CMSC 190 Part 1

Forecasting Weekly Dengue Cases by Integratin Google Earth Engined Based Risk Prediction and Cloud Based Learning

Nathaniel Aquino

University of the Philippines - Baguio

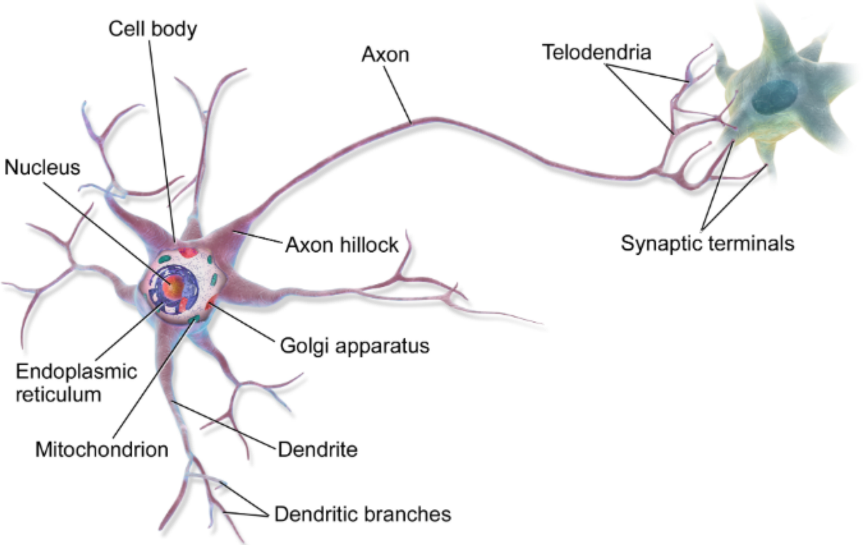
November 4, 2025

Overview

Related Literature Review: Neural Networks

- **Neural Network:**
  "A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use." [1]
- **Neuron (Biological):**
  "The neuron is the basic signaling unit of the nervous system... Neurons are specialized for sending signals over long distances... They communicate with one another at specialized contact points called synapses." [2]
- **Network (General Definition):**
  "A network is a collection of objects (nodes) connected by links (edges)." [3]

Related Literature Review: Neural Network



Review Literature Review: Artificial Neural Network

Definitions

Zhang et al. (2016)

An artificial neural network (ANN) mimics the human brain in processing input signals and transforms them into output signals. It is a modelling algorithm that allows for non-linearity between feature variables and output signals.

Maind & Wankar (2014)

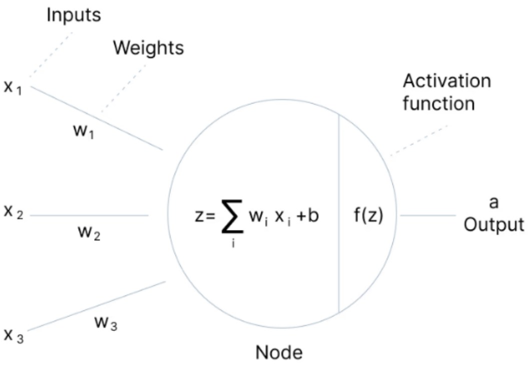
An artificial neural network is an information-processing paradigm inspired by the way biological nervous systems process information. It consists of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems.

Key Components of a Neuron

Core definitions and their roles

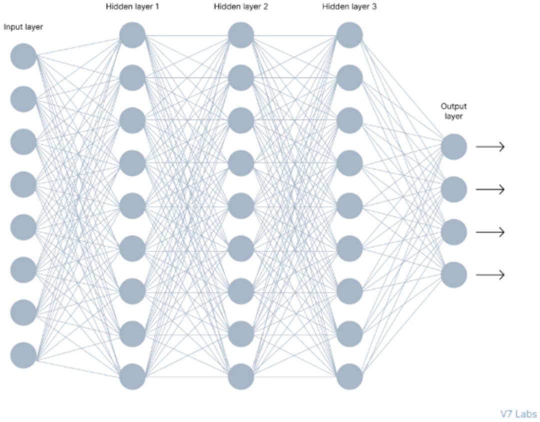
- **Inputs (x)**
  - A vector of numeric feature values (e.g., pixel data, predictors) fed into a neuron at a given time step [4].
- **Weights (w)**
  - Adjustable parameters that scale the influence of each input. They determine the strength and direction of the connection between neurons [4, 1].
- **Bias (b)**
  - An additive parameter that shifts the activation function, allowing the neuron to activate even if all inputs are zero. It is added to the weighted sum of inputs [4].
- **Activation Function (ϕ)**
  - A non-linear function (like Sigmoid, Tanh, or ReLU) applied to the neuron's weighted sum plus bias ($a = \phi(z)$) to introduce non-linearity, which allows the network to learn complex patterns [4, 5].

