Introduction to Artificial Neural Network Python

December 5, 2024 by akshay Tondak (https://k21academy.com/author/akshay-tondak/)

In this blog, we are going to cover everything about *Artificial Neural Network* and *how to build an Artificial Neural Network Python.*

Artificial Neural Networks are at the very core of Deep Learning. They are powerful, versatile, and scalable, making them ideal to tackle large and highly complex Machine Learning tasks powering speech recognition services (e.g., Apple's Siri), classifying billions of images (e.g., Google Images), recommending the best videos to watch (e.g., YouTube).

What Are Artificial Neuron?

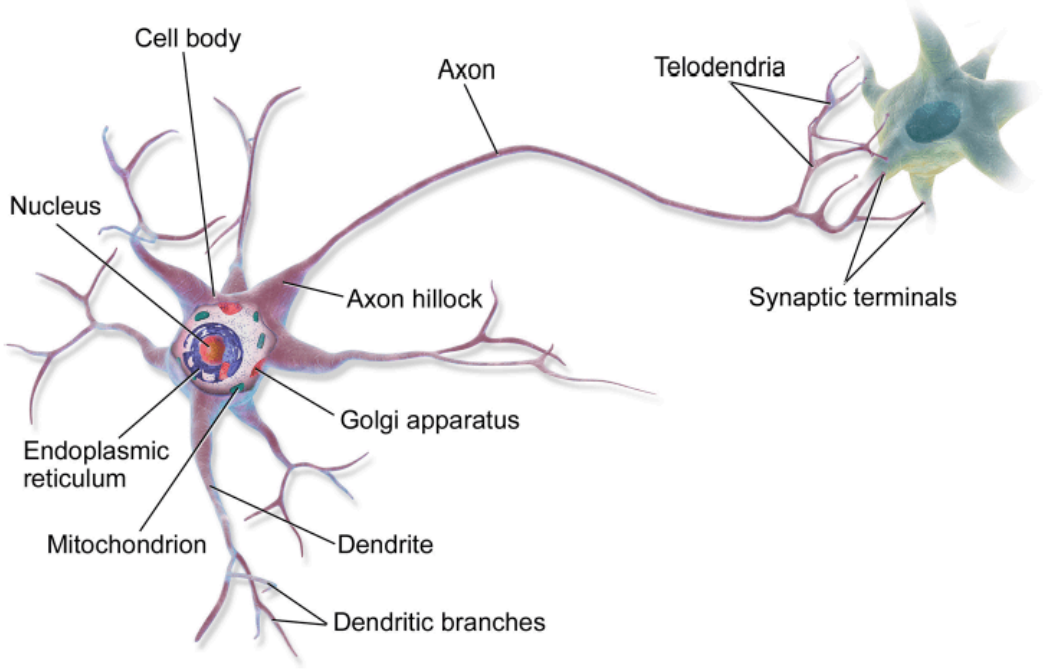
- Artificial Neurons were first popularized in 1943 by the neurophysiologist **Warren McCulloch** and the mathematician **Walter Pitts**.
- Pitts and McCulloch presented a simplified computational model of how biological neurons might work together in animal brains to perform complex computations using propositional logic.
- At that time due to a lack of resources, the progress of ANN's technique was slow, and by the 1990s other machine learning techniques offers better results.
- Due to the huge quantity of data available to train neural networks, a tremendous increase in computing power, ANNs seems to have entered a virtuous circle of funding and progress.
- Before we deep dive into artificial neurons, let's take a quick look at a biological neuron.

What Are Biological Neurons?

- It's composed of a cell body containing the **nucleus** and most of the cell's complex components, one very long extension called the **axon**. plus many branching extensions called

branches are minuscule structures called **synaptic terminals.**

- Biological neurons generate short electrical which travel along the axons and make the synapses discharge chemical signals called **neurotransmitters.**
- Each neuron commonly connected to thousands of other neurons. Highly complex computations can be performed by a network of fairly simple neurons.



How Does Artificial Neural Network Work?

