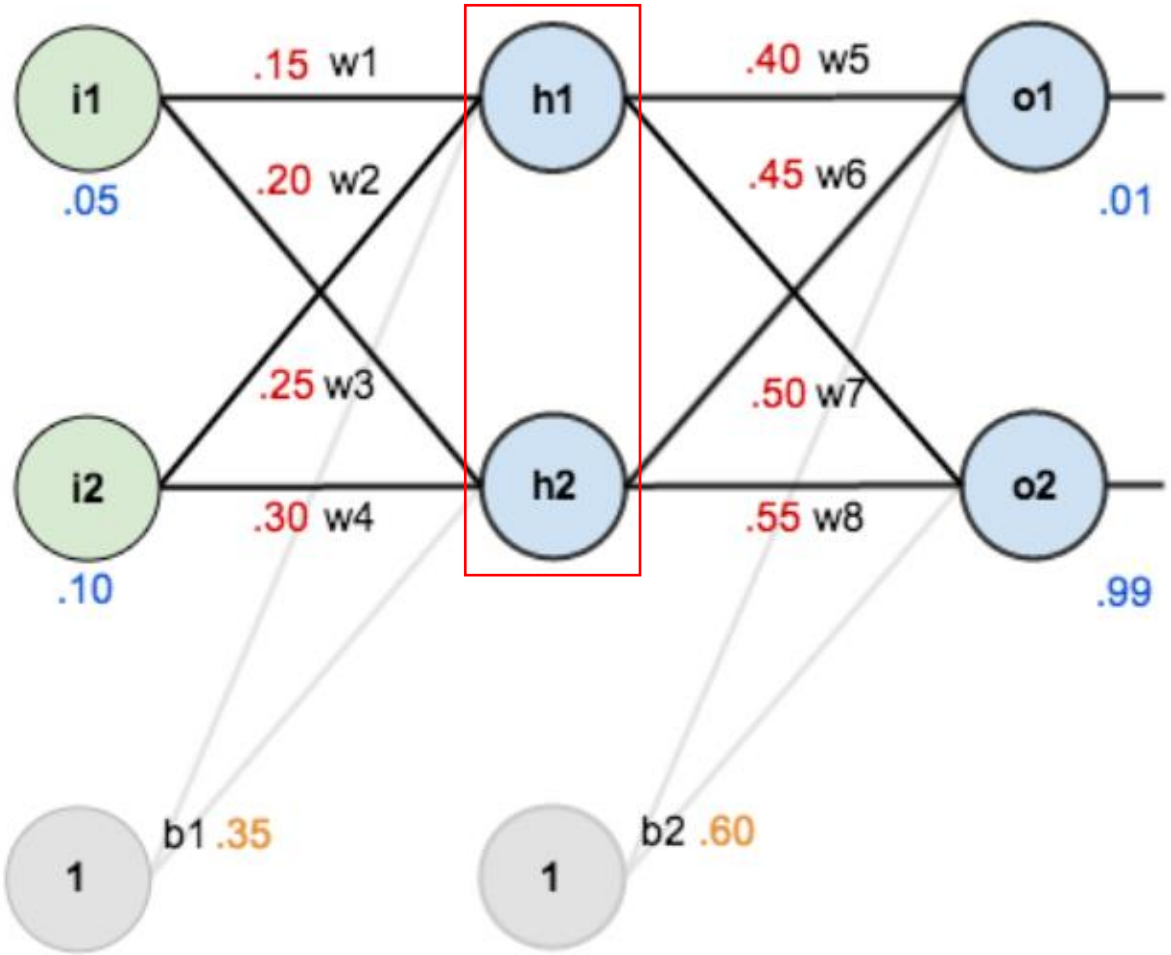
$$= 0.1385 * 0.40 = 0.0554$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Calculate the partial derivative $\frac{\partial E_{total}}{\partial a_{h1}}$

$$\frac{\partial E_{total}}{\partial a_{h1}} = \frac{\partial E_{o1}}{\partial a_{h1}} + \frac{\partial E_{o2}}{\partial a_{h1}}$$

(ii) $$\frac{\partial E_{o2}}{\partial a_{h1}} = \frac{\partial E_{o2}}{\partial z_{o2}} * \frac{\partial z_{o2}}{\partial a_{h1}}$$

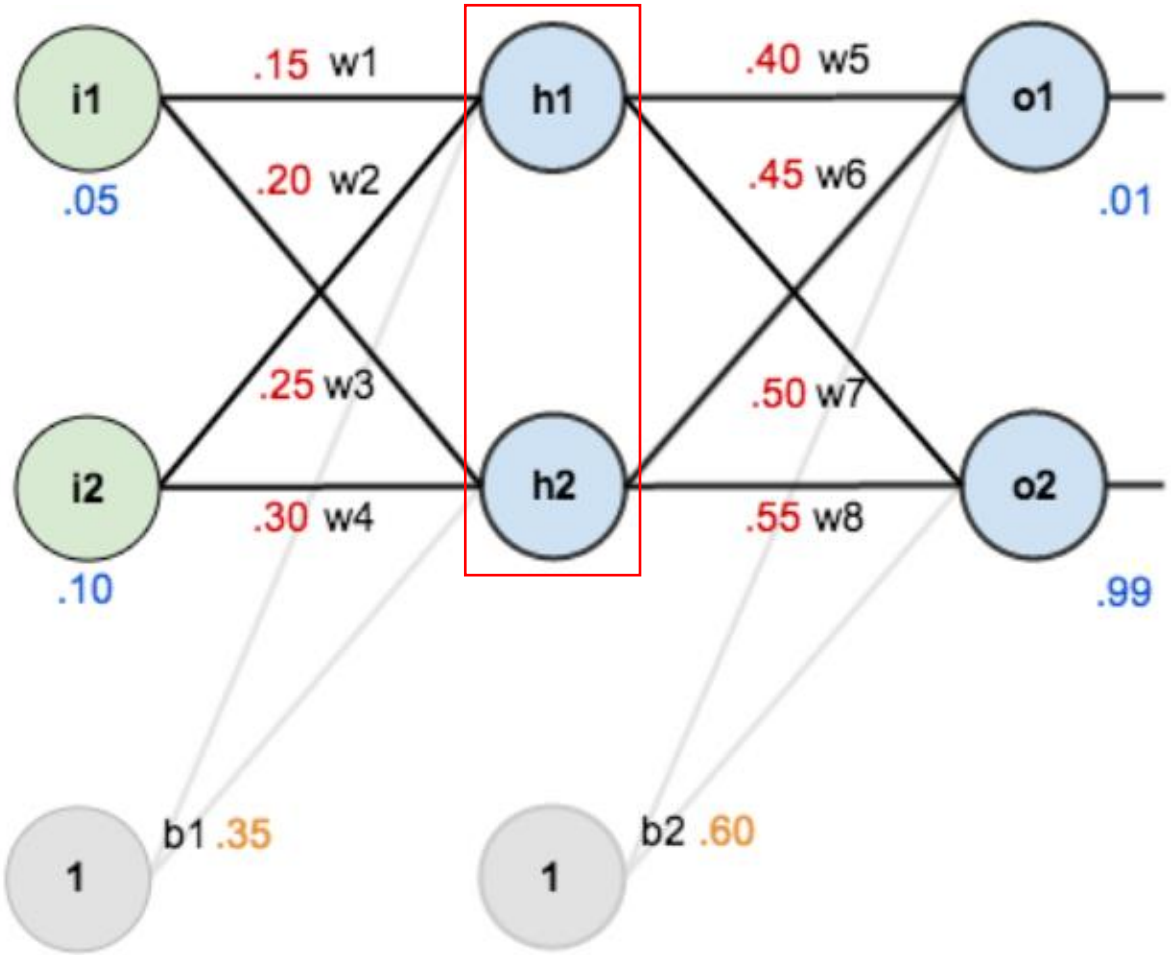
$$= -0.0190$$ (follow same process of (i))

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial z_{h1}} * \frac{\partial z_{h1}}{\partial w_1}$$

  - Calculate the partial derivative $\frac{\partial E_{total}}{\partial a_{h1}}$

$$\frac{\partial E_{total}}{\partial a_{h1}} = \frac{\partial E_{o1}}{\partial a_{h1}} + \frac{\partial E_{o2}}{\partial a_{h1}}$$

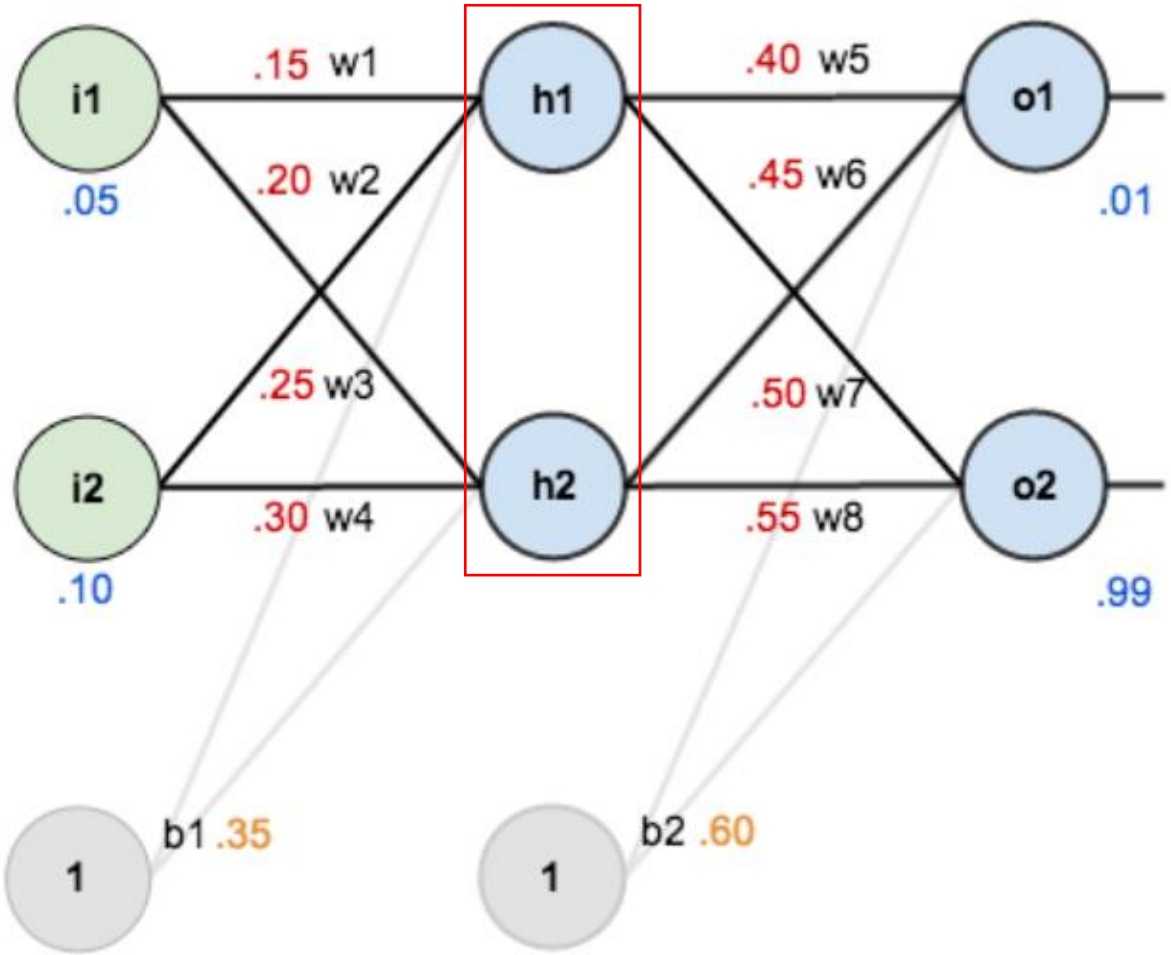
$$= 0.0554 - 0.0190 = 0.0364$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial z_{h1}} * \frac{\partial z_{h1}}{\partial w_1}$$

