

The background is a solid blue color with various light blue mathematical symbols scattered across it. These symbols include plus signs (+), minus signs (-), multiplication signs (x), and squares (□).

DEEP LEARNING FOUNDATION

Lecture 5

Copyright © 2018 Ankita Sinha

Motivation for Deep Learning

Limits of Traditional Computer Programs

- ✓ Perform Arithmetic Computations Really Fast
- ✓ Explicitly Follow a List of Instructions
- ✓ Solving Equations by Formula

Motivation for Deep Learning

Limits of Traditional Computer Programs

- ✖ Write a Program to Read Someone's Handwriting
- ✖ Automatically detect diseases
- ✖ Assistance in cleaning Our Homes
- ✖ Learning by Example



DEEP LEARNING



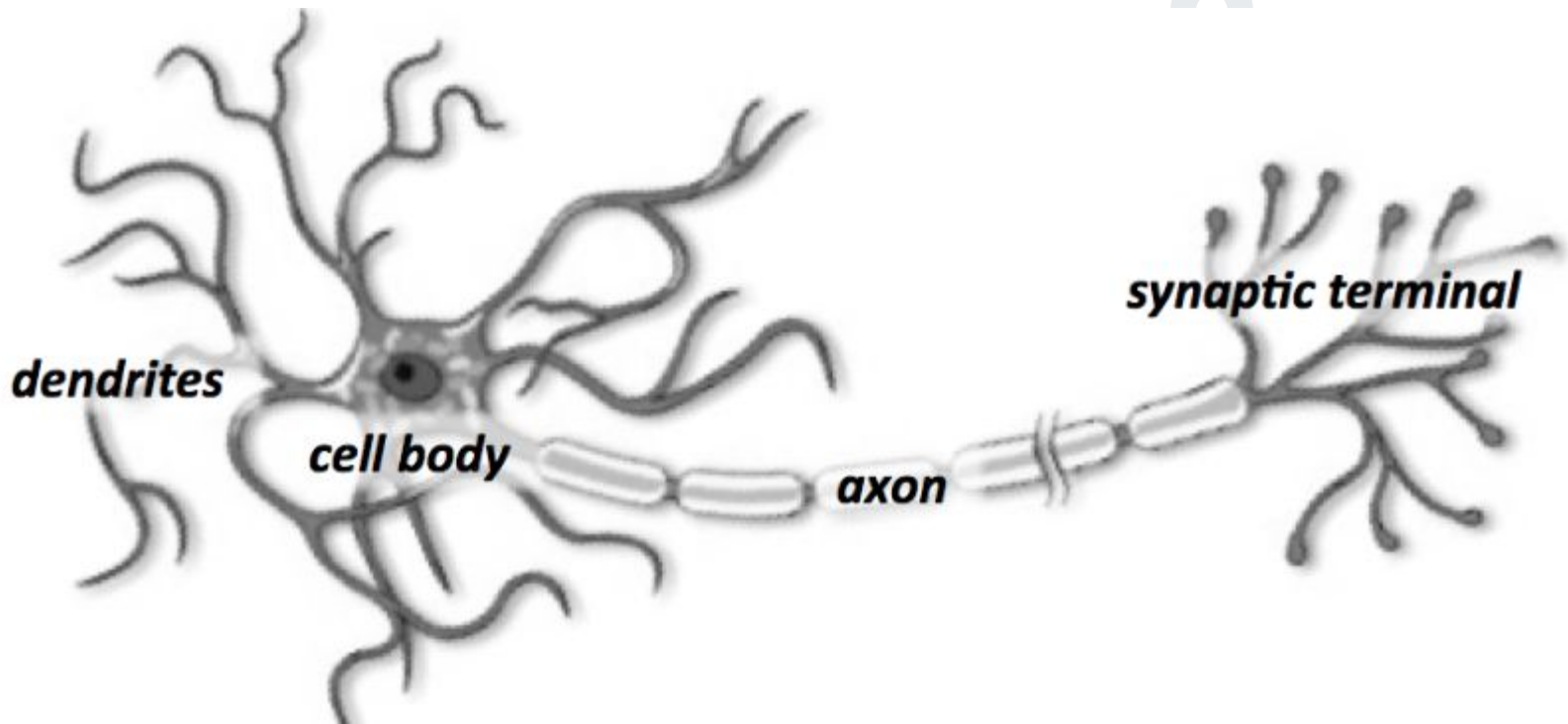
Complex Problems require more complex models.



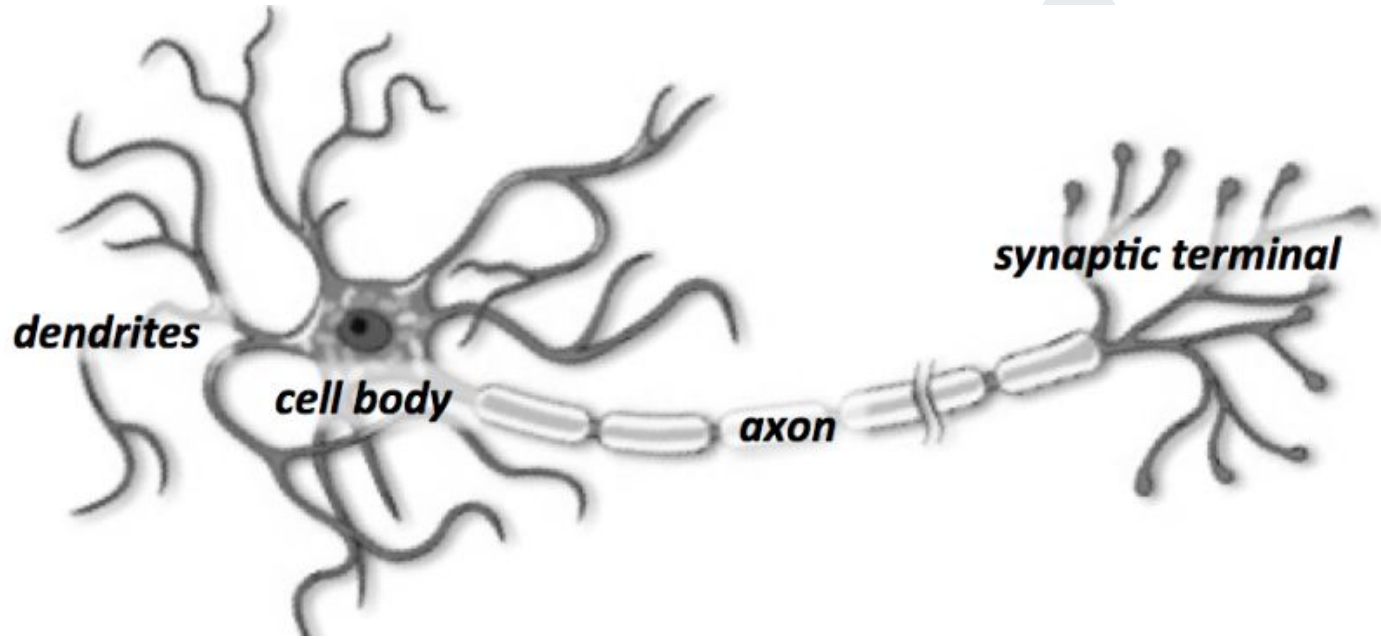
Deep Learning Models are similar to the neural network models built by our brain.

The Neuron

- 💡 A tiny piece of brain contains over 10,000 neurons
- 💡 Each neuron forms over 6,000 connections with other neurons.
- 💡 Building A Massive Biological Network
- 💡 Each neuron receives information from other neurons, process it & sends it to other cells.