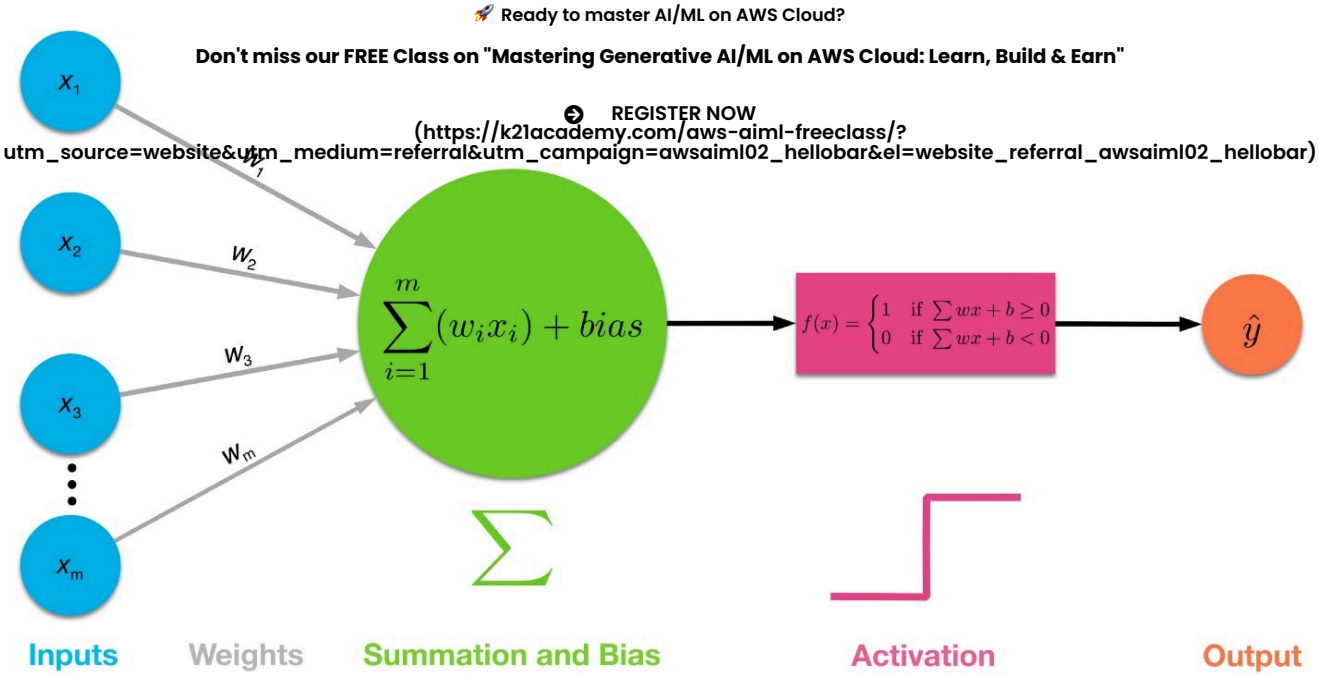
- To understand Artificial neural networks, we need to understand the most basic unit of an Artificial Neural Network, i.e. a **Perceptron**.
- The artificial neuron also has inputs and outputs so we can attempt to mimic the biological neuron.

What Is A Perceptron?

- The Perceptron is one of the simplest ANN architectures, invented in 1957 by **Frank Rosenblatt**.
- The inputs and output are numbers (instead of binary on/off values), and each input connection is associated with a weight.

- The inputs of the Perceptron are fed to special passthrough neurons called input neurons: they output whatever input they are fed.
- It has 4 important components:
    1. Inputs

4. Activation or transformation Function



The basic logic behind a Perceptron is as follows:

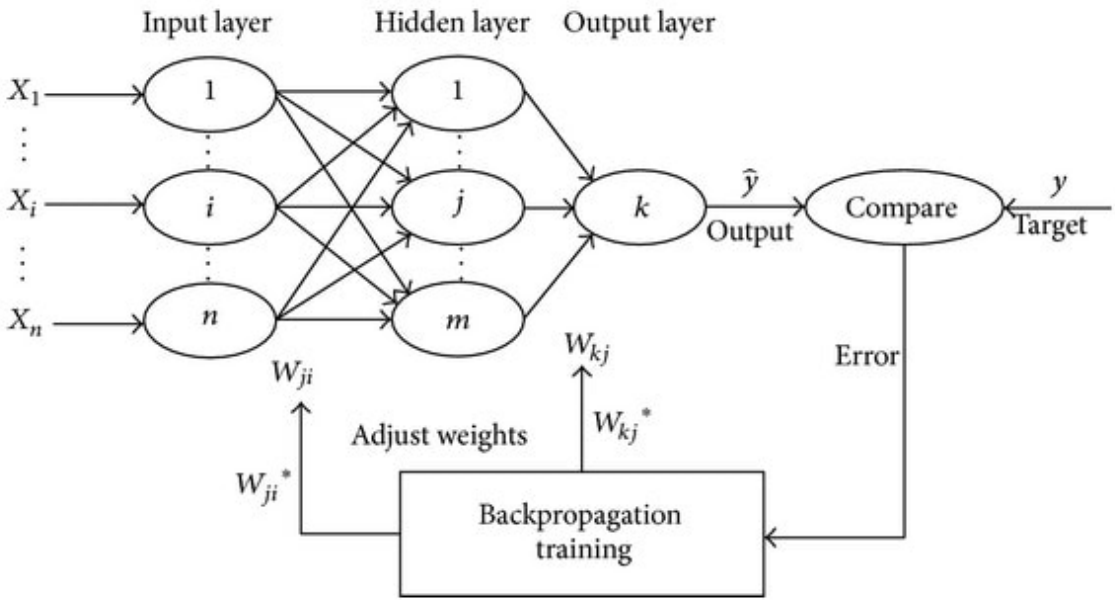
- **X** represents the matrix of input features. It has one row per instance and one column per feature.
- The weight matrix **W** contains all the connection weights except for the ones from the bias neuron.
- The bias vector **b** contains all the connection weights between the bias neuron and the artificial neurons. It has one bias term per artificial neuron.
- The function **ϕ** is called the **activation function.**