  - Calculate the partial derivative $\frac{\partial a_{h1}}{\partial z_{h1}}$

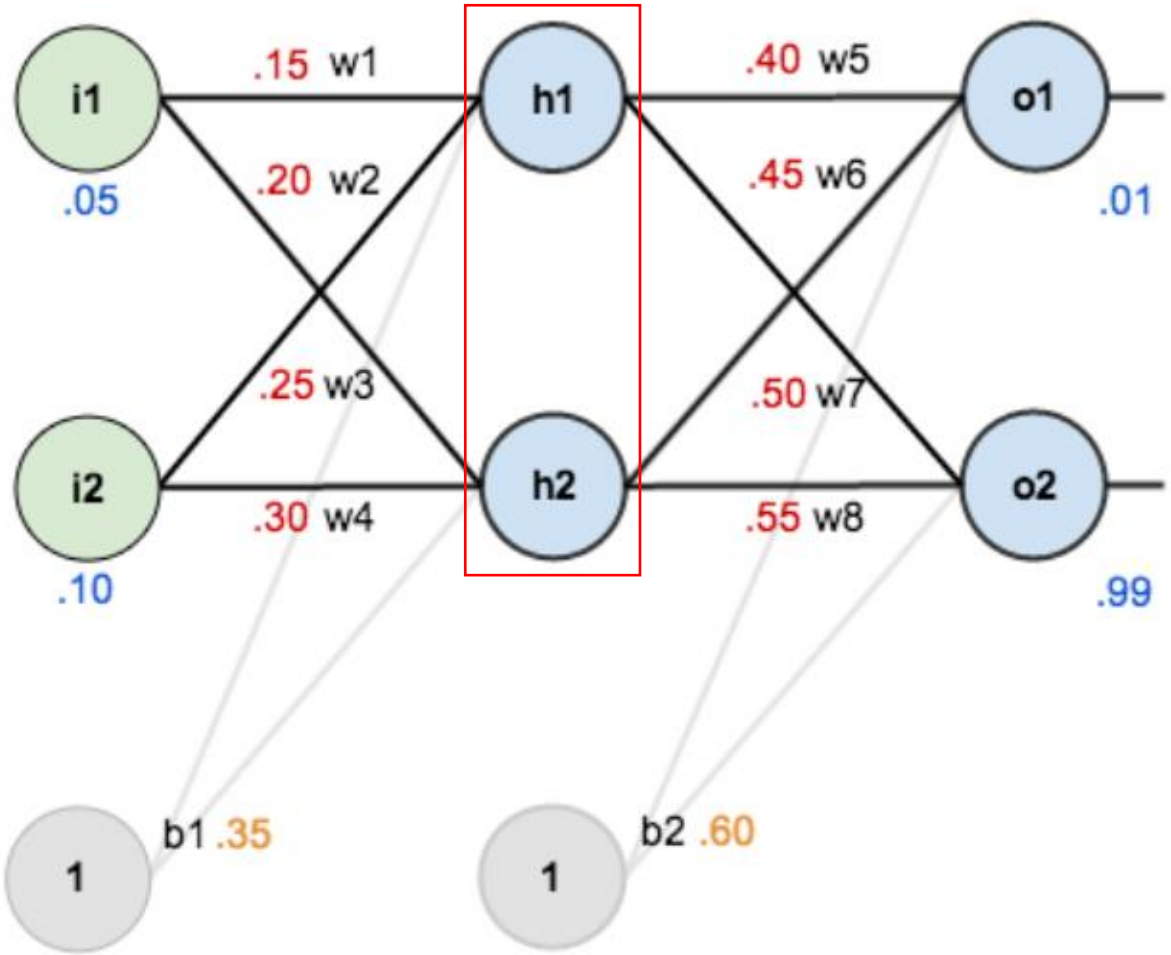
$$\frac{\partial a_{h1}}{\partial z_{h1}} = a_{h1} * (1 - a_{h1})$$
$$= 0.5933 * (1 - 0.5933) = 0.2413$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial z_{h1}} * \frac{\partial z_{h1}}{\partial w_1}$$

  - Calculate the partial derivative $\frac{\partial z_{h1}}{\partial w_1}$

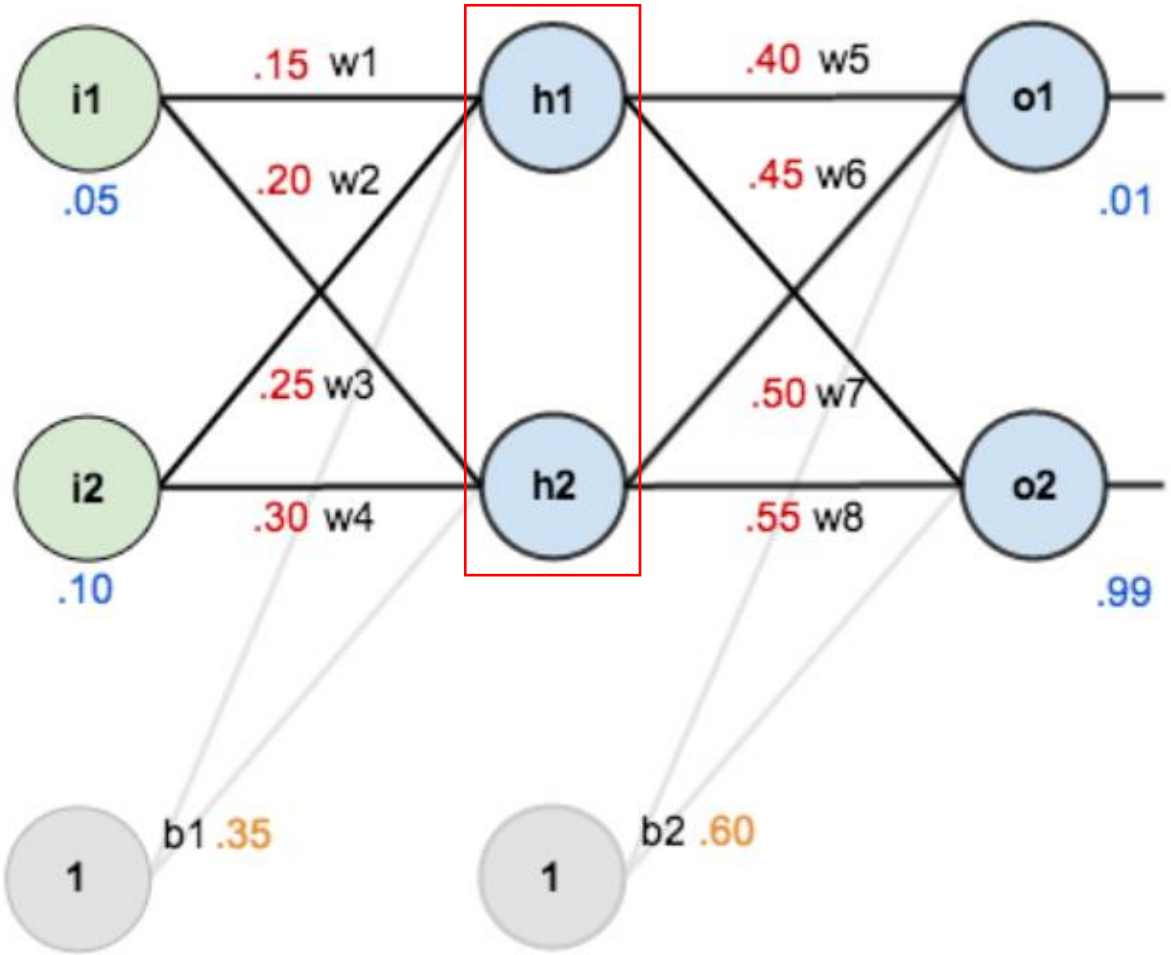
$$\frac{\partial z_{h1}}{\partial w_1} = \frac{\partial (w1 * a_{i1} + w2 * a_{i2} + b1 * 1)}{\partial w1} = a_{i1} = 0.05$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for hidden layer
  - Chain rule for partial derivative

$$\frac{\partial E_{total}}{\partial w1} = \frac{\partial E_{total}}{\partial a_{h1}} * \frac{\partial a_{h1}}{\partial z_{h1}} * \frac{\partial z_{h1}}{\partial w_1}$$
$$= 0.0364 * 0.2413 * 0.05 = 0.0004$$