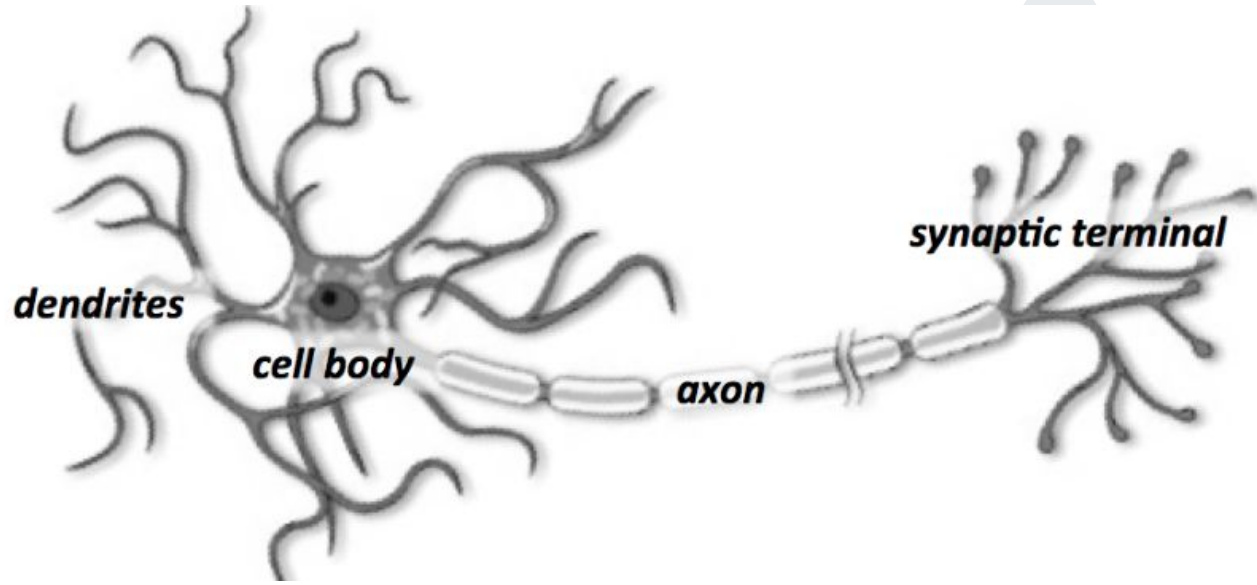


A Biological Neuron Structure



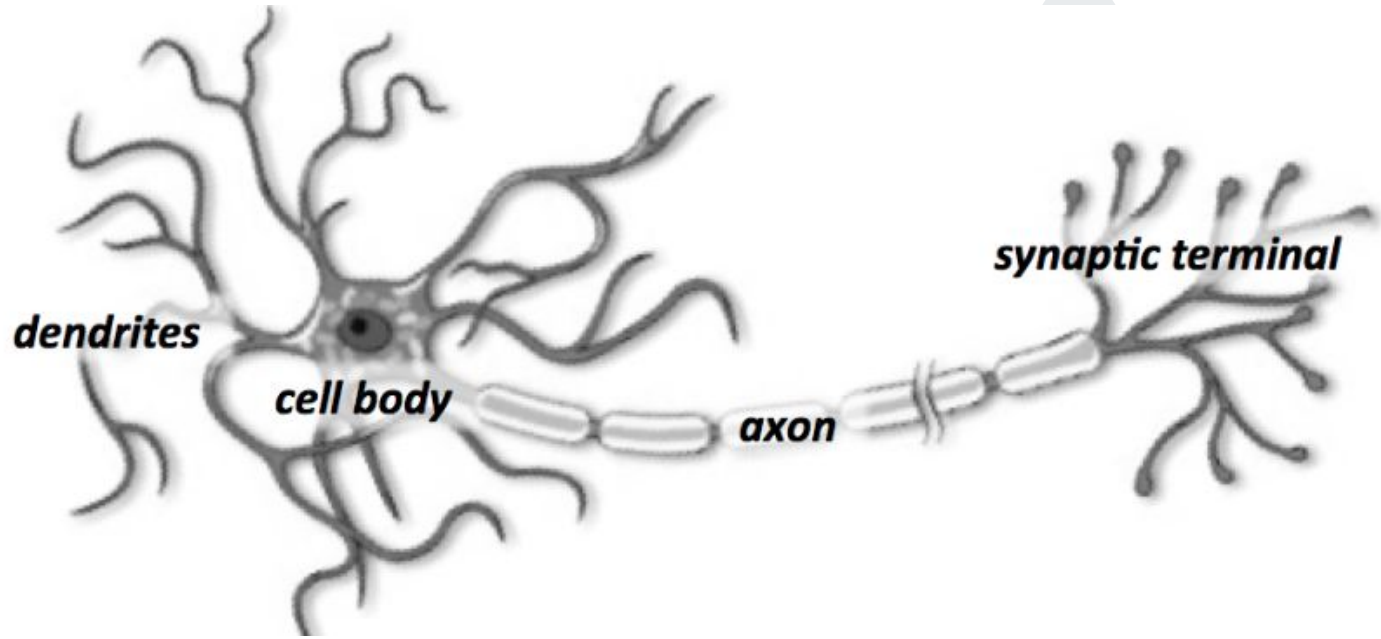
Input (Dendrites)

→ The neuron receives its input along antennae-like structures called **dendrites**.



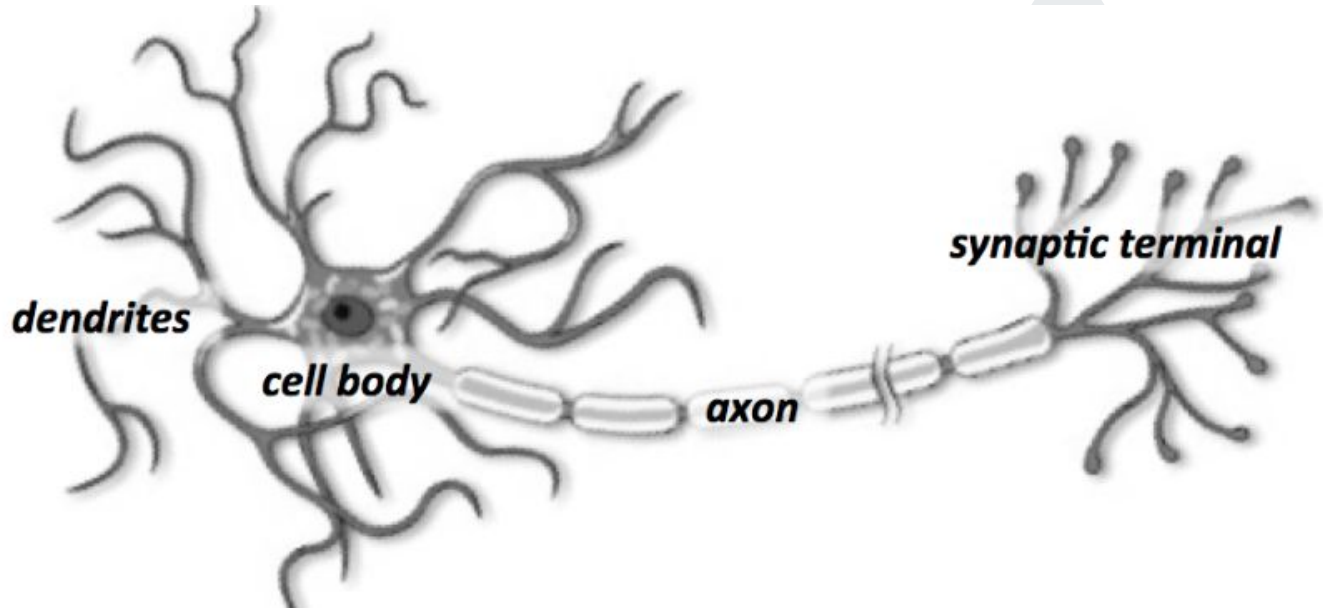
Strengths (Cell Body)

- Each incoming connection is dynamically strengthened or weakened based on how often it is used.
- The strength of inputs are summed together in cell body.