Related Literature Review: Neural Network



Schmidhuber (2015):

Characterizes forward propagation in deep networks as computing each neuron's output by applying an activation to an affine transformation of its inputs, layer by layer from inputs to outputs.



Related Literature Review: Forward Propagation

Schmidhuber (2015)

characterizes forward propagation in deep networks as computing each neuron's output by applying an activation to an affine transformation of its inputs, layer by layer from inputs to outputs.

Related Literature Review: Forward Propagation and Activation

The flow of prediction and introducing non-linearity

Forward Pass

- The network takes input (x), processes it through weighted connections (W), and generates a prediction ($\hat{y}$).
- **Activation Functions** (ϕ) introduce non-linearity [4].

Single Neuron Formula:

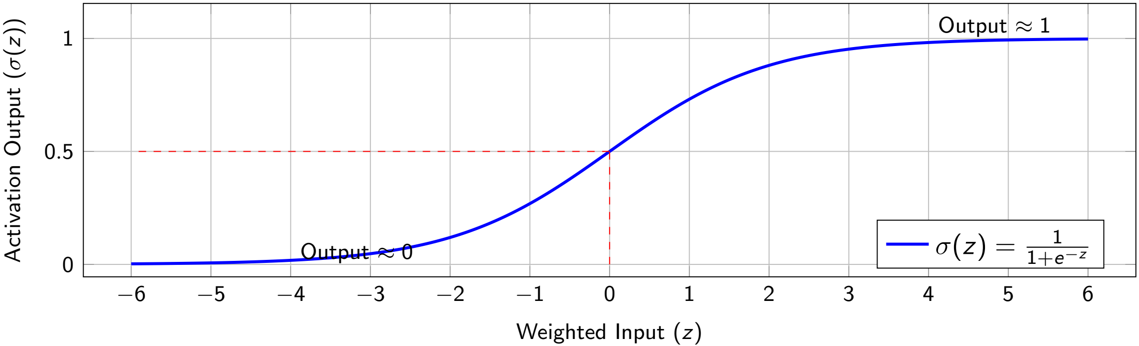
$$a = \phi(z) \quad \text{where} \quad z = (\mathbf{w} \cdot \mathbf{x}) + b$$

Activation Example: Sigmoid

- Restricts outputs between 0 and 1.
- Formula: $\sigma(z) = \dfrac{1}{1 + e^{-z}}$
- Often interpreted as probability.

Visualization: The Sigmoid Activation Function

How the Sigmoid (σ) transforms input



Defining the Error: Cost vs. Loss

The metric optimization seeks to minimize

Loss (Single Example)

$$\text{Loss}_i = (\hat{y}^{(i)} - y^{(i)})^2$$

Measures the error for one data point.

Cost (Overall Objective)

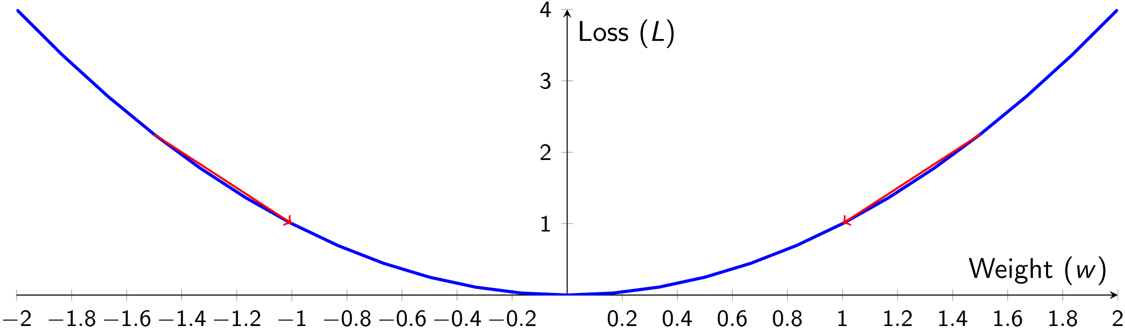
$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})^2$$

Average loss defines the **loss surface**.

