An Introduction to Artificial Neural Networks

Skyrocket your model performance with Artificial Neural Networks. A Walkthrough in Tensorflow!

https://towardsdatascience.com/an-introduction-to-artificial-neural-networks-5d2e108ff2c3

Artificial Neural Network (ANN)

Artificial Neural Network (ANN) is a deep learning algorithm that emerged and evolved from the idea of **Biological Neural Networks of human brains**. An attempt to simulate the workings of the human brain culminated in the emergence of ANN. ANN works very similar to the biological neural networks but doesn't exactly resemble its workings.

ANN algorithm would accept only numeric and structured data as input. To accept unstructured and non-numeric data formats such as Image, Text, and Speech, **Convolutional Neural Networks (CNN),** and **Recursive Neural Networks (RNN)**

are used respectively. In this post, we concentrate only on Artificial Neural Networks.

Biological neurons vs Artificial neurons

Structure of Biological neurons and their functions

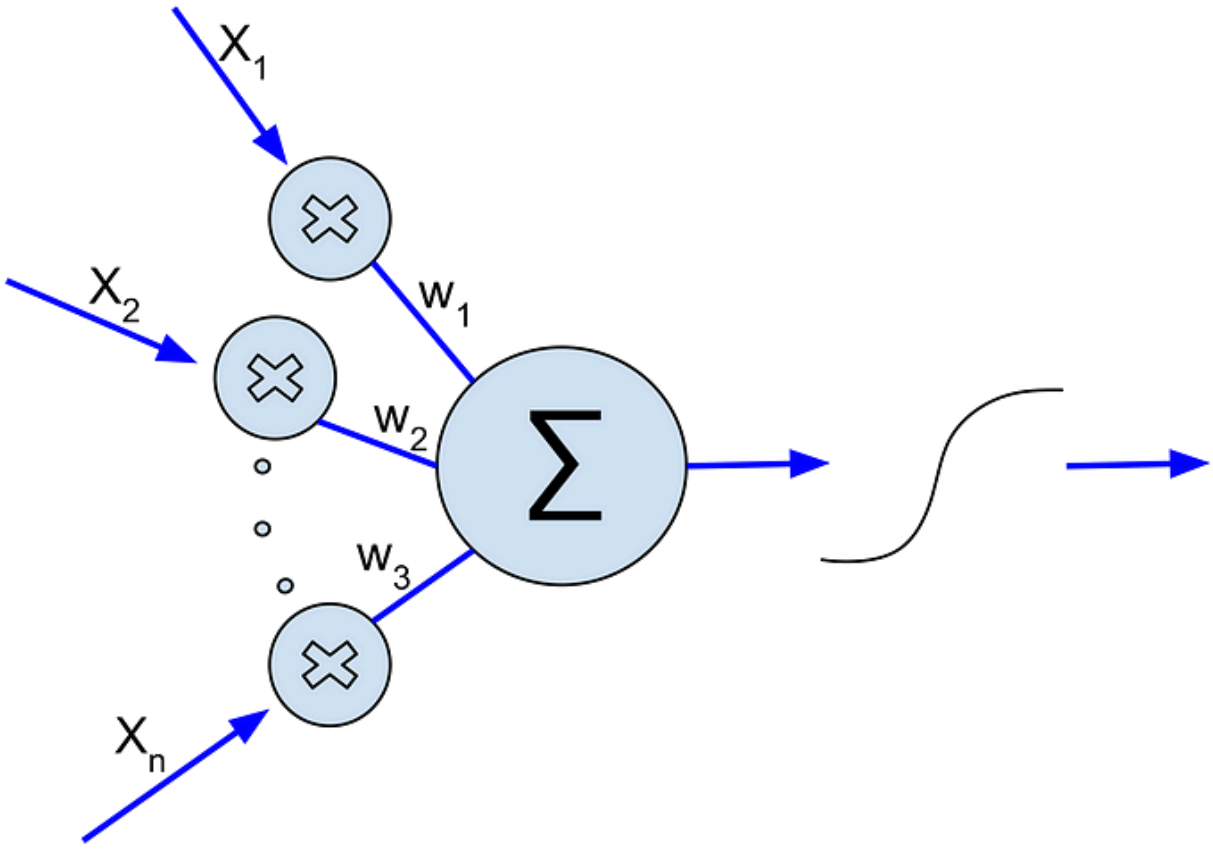
- **Dendrites** receive incoming signals.
- **Soma** (cell body) is responsible for processing the input and carries biochemical information.
- **Axon** is tubular in structure responsible for the transmission of signals.
- **Synapse** is present at the end of the axon and is responsible for connecting other neurons.