Signal Propagation (axon)

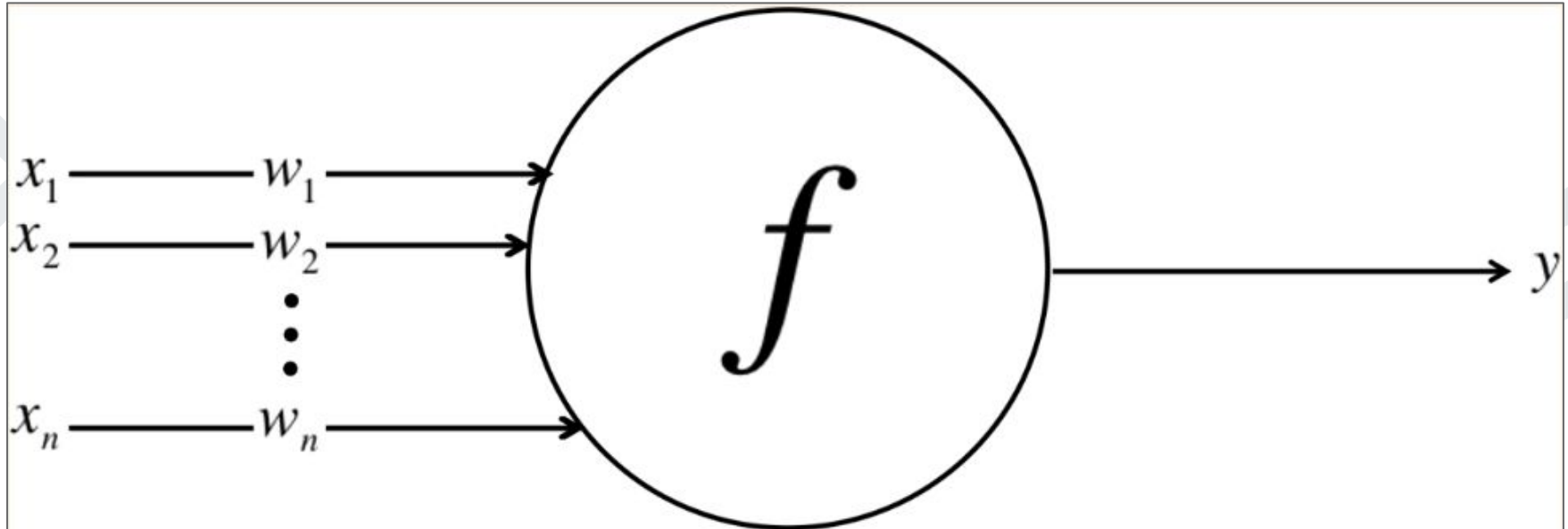
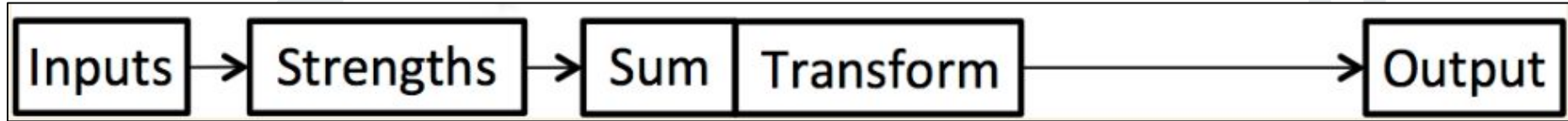
- The sum is then transformed into a new signal that is propagated to other neurons through axon.



Synaptic Terminal (Output)

The strength of each connection determines the contribution of the input to the neuron's output through the synaptic terminal.




Functional Description of A Neuron in Artificial Neural Network



Expressing ANN in Vector Form


- ✖ Reformulate the inputs as $X=[x_1, x_2, \dots, x_n]$
- ✖ Weights of the neuron as $W=[w_1, w_2, \dots, w_n]$
- ✖ Output of the neuron : $y = f(\mathbf{x} \cdot \mathbf{w} + b)$
- ✖ Output = Dot Product of input vector and weight vector, adding the bias (constant) term
- ✖ Neurons \Rightarrow Series of Vector Manipulations

Perceptron


-  Perceptron is a single layer neural network
-  Perceptron input is designed to classify or categorize visual inputs using a linear binary classifier
-  A single linear perceptron is expressed as a single neuron.


Exam Performance


Perceptron as a Neuron

 **GOAL:** Determine how to predict exam performance based on the number of hours of sleep we get and the number of hours we study the previous day.

 x_1 : The number of hours of sleep

 x_2 : The number of hours we spent studying

 $\mathbf{x} = [x_1 \ x_2]^T$: Input Vector

 $\theta = [\theta_0 \ \theta_1 \ \theta_2]^T$: Output Vector

 $h(\mathbf{x}, \theta)$: Model

Exam Performance

Perceptron as a Neuron

$$h(\mathbf{x}, \theta) = \begin{cases} -1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 < 0 \\ 1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 \geq 0 \end{cases}$$

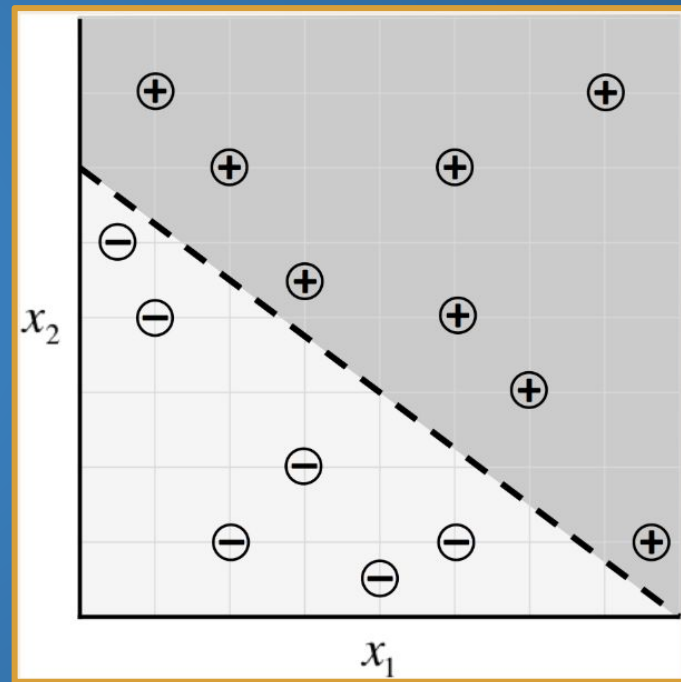
Describe a linear classifier model that divides the coordinate planes into two halves such that for vector parameter, θ model predicts -1 if we perform below average and +1 if we perform above average.

Exam Performance

Perceptron as a Neuron

🔗 **Optimization** : *An optimal parameter vector positions the classifier so that we make as many correct predictions as possible.*

🔗 Optimizer aims to maximize the performance of the model by iteratively tweaking its parameters until error is minimized.