  - Gradient descent update for w1

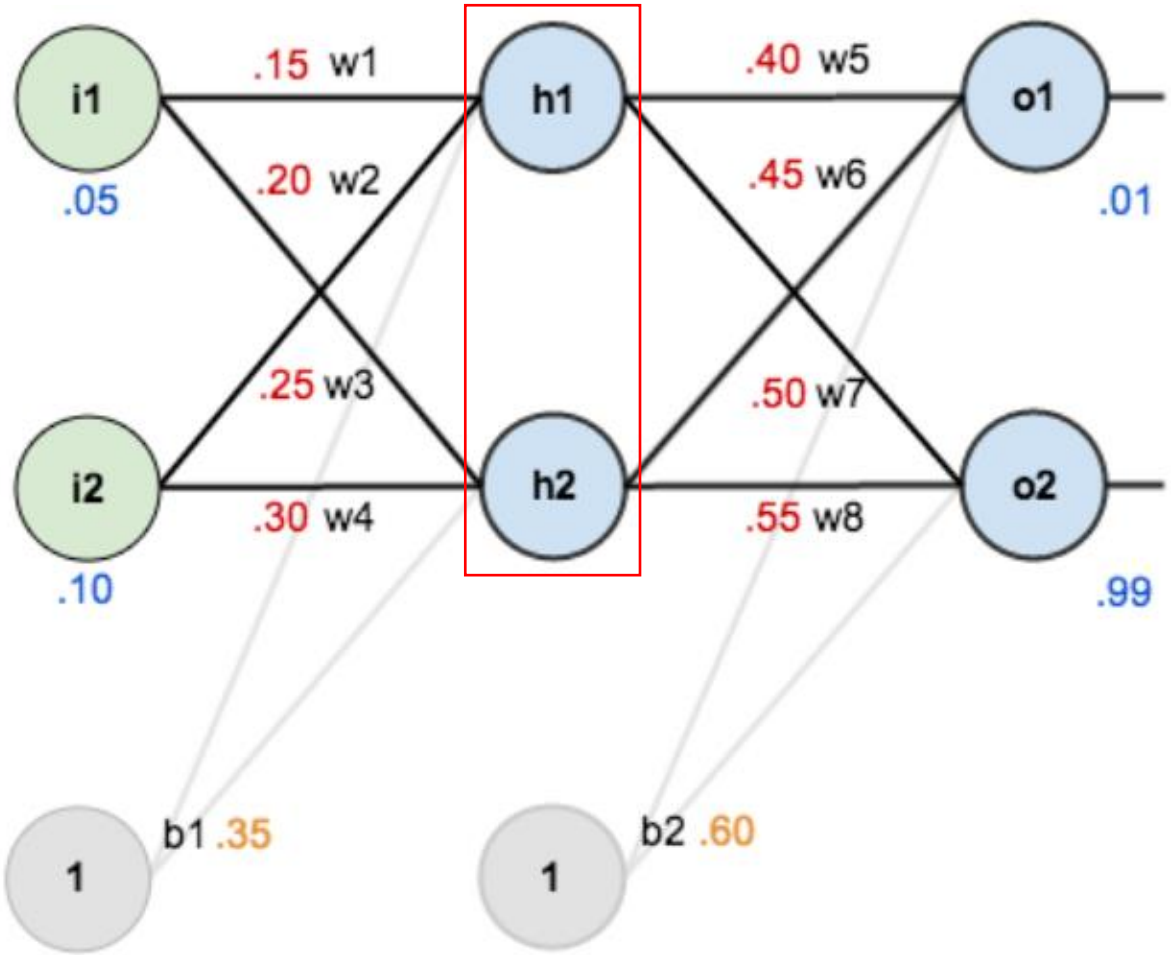
$$w1^{+} \leftarrow w1 - \alpha * \frac{\partial E_{total}}{\partial w1}$$
$$w1^{+} \leftarrow 0.15 - 0.5 * 0.0004 = 0.1498$$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 5:

- Backward propagation for output layer
  - Following the same process

$$w1^{+} \leftarrow w1 - \alpha * \frac{\partial E_{total}}{\partial w1} = 0.1498$$

$$w2^{+} \leftarrow w2 - \alpha * \frac{\partial E_{total}}{\partial w2} = 0.1996$$

$$w3^{+} \leftarrow w3 - \alpha * \frac{\partial E_{total}}{\partial w3} = 0.2498$$

$$w4^{+} \leftarrow w4 - \alpha * \frac{\partial E_{total}}{\partial w4} = 0.2995$$

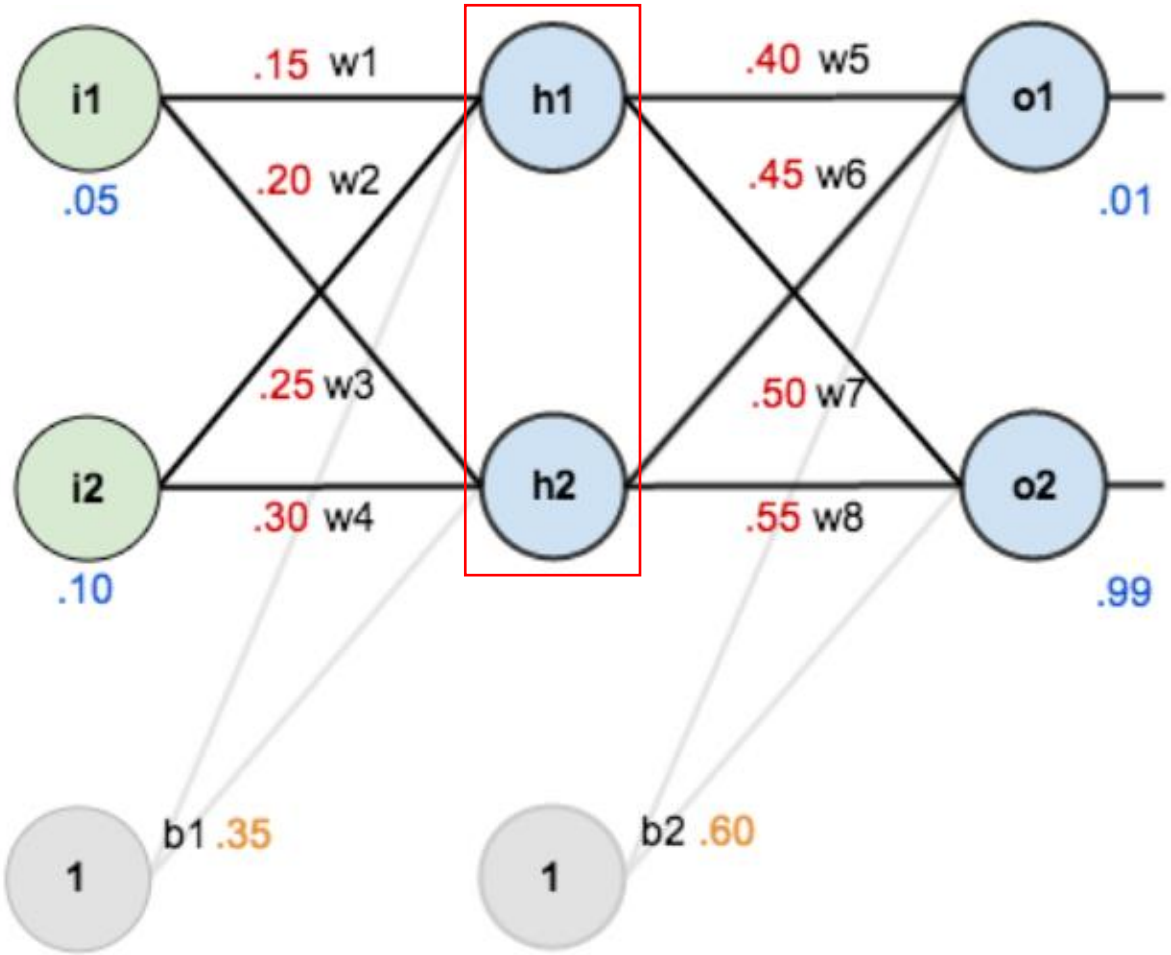
  - Note: the updating of w and b happens actually after the back-propagation step.

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | Backpropagation

**Example 8:** Manually calculate backpropagation on the neural network below. Knowing that:

- Activation function is sigmoid function
- Learning rate $\alpha = 0.5$




## Step 6:

- Actual update all the w and b
- Repeat forward propagation and backward propagation until convergence.

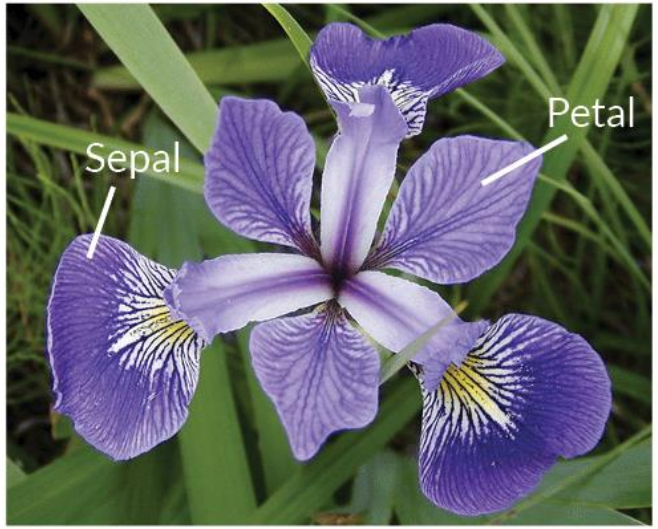
## Other notes:

- Batch gradient descent: use all data to calculate gradient descent.
- Stochastic gradient descent (SGD): use a single sample to calculate gradient descent.
- Mini-batch.

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: A Step by Step Backpropagation Example. Link: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

# II. Neural Networks | The architecture

- **Example 9:** Simple neural network to classify Iris flower types.




**Iris Versicolor**

**Iris Setosa**

**Iris Virginica**

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | The architecture

- **Example 9:** Simple neural network to classify Iris flower types.

  - **Q1:** Build a simple neural networks models with 1 hidden layers, 10 sigmoid neurons. What is the performance of the model?