Visualization: The Cost Function Loss Surface

The landscape that Gradient Descent navigates

Cost Function $L(w) = (y - wx)^2$



Backpropagation: Core Concept

How neural networks learn through gradient flow

- Backpropagation computes gradients of the loss function with respect to each weight.
- Uses the **chain rule** of calculus to propagate errors backward [6].
- Updates weights to minimize the loss:

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial L}{\partial w_{ij}}$$

Mathematical Formulation

Gradient computation using the chain rule

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial a_j} \cdot \frac{\partial a_j}{\partial z_j} \cdot \frac{\partial z_j}{\partial w_{ij}}$$

$$a_j = \sigma(z_j), \quad z_j = \sum_i w_{ij} a_i + b_j$$

$$\text{Error term:} \quad \delta_j = \frac{\partial L}{\partial z_j} = \frac{\partial L}{\partial a_j} \cdot \sigma'(z_j)$$

$$\text{Weight update:} \quad \Delta w_{ij} = -\eta\, \delta_j\, a_i$$

- Gradient Descent is an iterative optimization algorithm used to find the local minimum of a differentiable function by taking steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point.

Vanishing and Exploding Gradients

Why deep networks sometimes fail to learn

- During backpropagation, gradients are multiplied layer by layer:

$$\frac{\partial L}{\partial w} \propto \prod_{l=1}^{L} \sigma'\left(z^{(l)}\right)$$

- If $\sigma'(z) < 1$ (e.g., sigmoid or tanh), the product becomes very small — gradients **vanish** [7].
- If $\sigma'(z) > 1$, the product grows uncontrollably — gradients **explode** [8].
- Both make weight updates unstable or ineffective.

Effects on Learning
Impact of unstable gradients

- **Vanishing Gradient:**
  - Early layers stop learning (near-zero updates).
  - Model converges slowly or not at all.
- **Exploding Gradient:**
  - Weights grow excessively large.
  - Loss oscillates or diverges.
- These issues are common in deep networks using sigmoid or tanh activations [9].

Gradient Descent as a Stabilizing Mechanism

How optimization counters gradient instability

- Gradient Descent updates parameters gradually to minimize loss:

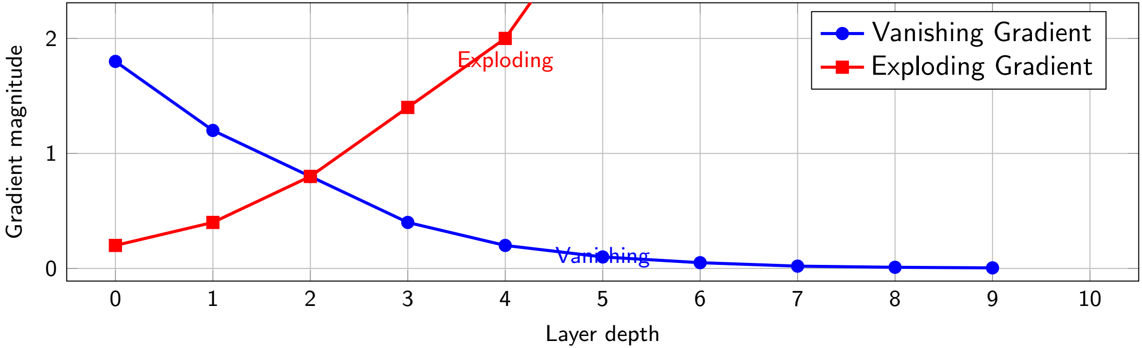
$$w_{t+1} = w_t - \eta \frac{\partial L}{\partial w_t}$$

- The learning rate η controls step size — too large causes divergence, too small slows convergence.

$\boxed{\text{Gradient Descent keeps learning stable by balancing correction size.}}$

Visual Summary

Gradient magnitude across layers



Methodology: Data Preparation

Methodology: Framework for Dengue Risk Prediction

Methodology: Random Forest

Methodology: Long Short Term Memory

Methodology: Long Short Term Memory - Attention Mechanism

References I

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd. Prentice Hall, 1999.

[2] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science*, 4th. McGraw-Hill, 2000.

[3] A.-L. Barabási, *Network Science*. Cambridge University Press, 2016.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org

[5] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010, pp. 807–814.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0

References II

[7] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[8] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998. DOI: 10.1142/S0218488598000094

[9] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. DOI: 10.1109/72.279181