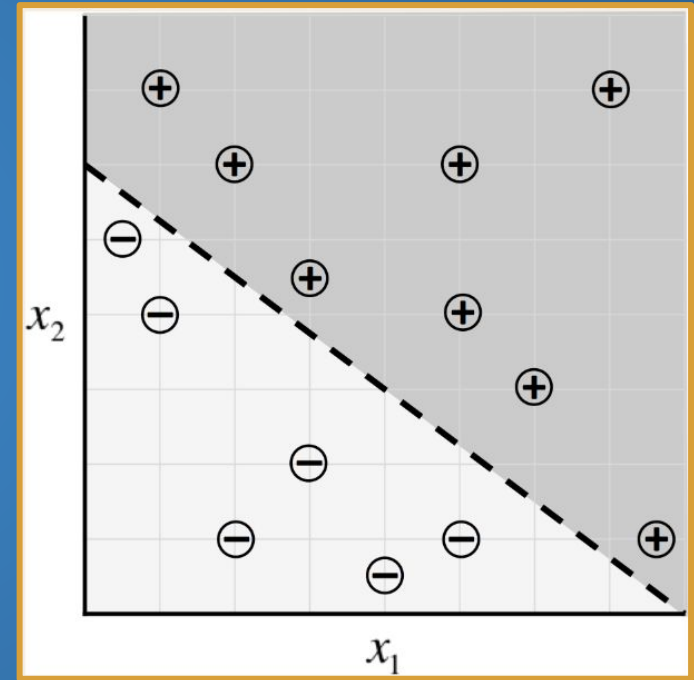


Potential Linear Perceptron

Exam Performance

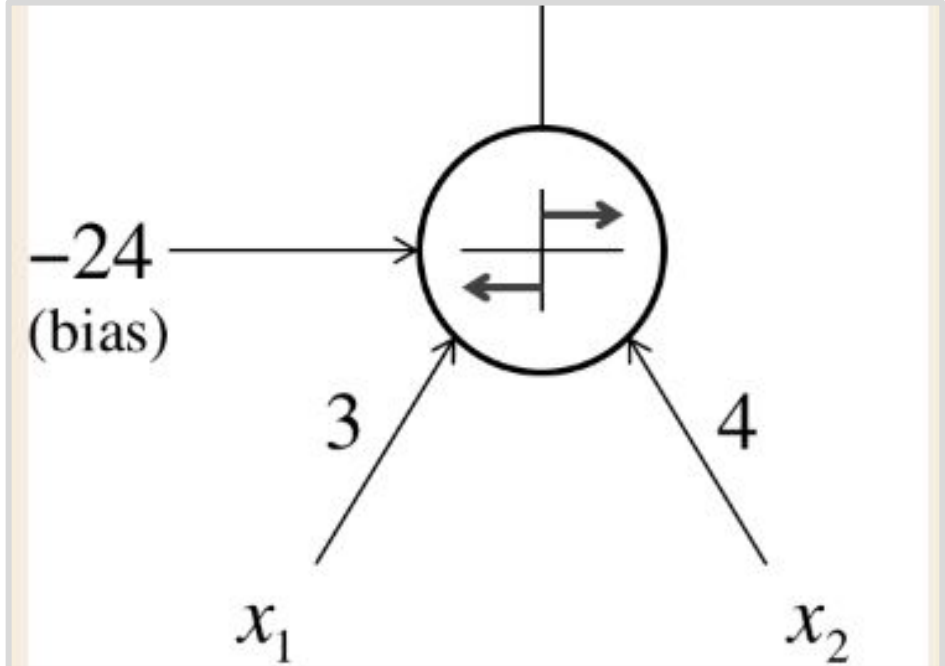
Perceptron as a Neuron

$$h(\mathbf{x}; \theta) = \begin{cases} -1 & \text{if } 3x_1 + 4x_2 - 24 < 0 \\ 1 & \text{if } 3x_1 + 4x_2 - 24 \geq 0 \end{cases}$$

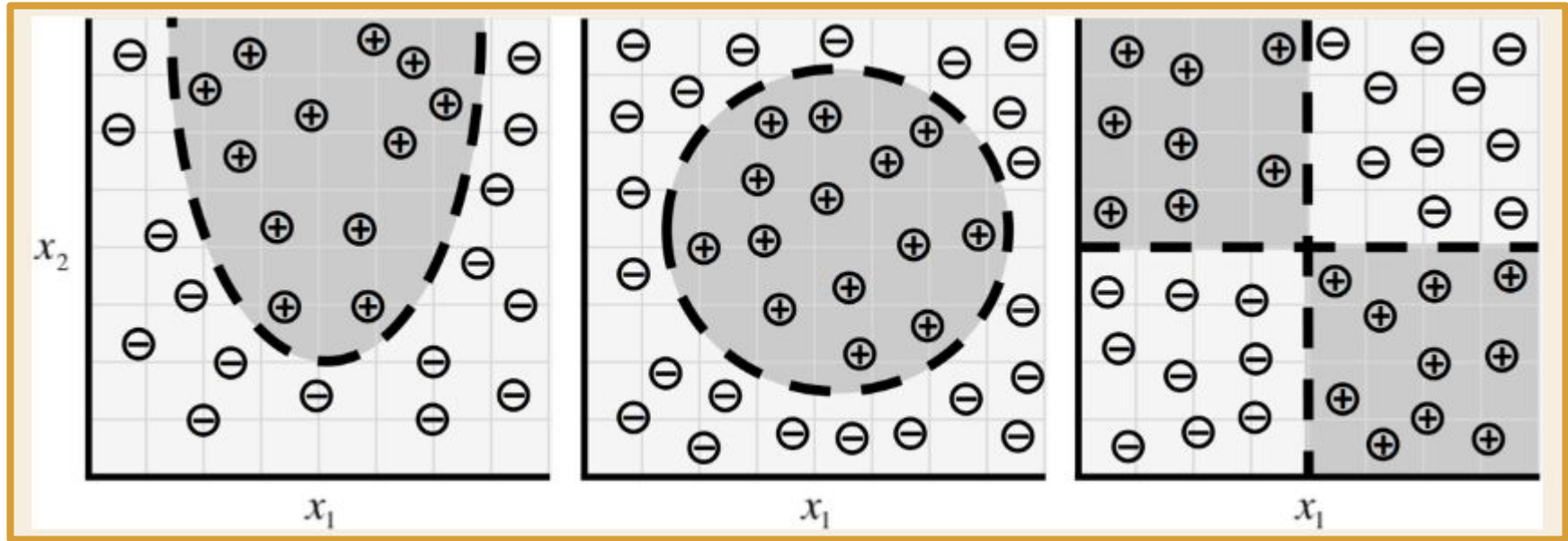


Potential Linear Perceptron

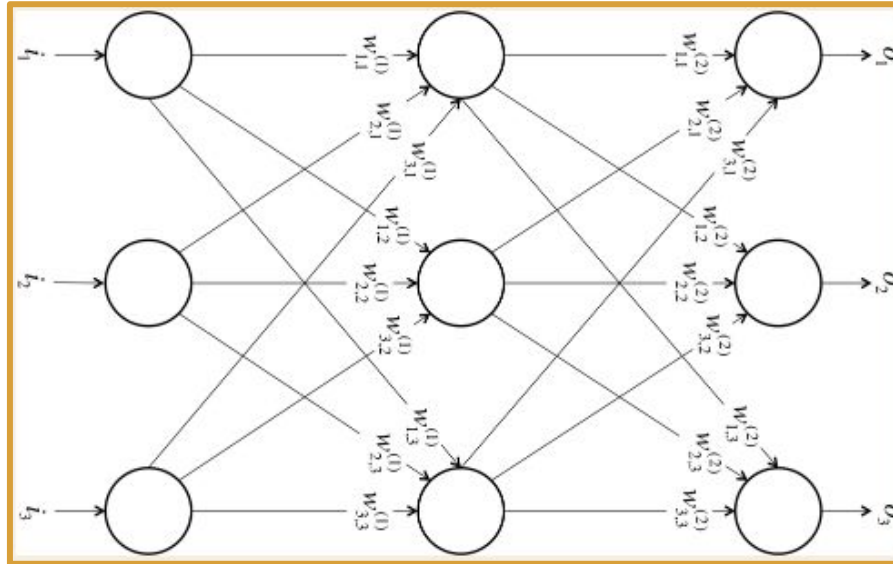
Expressing Linear Perceptron Model as a Single Neuron



Complex Models that cannot be described by Linear Perceptron Model



Feed Forward Neural Networks



Feed Forward Neural Network with Three Layers and Three Neuron Per Layers

Feed Forward Neural Networks

Input

The bottom layer of the network pulls in the input data.



Hidden

The middle layer of neurons are called hidden layers.






Output

The top layer of our network computes our final answer.

