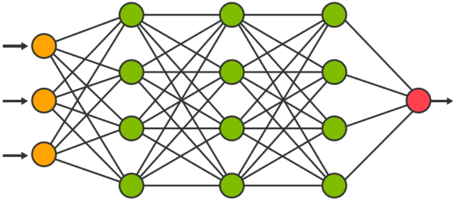
Introduction to Neural Networks

ML Instruction Team, Fall 2022

CE Department
Sharif University of Technology



Biological Analogy

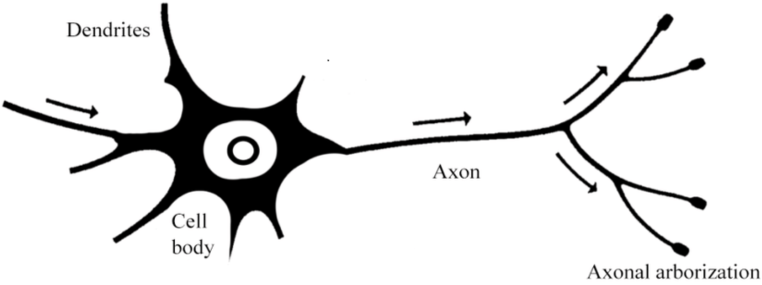


Figure: Anatomy of a biological neuron [1].

Activation Functions

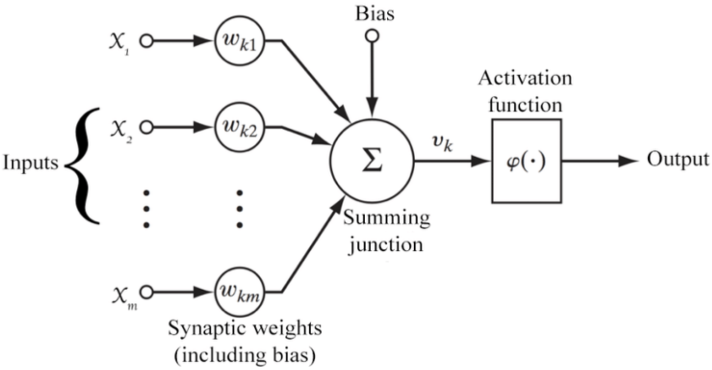


Figure: Neural network neuron [1].

Activation Functions

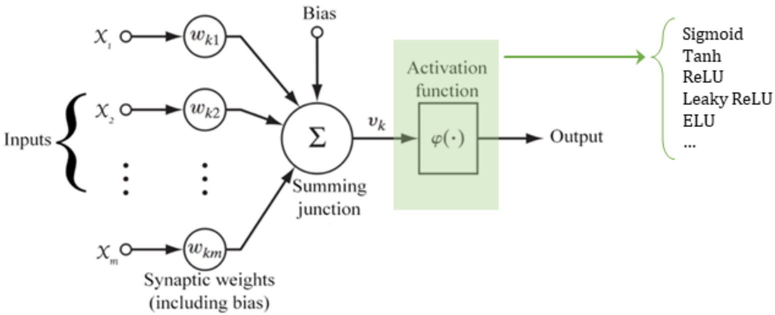
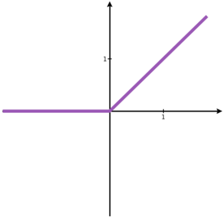
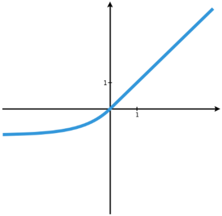
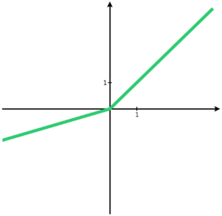
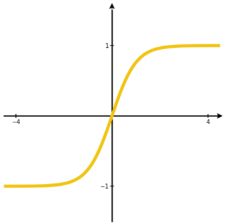
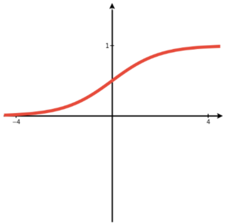
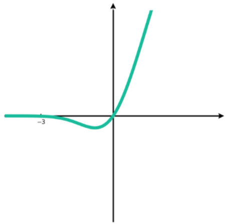