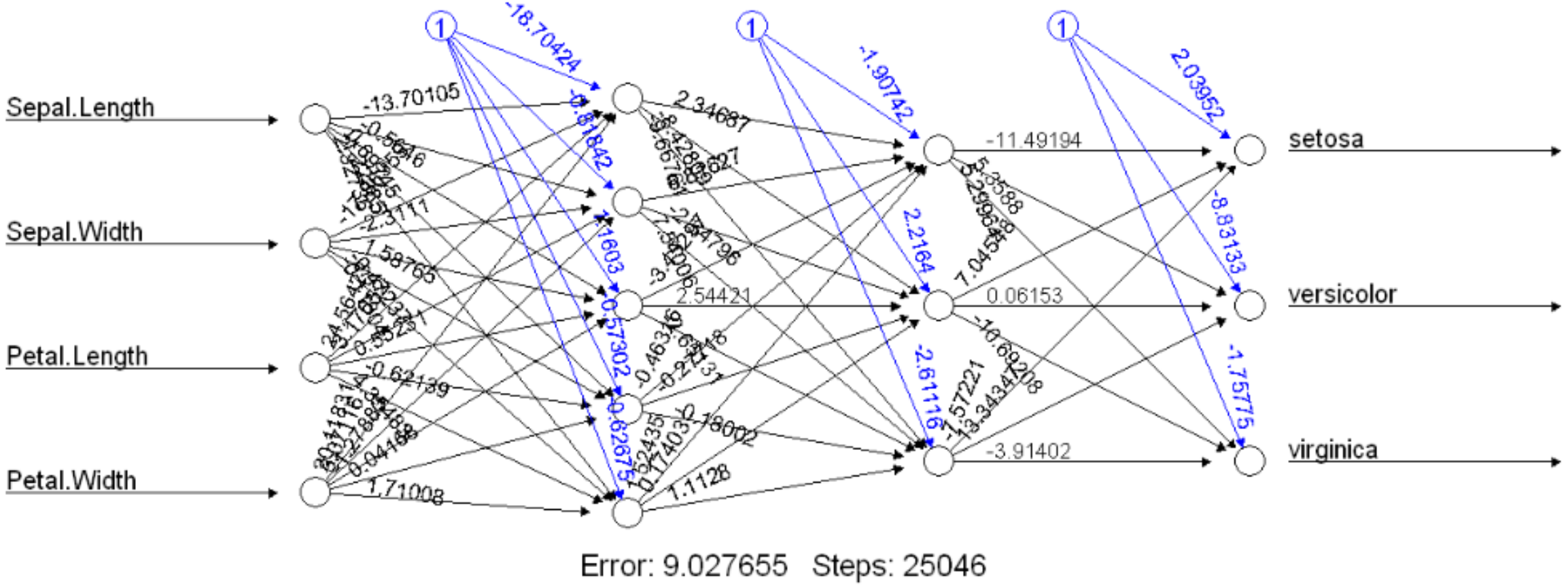
  - **Q2:** Build a multilayer neural networks with 2 hidden layers, (5, 3) sigmoid neurons. What is the performance of the model?

  - **Q3:** Build your own neural networks model. Compare the performance with 2 previous models.

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | The architecture

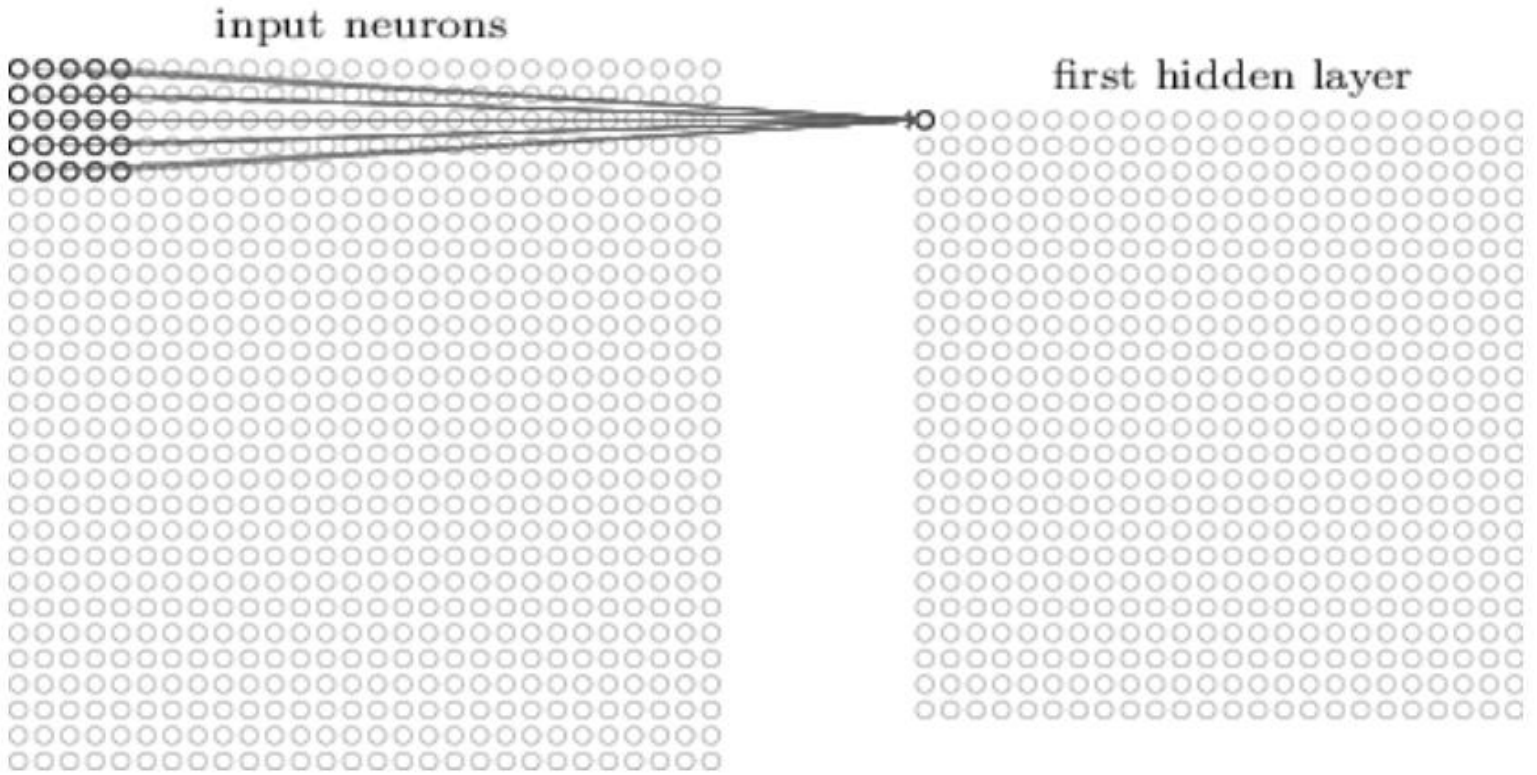
- **Example 9:** Simple neural network to classify Iris flower types.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Introducing convolutional networks

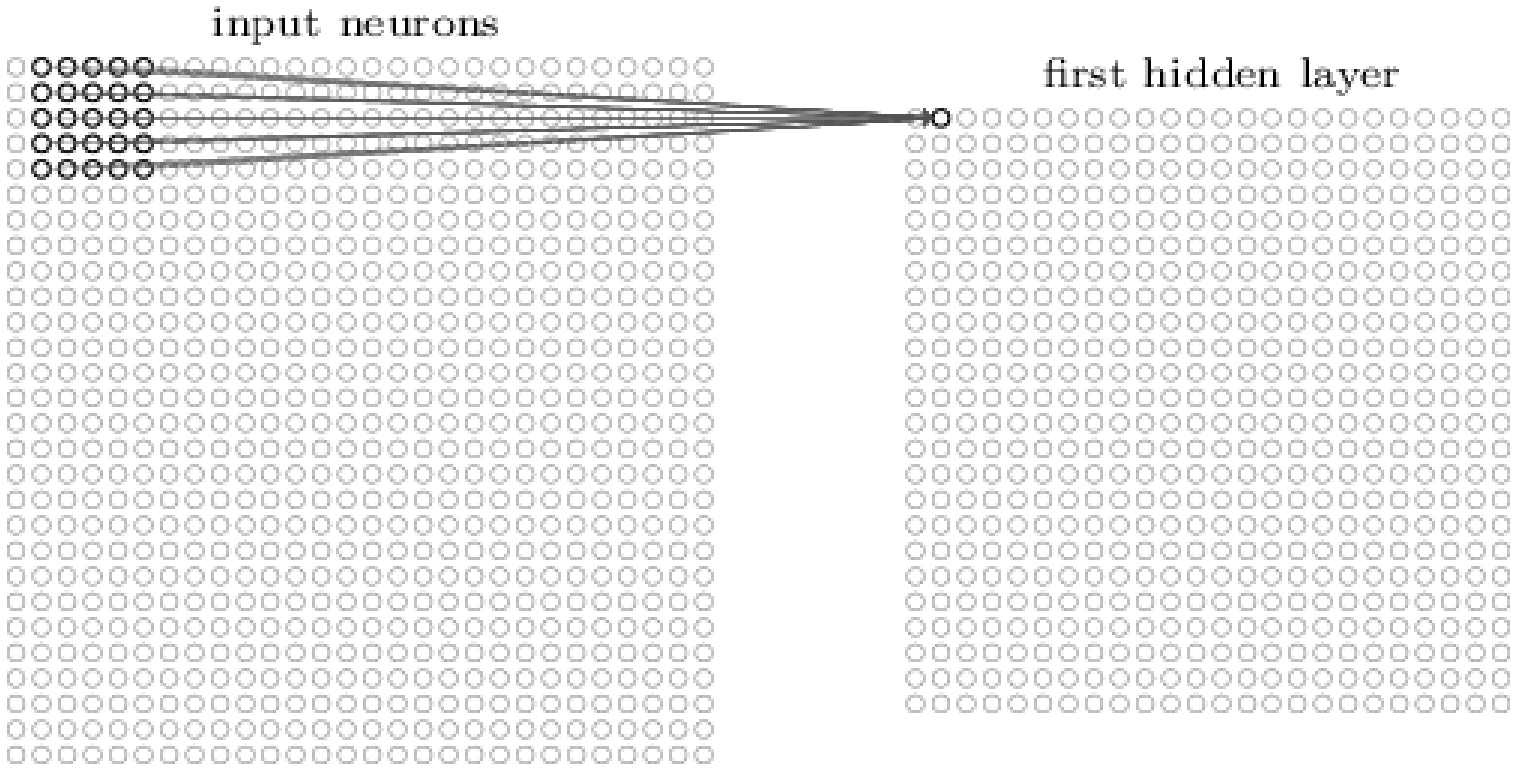
- **Local receptive fields:** Each neuron in the first hidden layer will be connected to a small region of the input neurons.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Introducing convolutional networks