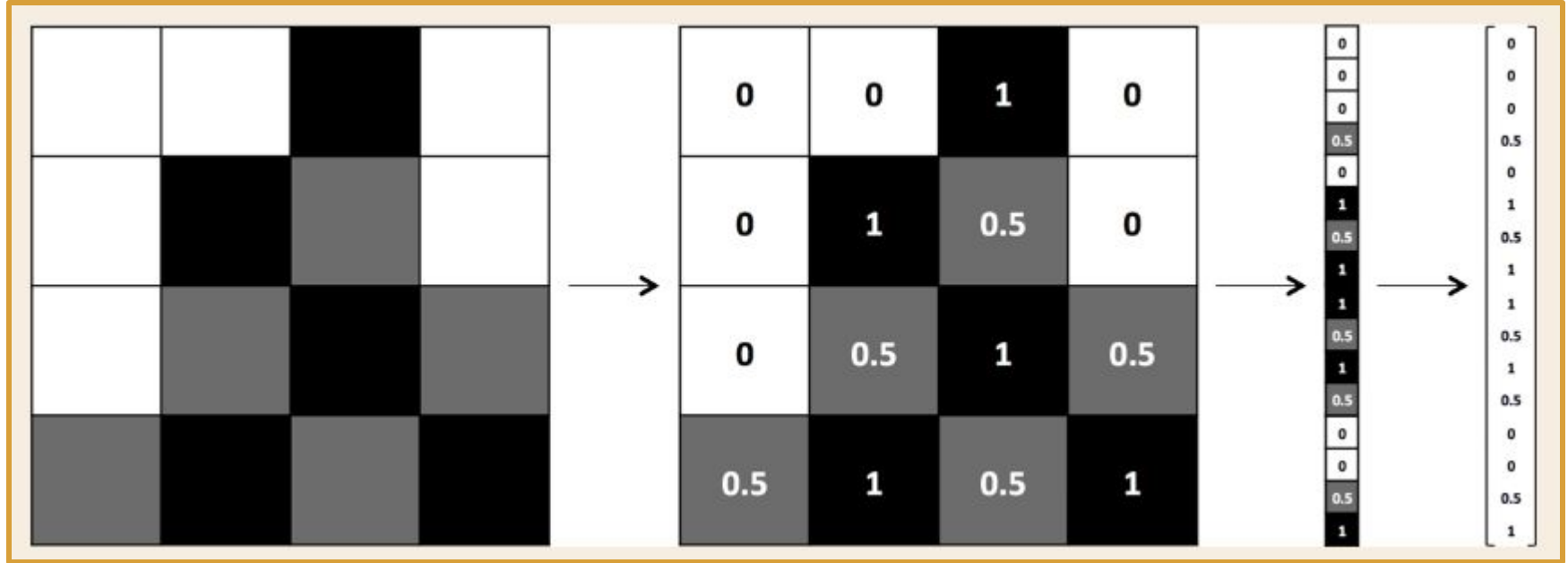
Limitations of Feed Forward Network

-  There are no connections between neurons in the same layer.
-  There are no connections that transmit data from a higher layer to a lower layer.

Features of Feed Forward Network

-  It is not necessary to have same number of neurons in every layer.
-  Each layer forces the network to learn compressed and better representations of the previous input.
-  It is not required that every neuron is connected to every other input neuron and every output neuron.
-  The inputs and outputs are vectorized representations.
-  Neural Networks can be represented as a series of vector and matrix operations.

Vectorized Representation of an Image



Vectorized Representation of an Image



Possible Inputs

Individual pixel RGB values in an image represented as a vector.



Output

Two neurons for four possible options

[1,0] - if image is a dog

[0,1] - if image is a cat

[1,1] - if contains both

[0,0] - if contains Neither

Mathematical Expression of Neural Network

Let vector $\mathbf{x} = [x_1, x_2, x_3, x_4, \dots, x_n]$ be input to the i^{th} layer.

Goal : Find the vector $\mathbf{y} = [y_1, y_2, y_3, y_4, \dots, y_m]$ produced by propagating the input through the neurons.

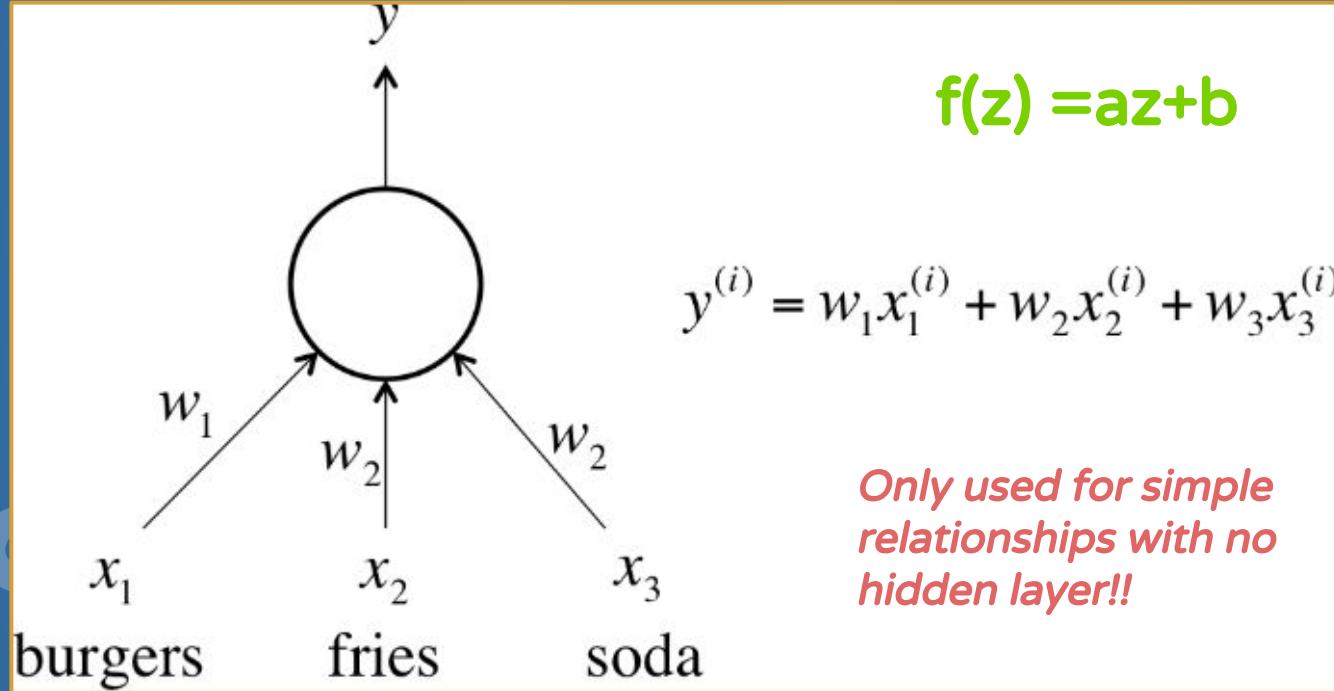
$$\mathbf{y} = f(\mathbf{W}^T \mathbf{x} + \mathbf{b})$$

\mathbf{W} - Weight matrix of size $n \times m$ where each column corresponds to a neuron..

T - Transformation function applied on matrix element wise.

$w(j,k)$ - Weight of the j^{th} input connection pulling the k^{th} element of the column (o/p) layer

Linear Neurons



Sigmoid, Tanh and ReLU Neurons

📌 Some Sort of Non-Linearity is required to describe complex relationships.