Figure: Activation function

Activation Functions

ReLU	ELU	Leaky ReLU
$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$	$f(x) = \begin{cases} x & x > 0 \\ \alpha(e^{x} - 1) & x \leq 0 \end{cases}$	$f(x) = \begin{cases} x & x \geq 0 \\ 0.01x & x < 0 \end{cases}$
		

Activation Functions

Tanh	Sigmoid	GELU
$f(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$	$\sigma(x) = \dfrac{1}{1 + e^{-x}}$	$f(x) = \dfrac{1}{2}x\left(1 + \text{erf}\left(\dfrac{x}{\sqrt{2}}\right)\right)$
		

Softmax

$$f(x) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad i = 1, \cdots, J$$

Gradient Descent

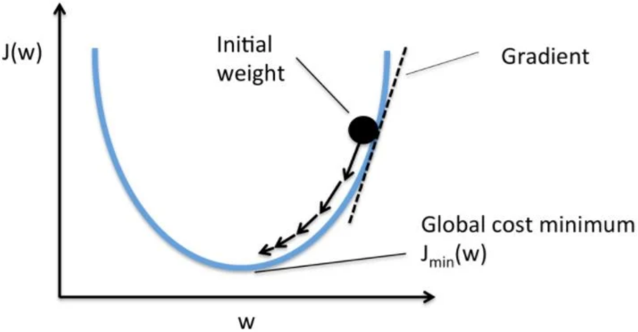


Figure: Gradient descent [3].

Gradient Descent

- Let's define our problem:
  - We have dataset $\mathcal{D} = \{x^i, y^{(i)}\}_{i=1}^{n}$.
  - f is a single layer perceptron.
  - Define $\hat{y}^{(i)} = f(x^{(i)})$.

- We want to minimize following cost function:

$$\mathcal{J}(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

- We are going to use gradient descent algorithm. $\boldsymbol{w}$ will be updated as follows:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta \nabla_{\boldsymbol{w}} \mathcal{J}$$

Gradient Descent

- Let's find $\nabla_{\boldsymbol{w}} \mathcal{J}$:

$$\begin{aligned}
\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i (y^{(i)} - \hat{y}^{(i)})^2 \\
&= \frac{1}{2} \sum_i \frac{\partial}{\partial w_j} (y^{(i)} - \hat{y}^{(i)})^2 \\
&= \frac{1}{2} \sum_i 2(y^{(i)} - \hat{y}^{(i)}) \frac{\partial}{\partial w_j} (y^{(i)} - \hat{y}^{(i)}) \\
&= \sum_i (y^{(i)} - \hat{y}^{(i)}) \frac{\partial}{\partial w_j} \left(y^{(i)} - \sum_j w_j x_j^{(i)} \right) \\
&= \sum_i (y^{(i)} - \hat{y}^{(i)})(-x_j^{(i)})
\end{aligned}$$

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = -\eta \sum_i (y^{(i)} - \hat{y}^{(i)})(-x_j^{(i)}) = \eta \sum_i (y^{(i)} - \hat{y}^{(i)}) x_j^{(i)}$$

$$\mathbf{w} := \mathbf{w} + \Delta \mathbf{w}$$

Thank You!

Any Question?

References

- M. D. Ergün Akgün, "Biological and neural network neuron," 2018. https://www.researchgate.net/publication/326417061_Modeling_Course_Achievements_of_Elementary_Education_Teacher_Candidates_with_Artificial_Neural_Networks.
- L. Nalborczyk, "A gentle introduction to deep learning in r using keras," 2021. https://www.barelysignificant.com/slides/vendredi_quanti_2021/vendredi_quantis#1.
- "Gradient descent." https://subscription.packtpub.com/book/big-data-&-business-intelligence/9781788397872/1/ch01lvl1sec22/gradient-descent.
- F.-F. L. . J. J. . S. Yeung, "Training neural networks," 2018. http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture07.pdf.
- I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
- K. Katanforoosh and D. Kunin, "Initializing neural networks," 2019. https://www.deeplearning.ai/ai-notes/initialization/.