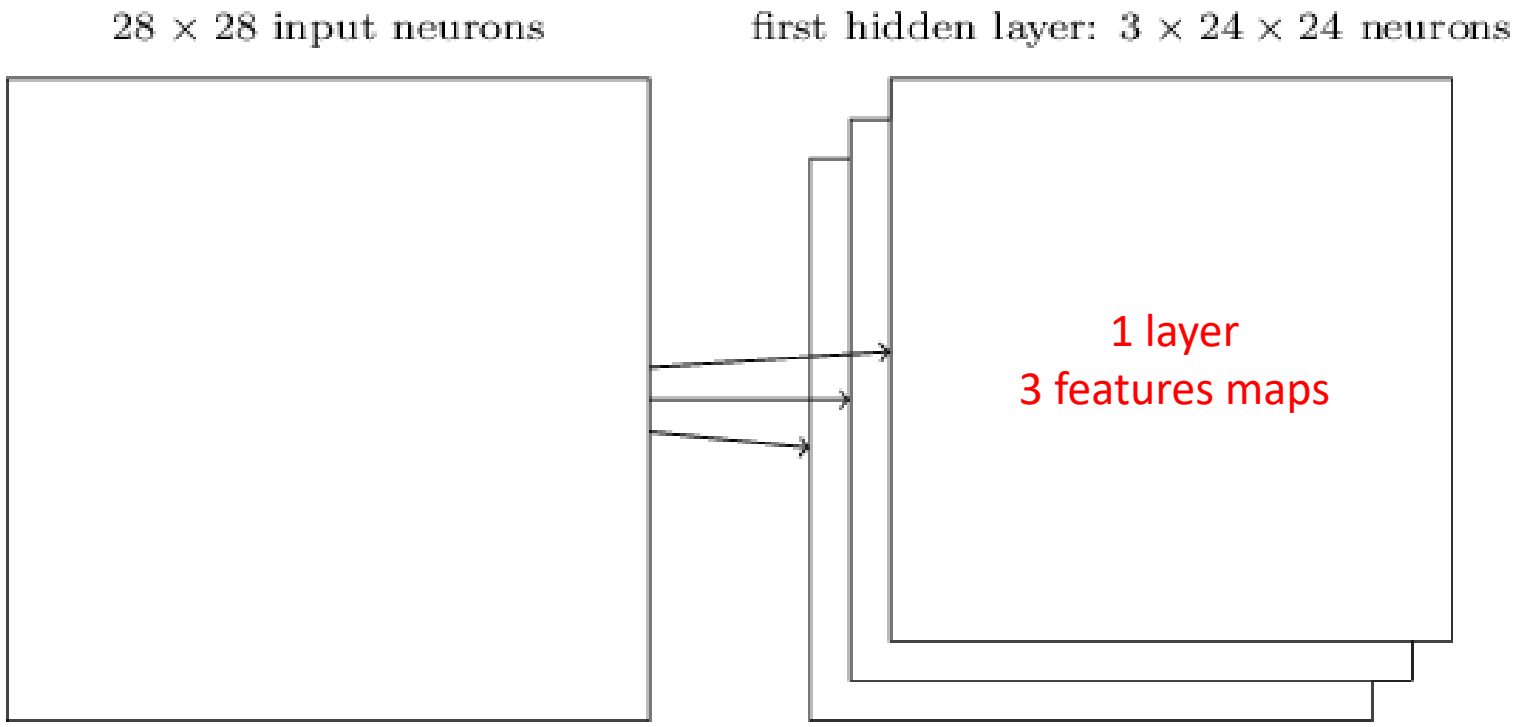
- **Local receptive fields:** Each neuron in the first hidden layer will be connected to a small region of the input neurons.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Introducing convolutional networks

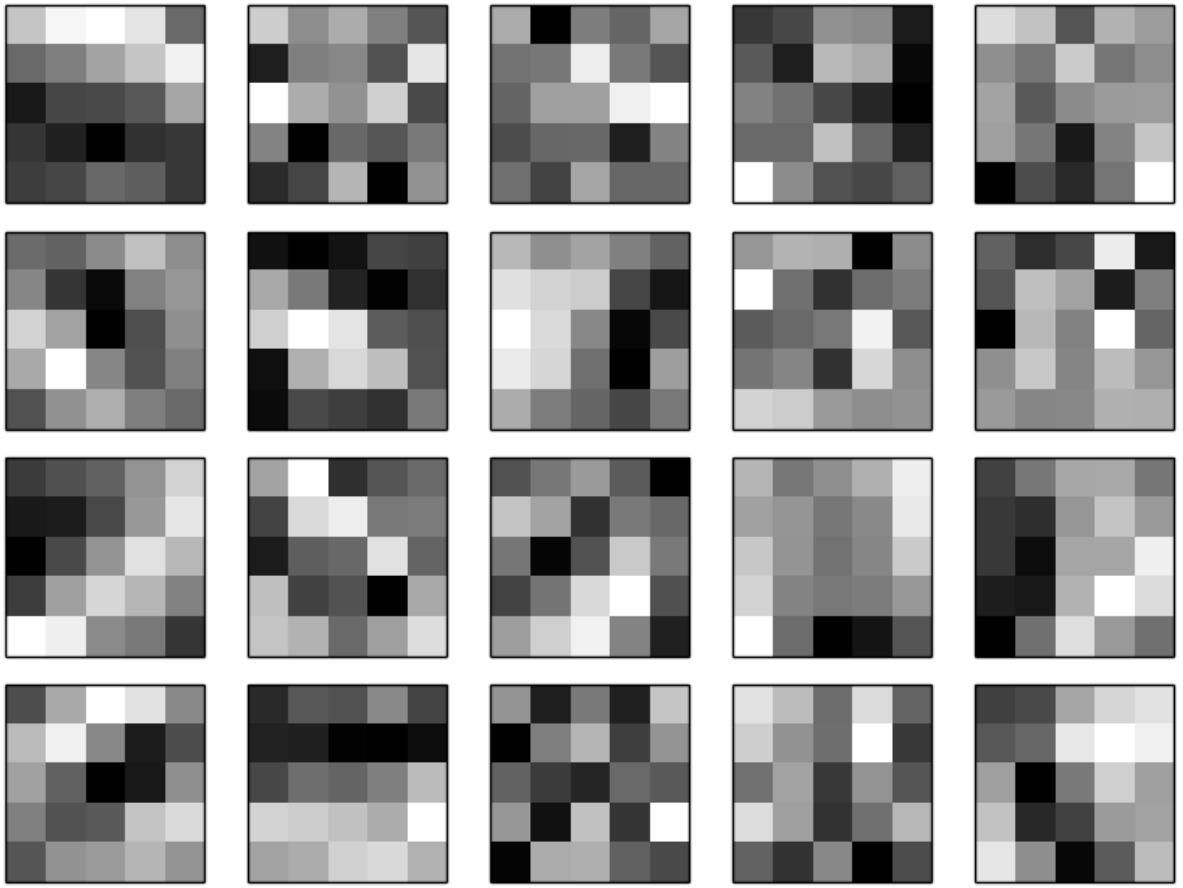
- **Shared weights and biases:** All 24 x 24 neurons in the first hidden layer use the same weights and bias. Then, all the neurons in the first hidden layer detect exactly the same feature, just at different locations in the input image.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Introducing convolutional networks

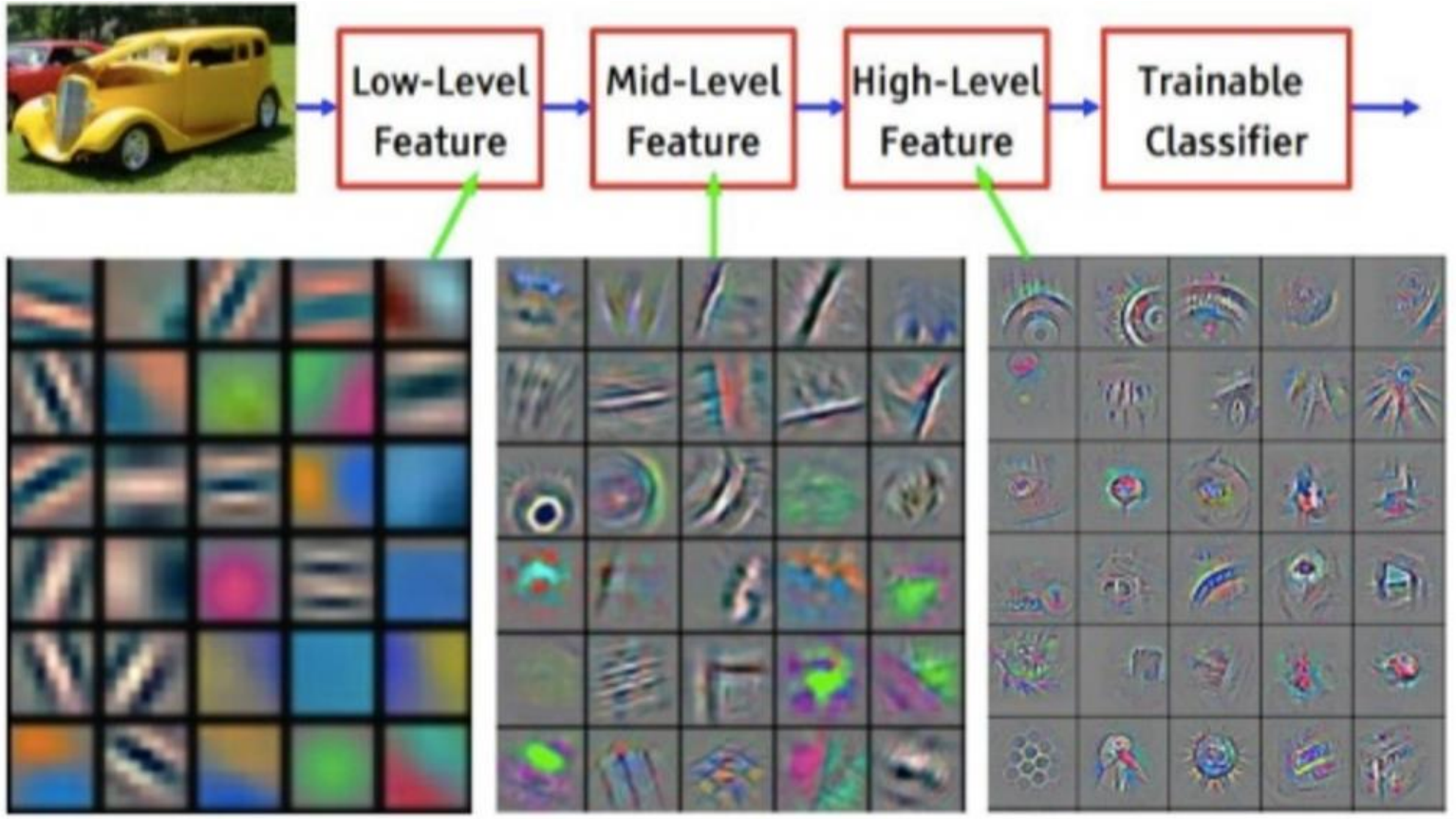
- **Example 10:** Convolutional layers with 20  different feature maps (or filters, or kernels).



Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Introducing convolutional networks

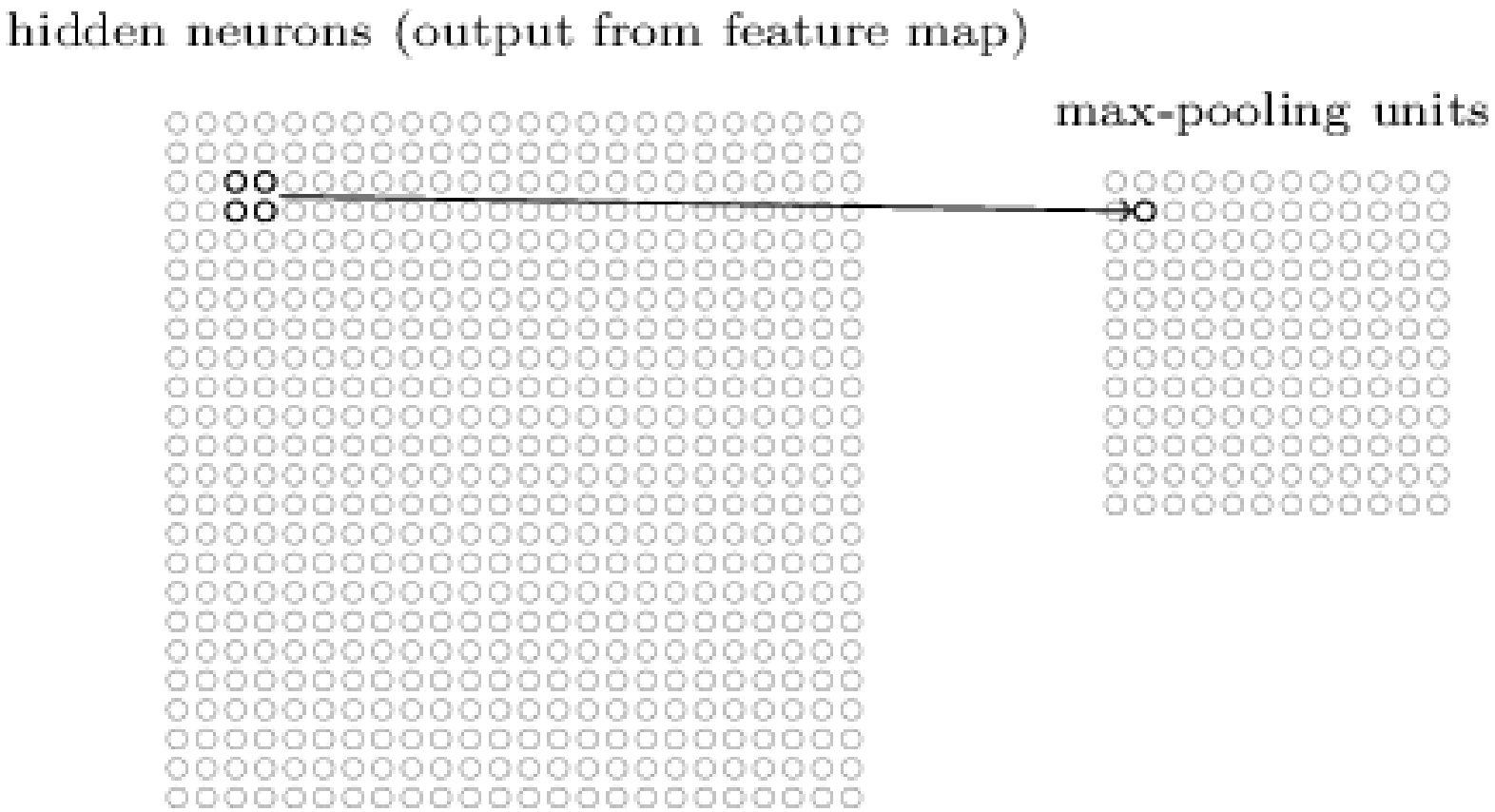
- **Example 10:** Convolutional 3 layers and 30 features maps.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html
Source: Identifying Traffic Signs with Deep Learning. Link: https://towardsdatascience.com/identifying-traffic-signs-with-deep-learning-5151eece09cb

# II. Neural Networks | Introducing convolutional networks

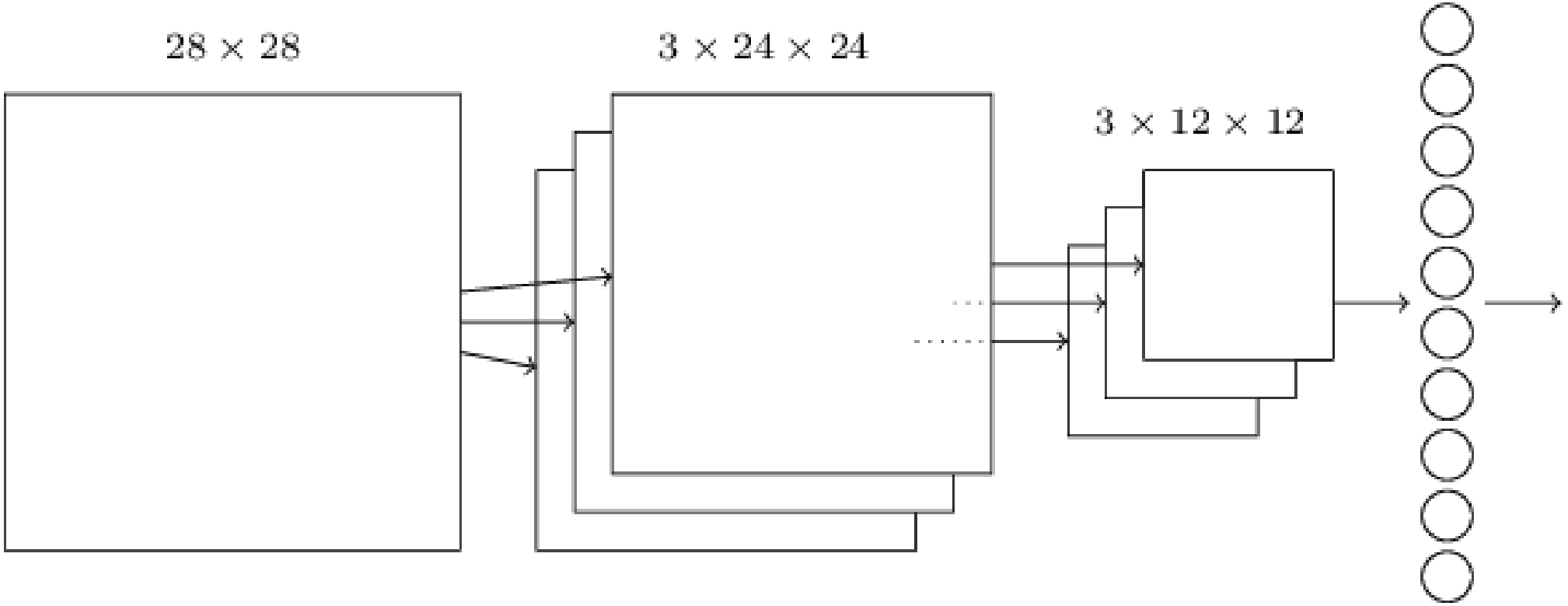
- **Pooling layers:** Simplify the information in the output from the convolutional layer.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Introducing convolutional networks

- Putting it all together




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# II. Neural Networks | Others

- Deep learning = Deep Neural Networks + other training algorithms.
- Recurrent neural networks (RNNs)
- Long short-term memory units (LSTMs)
- Deep belief nets, generative models, and Boltzmann machines