📌 3 types of neurons that introduce Non-Linearity

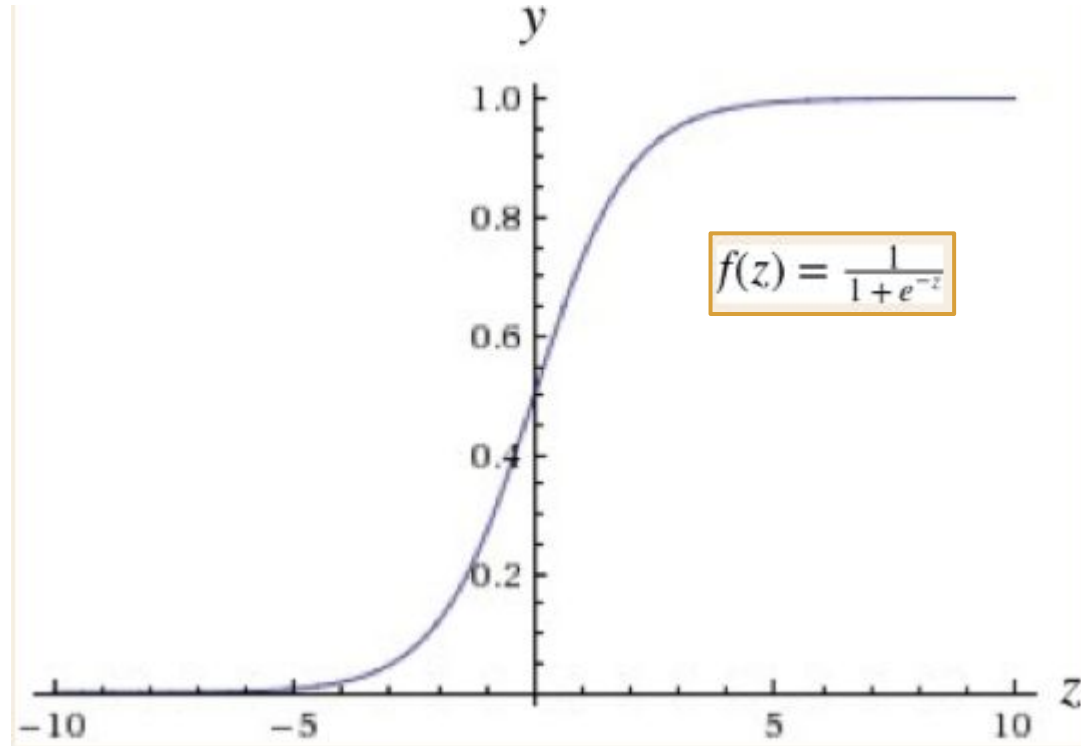
📌 Sigmoid Neurons

📌 Tanh Neurons

📌 ReLU Neurons

Sigmoid Neurons

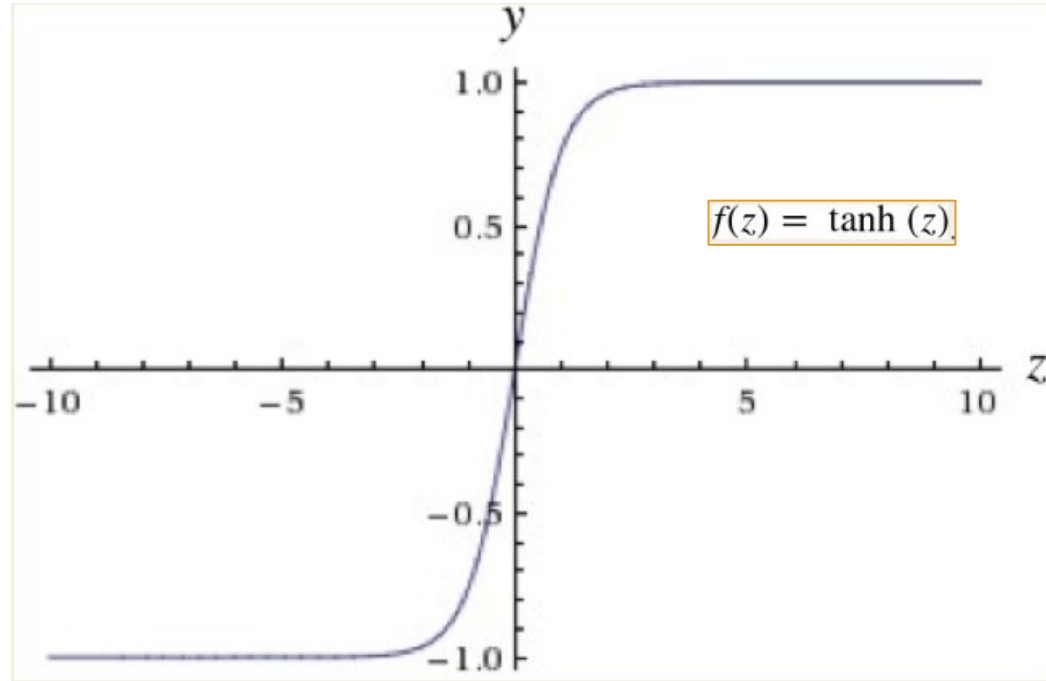
- 📌 The neuron assumes an S-Shape.
- 📌 When the logit is very small, the output of a logistic neuron is close to **0**.
- 📌 When logit is very high, the output of the logistic neuron is close to **1**.



Output of a Sigmoid Neuron as z varies

Tanh Neurons

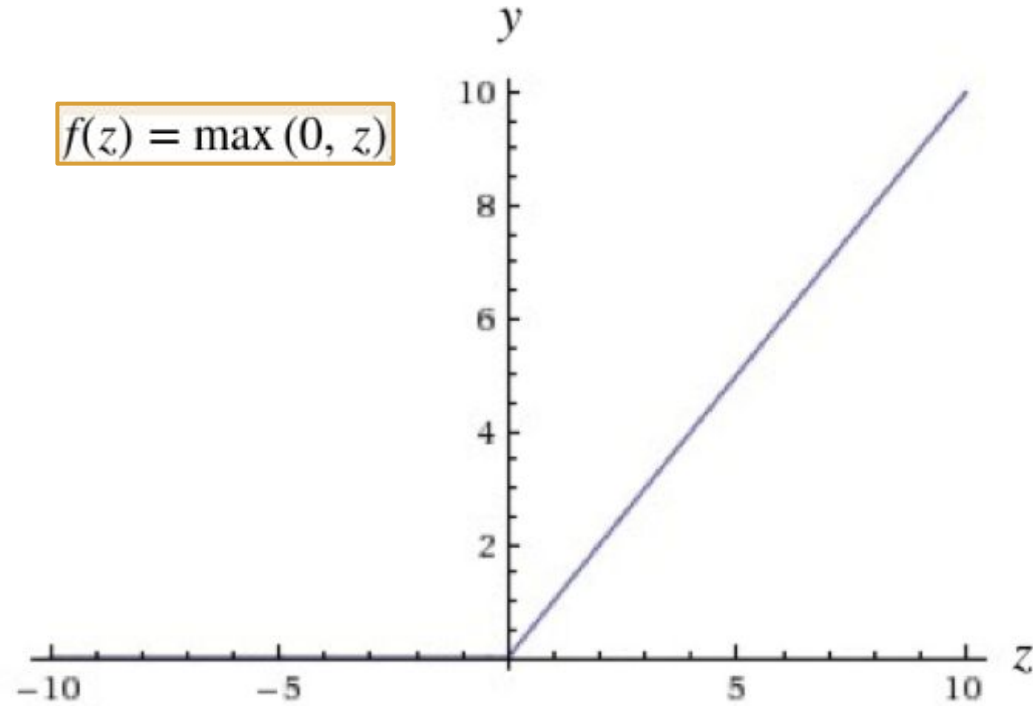
- ✚ The neuron assumes an S-Shaped nonlinearity.
- ✚ The output of tanh neurons range from -1 to 1.
- ✚ Tanh neurons are preferred over sigmoid neurons because it is zero centered.