Structure of Artificial neurons and their functions

- A neural network with a single layer is called a **perceptron**. A multi-layer perceptron is called **Artificial Neural Networks.**

- A Neural network can possess any number of layers. Each layer can have one or more neurons or units. Each of the neurons is interconnected with each and every other neuron. Each layer could have different **activation functions** as well.
- ANN consists of two phases **Forward propagation and Backpropagation.** The forward propagation involves multiplying weights, adding bias, and applying activation function to the inputs and propagating it forward.
- The backpropagation step is the most important step which usually involves finding optimal parameters for the model by propagating in the backward direction of the Neural network layers. The backpropagation requires an optimization **function** to find the optimal weights for the model.

- ANN can be applied to both **Regression and Classification tasks** by changing the activation functions of the output layers accordingly. (Sigmoid activation function for binary classification, Softmax activation function for multi-class classification and Linear activation function for Regression).



Perceptron. Image Source

Why Neural Networks?

- Traditional Machine Learning algorithms tend to perform at the same level when the data size increases but ANN outperforms traditional Machine Learning algorithms when the data size is huge as shown in the graph below.
- **Feature Learning**. The ANN tries to learn hierarchically in an incremental manner layer by layer. Due to this reason, it is not necessary to perform feature engineering explicitly.
- Neural Networks can handle **unstructured data** like images, text, and speech. When the data contains unstructured data the neural network algorithms such as CNN (Convolutional Neural Networks) and RNN (Recurrent Neural Networks) are used.

How ANN works

The working of ANN can be broken down into two phases,

- Forward Propagation
- Back Propagation

Forward Propagation

- Forward propagation involves multiplying feature values with weights, adding bias, and then applying an activation function to each neuron in the neural network.
- Multiplying feature values with weights and adding bias to each neuron is basically applying **Linear Regression**. If we apply Sigmoid function to it then each neuron is basically performing a **Logistic Regression.**

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

β_0 = *Bias*

β_i = *Weights/ coefficients*

x_i = *Features*

- If we apply **Sigmoid activation function** to $\hat{y}$ then the resultant function would look like,

$$\sigma(\hat{y}) = \frac{1}{1+e^{-\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n}} = \frac{1}{1+e^{-\hat{y}}}$$

Activation functions

- The purpose of an activation function is to introduce **non-linearity** to the data. Introducing non-linearity helps to identify the underlying patterns which are complex. It is also used to scale the value to a particular interval. For example, the sigmoid activation function scales the value between 0 and 1.

Logistic or Sigmoid function

- Logistic/ Sigmoid function scales the values between 0 and 1.
- It is used in the output layer for Binary classification.

- It may cause a **vanishing gradient** problem during backpropagation and slows the training time.

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

Sigmoid function

Tanh function

- Tanh is the short form for **Hyperbolic Tangent**. Tanh function scales the values between -1 and 1.

$$f(x) = tanh(x) = \frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$$

Hyperbolic Tangent function

ReLU function

- **ReLU (Rectified Linear Unit)** outputs the same number if x>0 and outputs 0 if x<0.
- It prevents the **vanishing gradient** problem but introduces an **exploding gradient problem** during backpropagation. The exploding gradient problem can be prevented by capping gradients.

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

ReLU function

Leaky ReLU function

- Leaky ReLU is very much similar to ReLU but when x<0 it returns (0.01 * x) instead of 0.