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# Outline

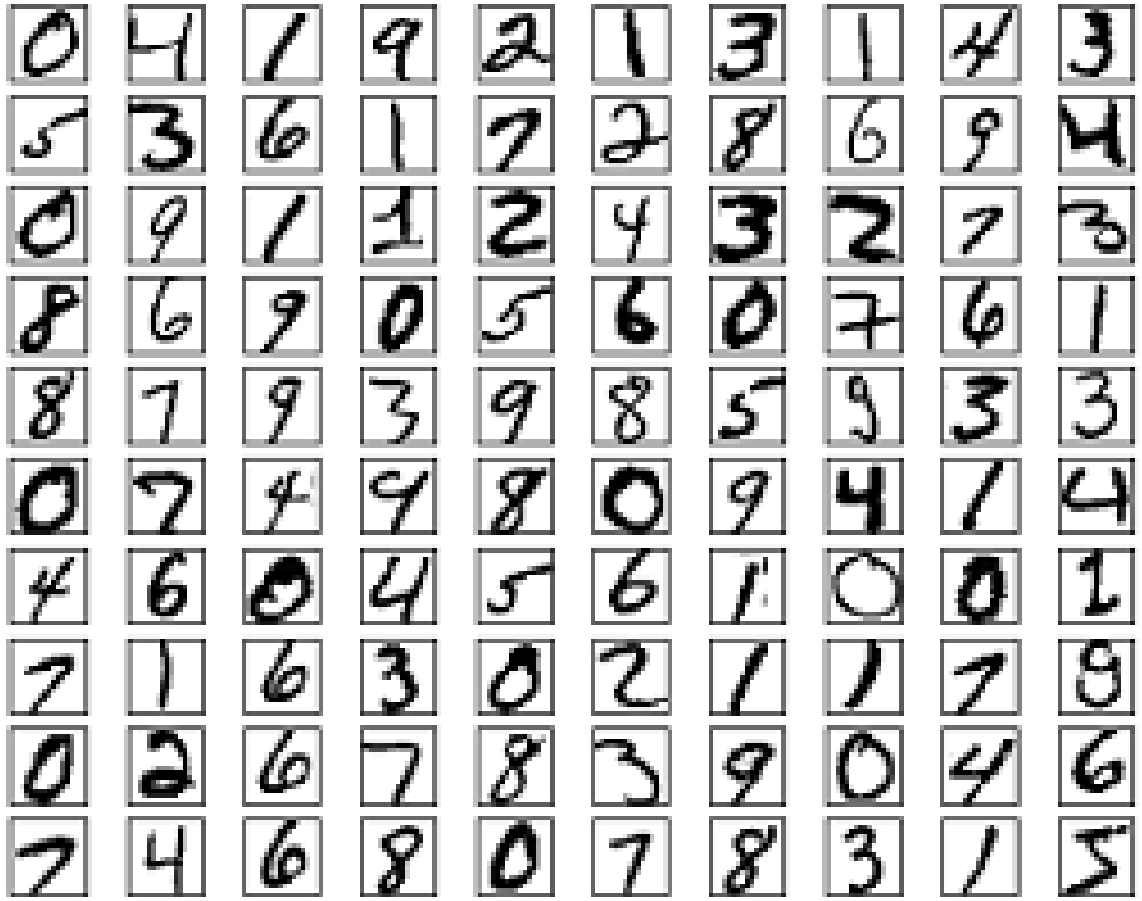
- I. Review: Unsupervised Learning
- II. Neural Networks
- Lab: Neural Networks

# Lab: Neural Networks

1. The MNIST Database of Handwritten Digits (package: neuralnet)
2. Step by step: Construct Neural Network function
3. Build Neural Network with Tensorflow 2.x and Keras

# Lab: Neural Networks

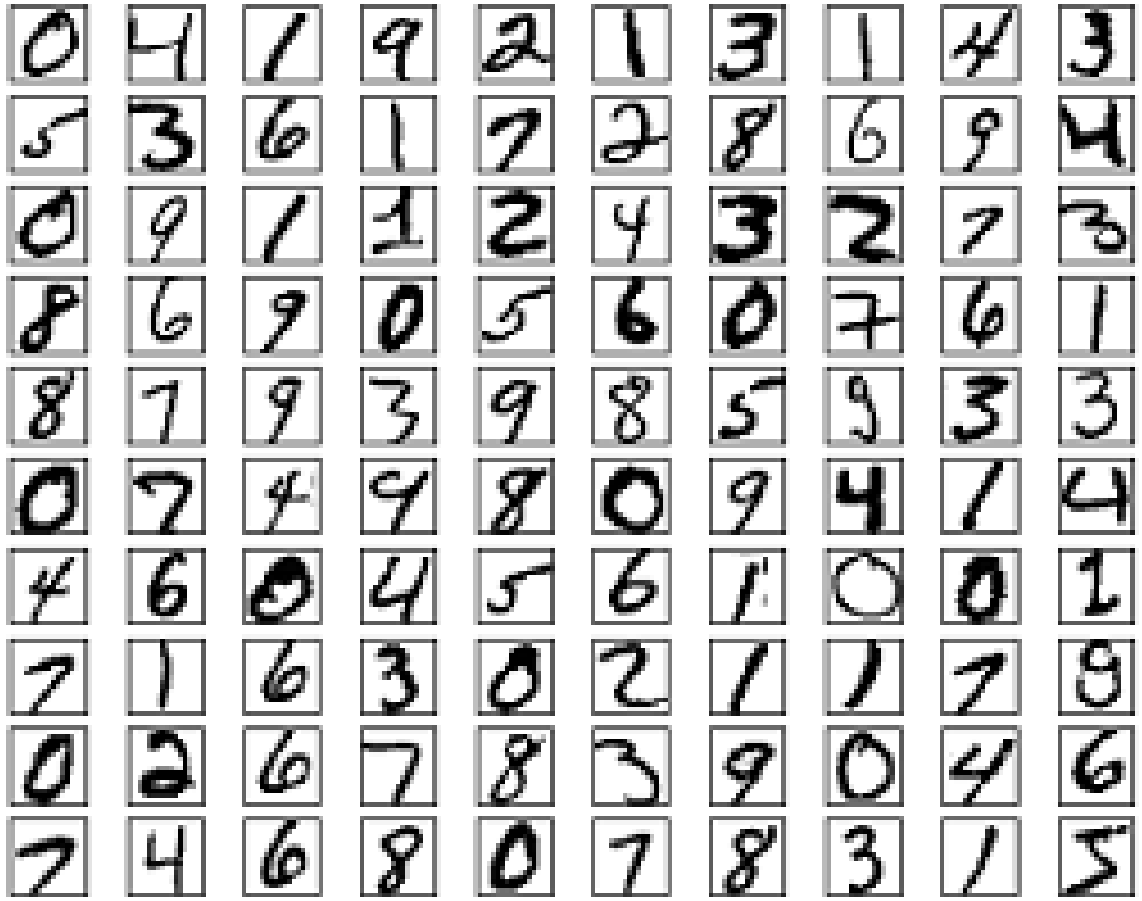
1. The MNIST Database of Handwritten Digits (package: neuralnet)
   - Simple neural network to classify handwritten digits (MNIST data).
   - Input image: 28 x 28 pixels
   - Labels: 0..9



- Design the solution network:
  - 3-layer neural network (1 hidden layer)
  - Input layer: 28 x 28 = 784 input neurons
  - Hidden layer: n = 15 neurons
  - Output layer: 10 output neurons

# Lab: Neural Networks

2. Step by step: Construct Neural Network function
   - Source: https://www.kaggle.com/russwill/build-your-own-neural-network-in-r



# Lab: Neural Networks

3. Build Neural Network with Tensorflow 2.x and Keras
   - Google Colab link: https://colab.research.google.com/drive/1lpntScgiCwasiv8FBP-P7DWJ-tKSqmwI

# Summary | Section 7 - Neural Networks

- I. Review: Unsupervised Learning
  - Unsupervised vs. Supervised Learning
  - PCA
  - Clustering
    - K-Means
    - Hierarchical clustering

- II. Neural Networks

- Lab: Neural Networks

- Exercise: Neural Network

# Homework | Section 7 - Neural Networks

- Exercise:
  - Neural Network Exercise in the next slide.

- Submission:
  - File: Jupyter Notebook (*.ipynb).
  - Upload to your personal GitHub + send the link to: m.phan@ieseg.fr
  - Deadline: before next session.

# Homework: Neural Network

- Topic: Company Bankruptcy Prediction
- Source: https://archive.ics.uci.edu/ml/datasets/Taiwanese+Bankruptcy+Prediction
- Data: ./data/bankruptcy_prediction/data.csv
- Library:
  - R: neuralnet
  - Python: Tensorflow

- Task:
  - Q1: Randomly divide data into train/test as 80/20 (set.seed=1).
  - Q2: Build a NN model with 1 hidden layer of 30 neurons, sigmoid activation function.
  - Q3: Build a deep NN model with multiple hidden layers (of your choice) and sigmoid activation function.
  - Q4: Build 5 other classification models and compare with the 2 previous NN models.

# Appendix | References


- Neural Networks and Deep Learning. Link: http://neuralnetworksanddeeplearning.com/index.html
- Neural Networks Learning | Backpropagation Algorithm | Andrew Ng. Link: https://www.youtube.com/watch?v=x_Eamf8MHwU
- Neural Networks Learning | Backpropagation Intuition | Andrew Ng. Link: https://www.youtube.com/watch?v=mOmkv5SI9hU
- For Dummies — The Introduction to Neural Networks we all need ! (Part 1). Link: https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-c50f6012d5eb
- What is the difference between Neural Networks and Deep Learning? Link: https://www.quora.com/What-is-the-difference-between-Neural-Networks-and-Deep-Learning#
- What is the difference between a neural network and a deep neural network, and why do the deep ones work better? Link: https://stats.stackexchange.com/questions/182734/what-is-the-difference-between-a-neural-network-and-a-deep-neural-network-and-w
- Artificial neural network. Link: https://en.wikipedia.org/wiki/Artificial_neural_network
- The MNIST Database of Handwritten Digits. Link: http://yann.lecun.com/exdb/mnist/
- A Basic Introduction To Neural Networks. Link: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html
- Build your own neural network in R. Link: https://www.kaggle.com/russwill/build-your-own-neural-network-in-r
- Mini Batch Gradient Descent (C2W2L01). Link: https://www.youtube.com/watch?v=4qJaSmvhxi8

# II. Neural Networks | Learning with gradient descent

- The quadratic cost function:

$$C(w,b) \equiv \frac{1}{2n}\sum_{x}\|y(x)-a\|^2$$

  - $x$ : Training input
  - $w, b$ : Weights and biases
  - $n$ : Total number of training inputs
  - $a$ : Output vector (estimated values)
  - $y(x)$ : Desired output (true values)

- The training algorithm will need minimize the cost function.