Output of a tanh neuron as z varies

ReLU Neurons

- 📌 ReLU stands for Restricted Linear Unit.
- 📌 It results in a hockey-stick shaped response $_/_$.
- 📌 Most popularly used in computer vision problems.
- 📌 It is zero when $z < 0$ and linear with slope 1 when $z > 0$



Output of a tanh neuron as z varies

Softmax Output Layer

- 📌 Softmax layer is a special output layer where the output of a neuron in a softmax layer depends on the outputs of all other neurons in its layer.
- 📌 The sum of all the outputs to be equal to 1.
- 📌 Example : Handwritten Digits Dataset
- 📌 Output Vector is a probability distribution over a set of mutually exclusive labels viz ... 0,1,2,3,4,5,6,7,8,9 i.e.,
$$[p_0 \ p_1 \ p_2 \ p_3 \ \dots \ p_9]$$

This shows how confident we are in our predictions.
Output Vector is of the form
$$\sum_{i=0}^9 p_i = 1;$$

Softmax Output Layer

- 📌 A strong prediction would have a single entry in the vector close to one.
- 📌 The remaining entries would be close to zero.
- 📌 A weak prediction would have multiple possible labels that are more or less equally likely.

Training Feedforward Neural Networks

📌 How do we figure out what the optimal weights for all the connections in our neural network should be??

$$E = \frac{1}{2} \sum_i (t^{(i)} - y^{(i)})^2$$

📌 Optimal Weights \Rightarrow Weights that minimize the errors the most

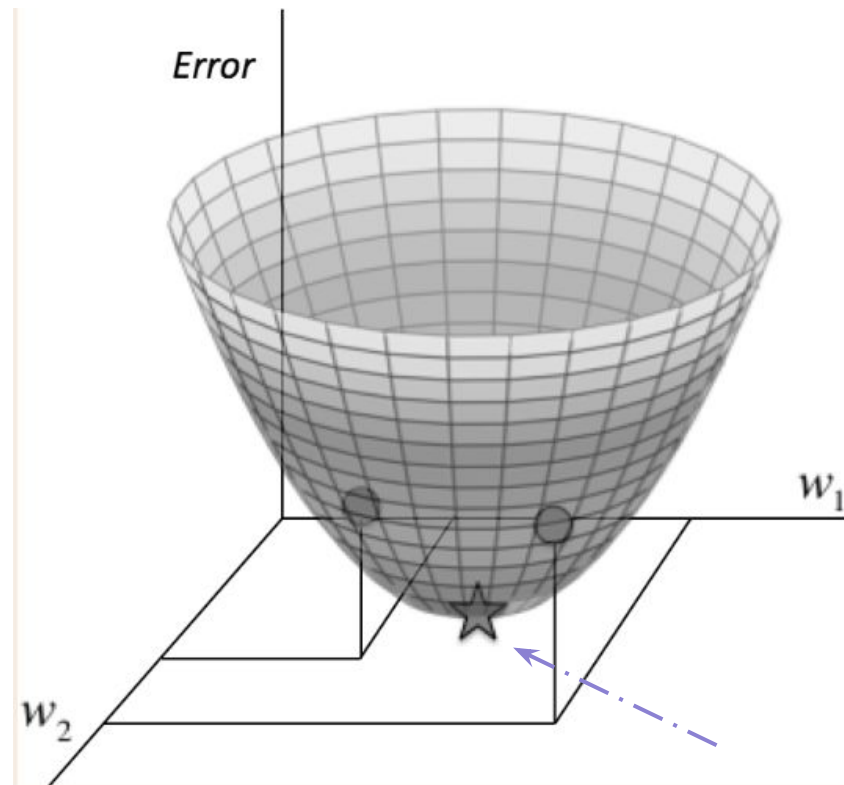
The closer E to 0, the better the model is !!

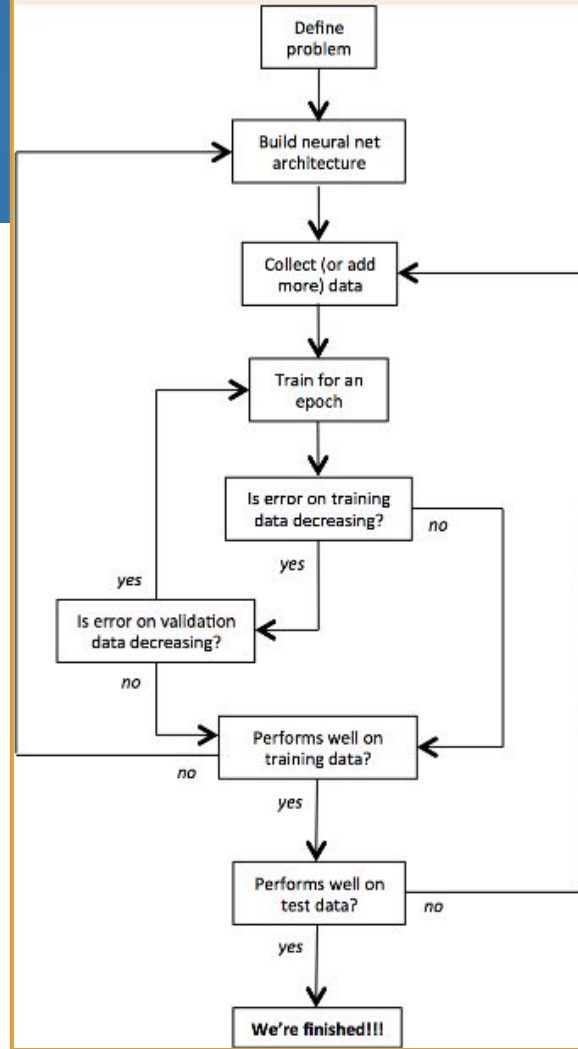
📌 The squared error is zero when our model makes a perfectly correct prediction on every training example.

Goal : Select Parameter Vector θ
i.e the value of all weights, such that E is as close to 0 as possible.

Gradient Descent

- Imagine a three dimensional space where horizontal dimensions correspond to weights w_1 and w_2 .
- The vertical dimension corresponds to the value of the error function.
- If we consider the errors we make over all possible weights we get a quadratic bowl where the minimum error is the center of ellipses.





Detailed workflow
of training and
evaluating a deep
learning model

Preventing Overfitting in Deep Neural Networks

- One Method of preventing overfitting is called Regularization
- Regularization modifies the objective function that we minimize by adding additional terms that penalize large weights
- The objective function becomes $Error + \lambda f(\theta)$ where $f(\theta)$ grows larger as the components of θ grow large and λ is the regularization strength
- λ acts as a hyperparameter that controls the capacity of the network.

Preventing Overfitting in Deep Neural Networks

- The value of λ determines how much we want to protect against overfitting.
- $\lambda = 0$ implies, that we do not take any measures against the possibility of overfitting.
- If λ is too large, then our model will prioritize keeping θ as small as possible.
- Selecting right λ can take some trial and error.