- If the data is normalized using Z-Score it may contain negative values and ReLU would fail to consider it but leaky ReLU overcomes this problem.

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Leaky ReLU function

Backpropagation

- Backpropagation is done to find the **optimal value for parameters** for the model by iteratively updating parameters by partially differentiating **gradients of the loss function** with respect to the **parameters**.
- An optimization function is applied to perform backpropagation. The objective of an optimization function is to find the optimal value for parameters.

The optimization functions available are,

- Gradient Descent
- Adam optimizer
- Gradient Descent with momentum
- RMS Prop (Root Mean Square Prop)

The **Chain rule of Calculus** plays an important role in backpropagation. The formula below denotes partial differentiation of Loss (L) with respect to Weights/ parameters (w).

A small change in weights ‘w’ influences the change in the value ‘z’ ($\partial z/\partial w$). A small change in the value ‘z’ influences the change in the activation ‘a’ ($\partial a/\partial z$). A small change in the activation ‘a’ influences the change in the Loss function ‘L’ ($\partial L/\partial a$).

$$\frac{\partial L}{\partial w} = \frac{\partial z}{\partial w} \times \frac{\partial a}{\partial z} \times \frac{\partial L}{\partial a}$$

Chain rule

- L - Loss function
  - Example: Cross entropy loss L = -y log($\hat{y}$) + (1 - y) log(1 - $\hat{y}$)
- w - weights
- z - Linear regression
  - Example: z = $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$
- a - activation function used.
  - Example: a = $\sigma(z) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n)$

Description of the values in the Chain rule

Terminologies:

Metrics

- A metric is used to gauge the performance of the model.
- Metric functions are similar to cost functions, except that the results from evaluating a metric are not used

when training the model. Note that you may use any cost function as a metric.

- We have used Mean Squared Logarithmic Error as a metric and cost function.

$$MSLE = \frac{1}{n}\sum_{i=1}^{n}((\log y_i + 1) - (\log \hat{y} + 1))^2$$

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}((\log y_i + 1) - (\log \hat{y} + 1))^2}$$

Mean Squared Logarithmic Error (MSLE) and Root Mean Squared Logarithmic Error(RMSLE)

Epoch

- A single pass through the training data is called an epoch. The training data is fed to the model in mini-batches and when all the mini-batches of the training data are fed to the model that constitutes an epoch.

Hyperparameters

Hyperparameters are the **tunable parameters** that are not produced by a model which means the users must provide a value for these parameters. The values of hyperparameters that we provide affect the training process so hyperparameter optimization comes to the rescue.

The Hyperparameters used in this ANN model are,

- Number of layers
- Number of units/ neurons in a layer
- Activation function
- Initialization of weights
- Loss function
- Metric
- Optimizer
- Number of epochs

Coding ANN in Tensorflow

Load the preprocessed data

The data you feed to the ANN must be preprocessed thoroughly to yield reliable results. The training data has been preprocessed already. The preprocessing steps involved are,

- MICE Imputation
- Log transformation
- Square root transformation
- Ordinal Encoding
- Target Encoding
- Z-Score Normalization

For the detailed implementation of the above-mentioned steps refer my notebook on data preprocessing

Notebook Link

Neural Architecture

- The ANN model that we are going to use, consists of seven layers including one input layer, one output layer, and five hidden layers.
- The first layer (input layer) consists of 128 units/ neurons with the ReLU activation function.
- The second, third, and fourth layers consist of 256 hidden units/ neurons with the ReLU activation function.
- The fifth and sixth layer consists of 384 hidden units with ReLU activation function.
- The last layer (output layer) consists of one single neuron which outputs an array with the shape (1, N) where N is the number of features.

Find this post in my Kaggle notebook:

https://www.kaggle.com/srivignesh/introduction-to-ann-in-tensorflow

References:

[1] Andrew Ng, Deep Learning Specialization.

Connect with me on LinkedIn, Twitter!

Happy Deep Learning!