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# Appendix | Learning with gradient descent

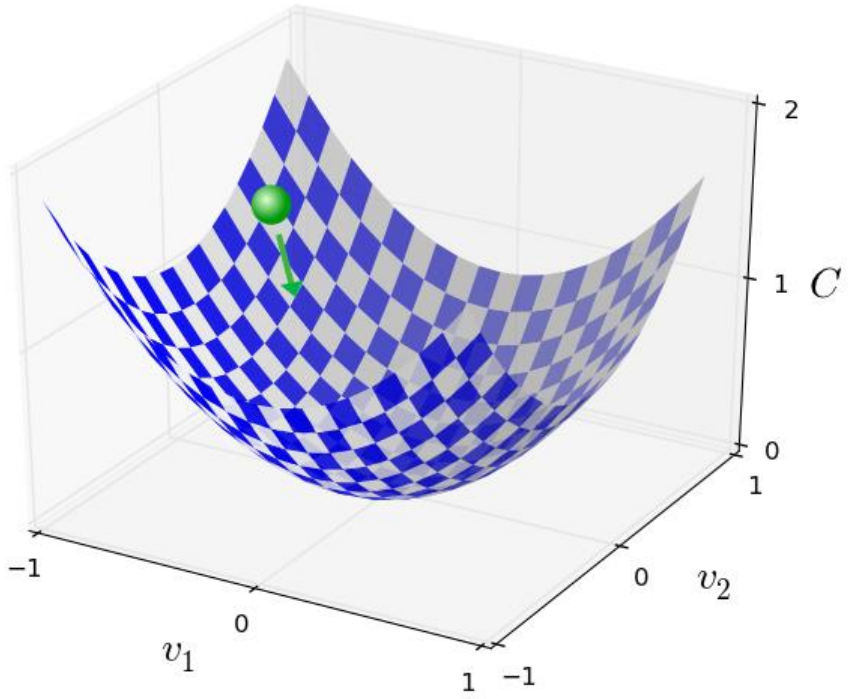
- Gradient descent with two variables $v = (v_1, v_2)$, minimize function $C(v)$.

- Simulate the motion of the ball rolling down the valley.

- The amount of $C$ changes when $v_1$ and $v_2$ change:

$$\Delta C \approx \frac{\partial C}{\partial v_1}\Delta v_1 + \frac{\partial C}{\partial v_2}\Delta v_2$$

- **Target:** choosing $\Delta v_1$ and $\Delta v_2$ to make $\Delta C$ negative.



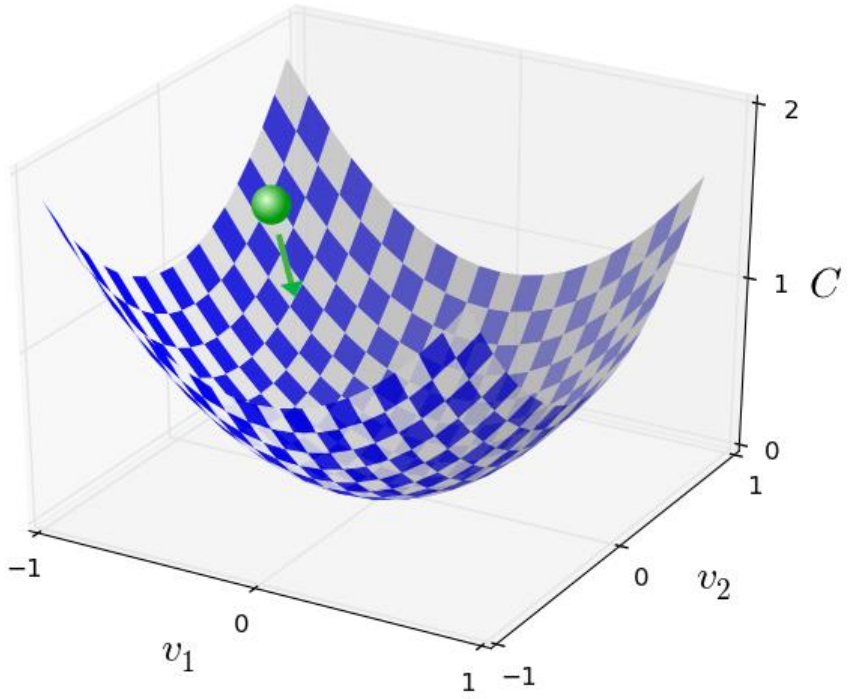

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# Appendix | Learning with gradient descent

- Rewriting the function of $\Delta C$:

$$\Delta C \approx \nabla C . \Delta\, v$$

- In which:
  - The gradient vector: $\nabla C \equiv \left(\frac{\partial C}{\partial v_1}, \frac{\partial C}{\partial v_2}\right)^T$
  - The vector of changes: $\Delta v \equiv (\Delta v_1, \Delta v_2)^T$
- **Target:** choosing $\Delta v_1$ and $\Delta v_2$ to make $\Delta C$ negative.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# Appendix | Learning with gradient descent

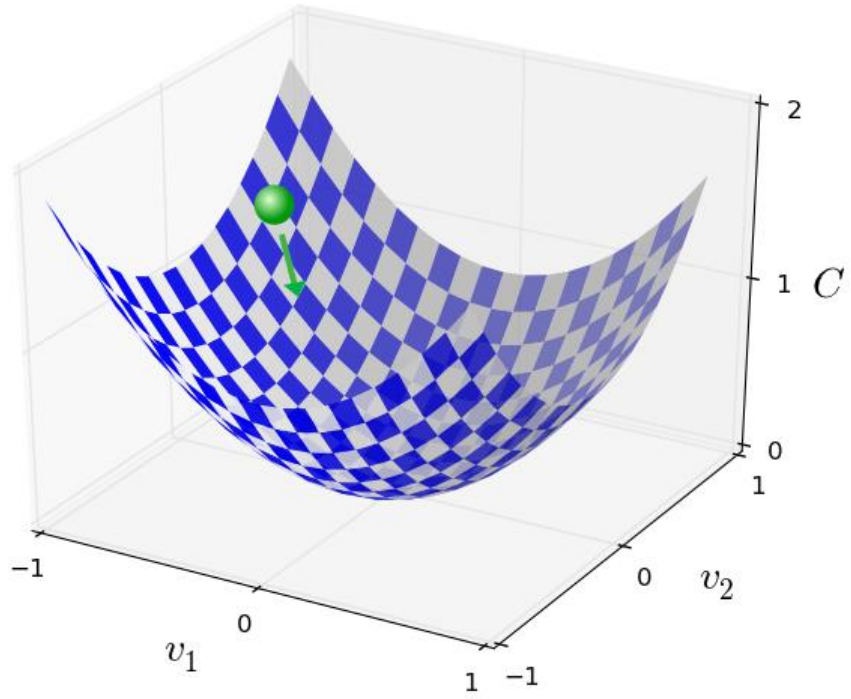
- Choosing vector of changes $\Delta v$: $\Delta v = -\eta \nabla C$

- The change of function $C$ is always negative:

$$\Delta C \approx \nabla C . \Delta v$$

$$\Delta C \approx -\eta \nabla C . \nabla C = -\eta \|\nabla C\|^2$$

- In which:
  - $\eta > 0$ (eta): Learning rate




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

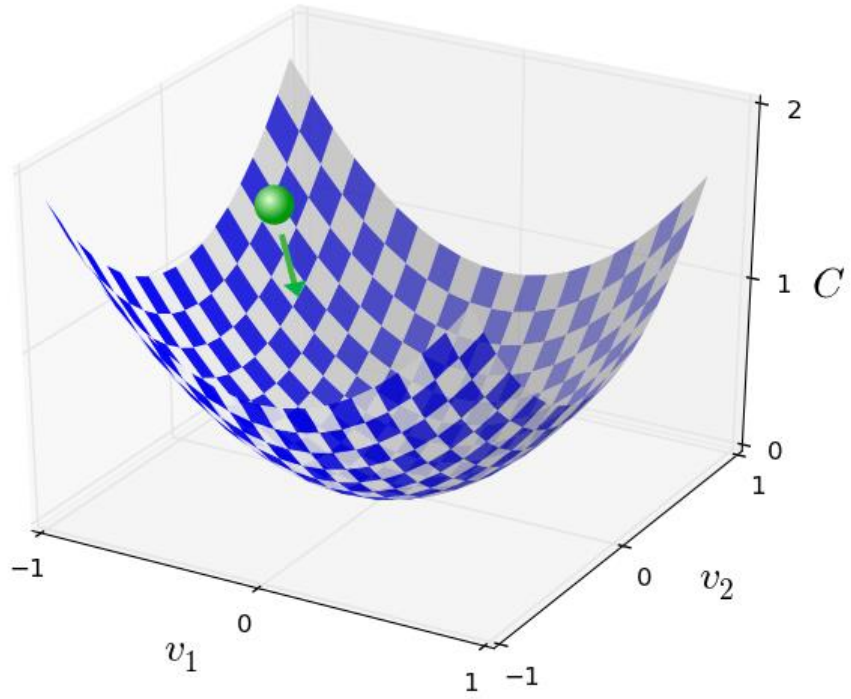
# Appendix | Learning with gradient descent

- Updating the new position of the ball:

$$v \rightarrow v' = v - \eta \nabla C$$

- Repeat until reaching the global minimum.

- Gradient descent works with multiple variables as well.




Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# Appendix | Learning with gradient descent

- The quadratic cost function:

$$C(w,b) \equiv \frac{1}{2n}\sum_{x}\|y(x)-a\|^2$$

- Applying the gradient descent updating rule ("rolling down the hill"):

$$w_k \rightarrow w_k' = w_k - \eta\frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b_l' = b_l - \eta\frac{\partial C}{\partial b_l}$$

- What is the potential problem?

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# Appendix | Learning with gradient descent

- The gradient descent updating rule: $v \rightarrow v' = v - \eta \nabla C$

- Rewriting the gradient vector:

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

$$\nabla C = \frac{1}{n} \sum_x \nabla C_x$$

- Stochastic gradient descent (SGD) to speed up the learning:

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C$$

  - The mini-batch ("baby"-training inputs): $X_1, X_2, \ldots, X_m$

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html

# Appendix | Learning with gradient descent

- The stochastic gradient descent updating rule:

$$w_k \rightarrow w_k' = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial wk}$$

$$b_l \rightarrow b_l' = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial bl}$$

  - The mini-batch ("baby"-training inputs): $X_1, X_2, \ldots, X_m$

- What is an epoch of training?

Source: Michael A. Nielsen. (2015). *Neural Networks and Deep Learning.* Determination Press. Link: http://neuralnetworksanddeeplearning.com/index.html