Types of Regularization

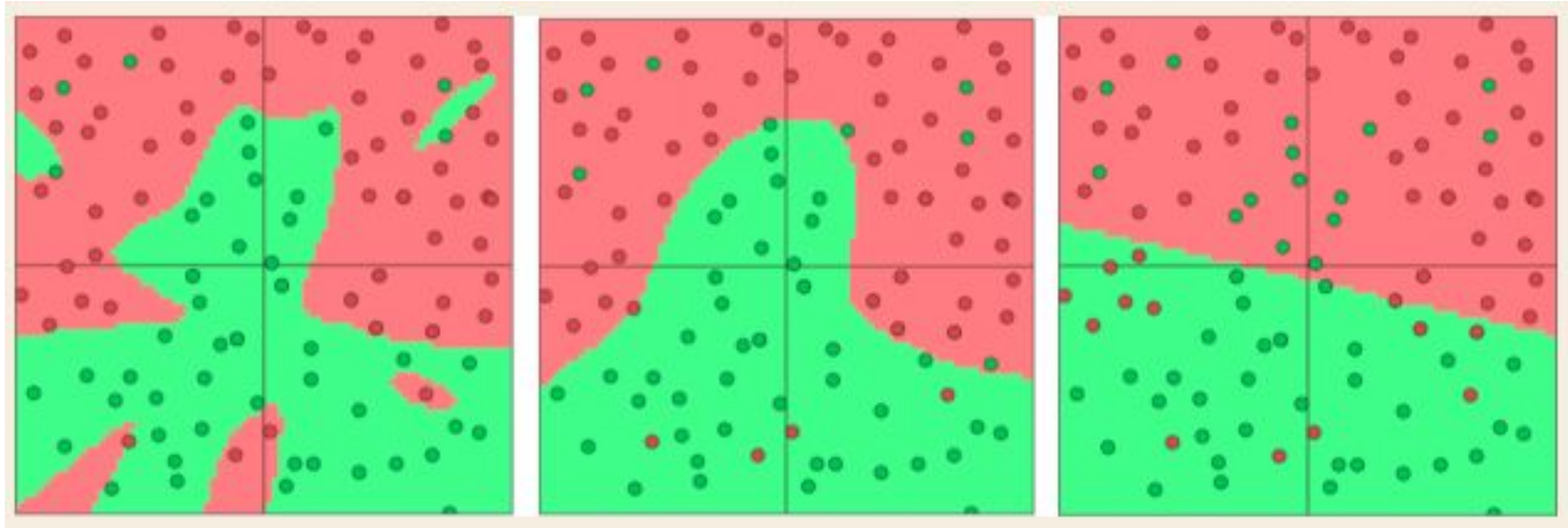
L2 Regularization

- Also Known as Weight Decay.
- Most commonly used.
- Heavily Penalizes Peaky Weight Vectors
- Prefers Diffuse Weight Vectors
- Encourages the network to use all of its inputs a little rather than some of its inputs a lot.

Types of Regularization

L2 Regularization

- Implemented by Augmenting the error function with the squared magnitude of all weights in the neural network.
- For every weight w in the neural network, add $\frac{1}{2}\lambda w^2$ to the error function.



Visualization of neural networks with 2 input layer, 2 output layer and a hidden layer with 20 neurons trained with L2 regularization strengths of 0.01, 0.1 and 1 (from left to right)

Types of Regularization

L1 Regularization

- Add the term $\lambda |w|$ for every weight w in the network
- It leads the weight vectors to come closer to zero during optimization
- Neurons with L1 regularization use a small subset of their most important inputs.

Types of Regularization

L1 Regularization

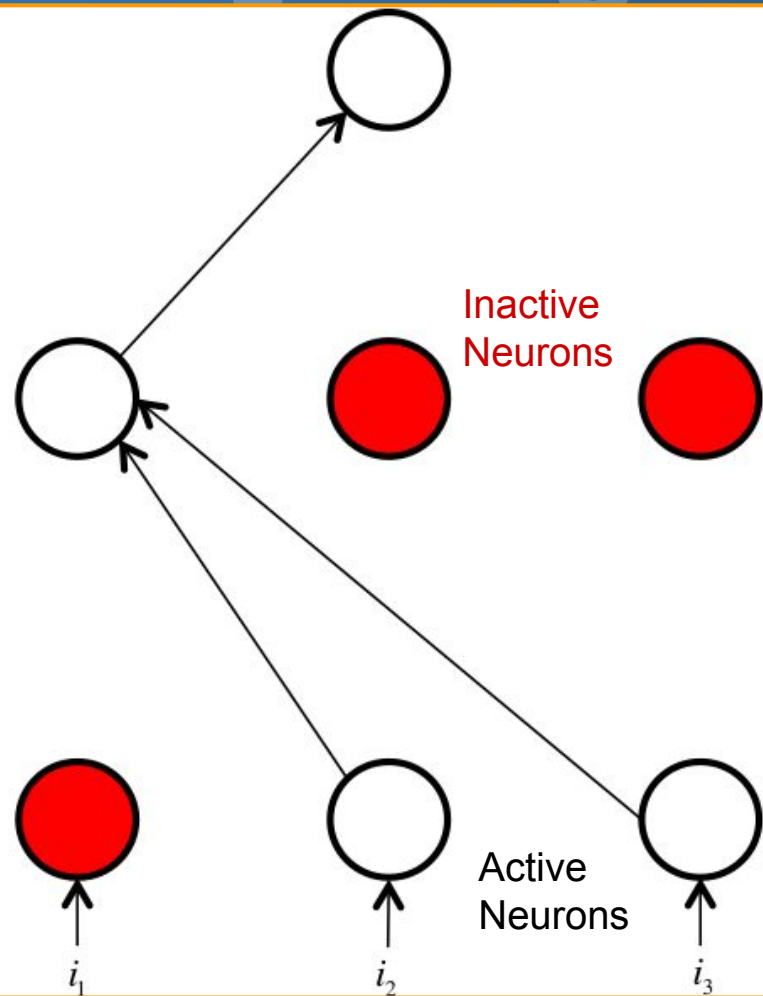
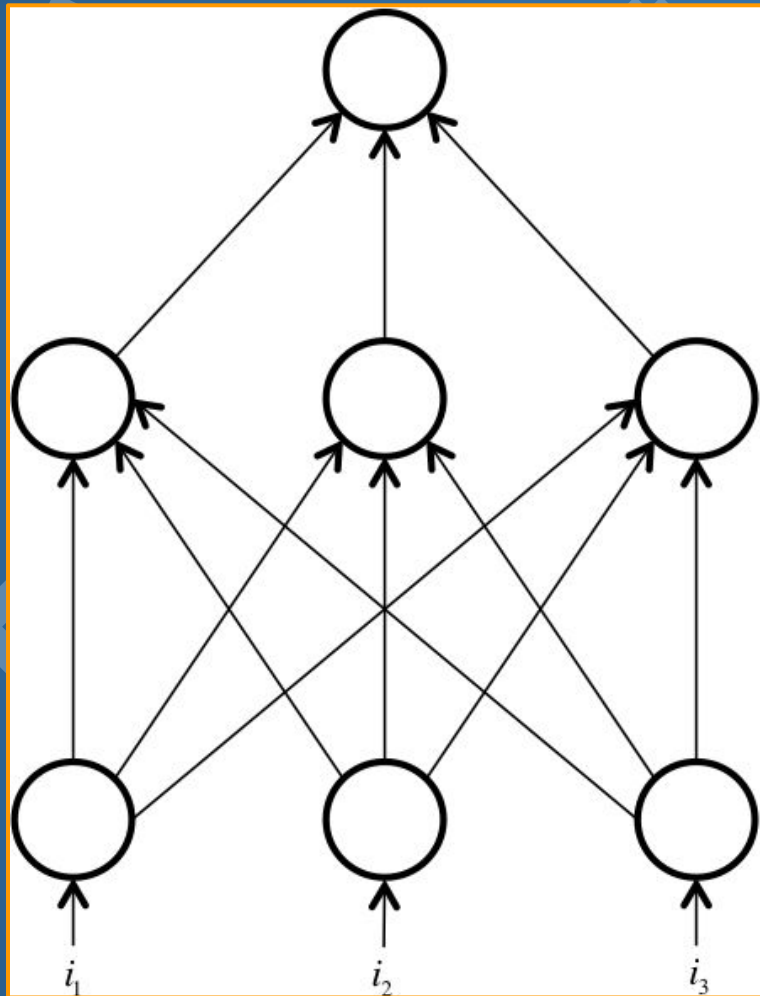
- Neurons become resistant to noise in the inputs.
- L1 is useful in finding out exactly which features are contributing to a decision.
- If this level of feature analysis is not necessary, go for L2 regularization.
- Empirically L2 regularization performs better.

Dropout

- Dropout is a method of preventing overfitting in deep neural networks
- It provides a way of approximately combining many different neural network architectures efficiently
- It prevents the network from becoming too dependent on any one of neuron

Dropout

- While training, dropout is implemented by either keeping a neuron active with some probability or keeping it inactive by setting it to zero.
- This forces the network to be accurate even in the absence of certain information.



Reading Materials

1. [Deep Learning Paper \(Nature Review\) by Geoffrey Hinton et. al.](#)
2. [Deep Learning MIT Technology Review](#)

You can find me at
@ankita90