How is Perceptron Trained?

- The Perceptron is supplied one training instance at a time, and for each instance, it makes its predictions.
- For every output neuron that produced a wrong prediction, it adjusts the connection weights from the inputs that would have contributed to the correct prediction.

The Multilayer Perceptron and Backpropagation

- An MLP is composed of one (passthrough) input layer, one or more layers of threshold logic units called hidden layers, and one final layer of threshold logic unit called the output layer.
- The backpropagation algorithm finds out how each connection weight and each bias term should be tweaked in order to reduce the error.



Artificial Neural Network Python

- Deep Learning with first the classic artificial neural network, meaning the fully connected neural network with only fully connected layers, with no convolutional layers or other types of layers.
- Here we will just have an input vector containing different features and we will predict an outcome that will be a binary variable because you have to know that actually artificial neural networks can be both used for regression or classification. And here we're going to do it for classification.

Implementation First Artificial Brain

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
1	15634602	Hargrave	619	France	Female	42	2	0	1	1	1	101348.88	1
2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
3	15619304	Onio	502	France	Female	42	8	159660.8	3	1	0	113931.57	1
4	15701354	Boni	699	France	Female	39	1	0	2	0	0	93826.63	0
5	15737888	Mitchell	850	Spain	Female	43	2	125510.8	1	1	1	79084.1	0
6	15574012	Chu	645	Spain	Male	44	8	113755.8	2	1	0	149756.71	1
7	15592531	Bartlett	822	France	Male	50	7	0	2	1	1	10062.8	0
8	15656148	Obinna	376	Germany	Female	29	4	115046.7	4	1	0	119346.88	1
9	15792365	He	501	France	Male	44	4	142051.1	2	0	1	74940.5	0
10	15592389	H?	684	France	Male	27	2	134603.9	1	1	1	71725.73	0
11	15767821	Bearce	528	France	Male	31	6	102016.7	2	0	0	80181.12	0
12	15737173	Andrews	497	Spain	Male	24	3	0	2	1	0	76390.01	0
13	15632264	Kay	476	France	Female	34	10	0	2	1	0	26260.98	0
14	15691483	Chin	549	France	Female	25	5	0	2	0	0	190857.79	0
15	15600882	Scott	635	Spain	Female	35	7	0	2	1	1	65951.65	0
16	15643966	Goforth	616	Germany	Male	45	3	143129.4	2	0	1	64327.26	0
17	15737452	Romeo	653	Germany	Male	58	1	132602.9	1	1	0	5097.67	1
18	15788218	Henderson	549	Spain	Female	24	9	0	2	1	1	14406.41	0
19	15661507	Muldrow	587	Spain	Male	45	6	0	1	0	0	158684.81	0

Data Preprocessing

1) The first thing we're gonna do here is to import the libraries.

Importing the libraries

2) We import the dataset. For ‘x ‘we will specify index three so that we can take all the columns starting from the column of index three up to the one before last and taking all the rows, all the values of the dataset. For ‘y’ will just take the last column of this dataset, which is exactly what we want for our dependent.

```
[4]  print(X)

     [[619 'France' 'Female' ... 1 1 101348.88]
      [608 'Spain' 'Female' ... 0 1 112542.58]
      [502 'France' 'Female' ... 1 0 113931.57]
      ...
      [709 'France' 'Female' ... 0 1 42085.58]
      [772 'Germany' 'Male' ... 1 0 92888.52]
      [792 'France' 'Female' ... 1 0 38190.78]]

[5]  print(y)

     [1 0 1 ... 1 1 0]
```


3) In our dataset there are two categorical variables. This first one, giving the country of residence of the customers. And the second one giving the gender of the customers. So, we'll have to do some encoding work here to encode these categorical data in either simple label.

Label Encoding the "Gender" column

```
[6]  from sklearn.preprocessing import LabelEncoder
     le = LabelEncoder()
     X[:, 2] = le.fit_transform(X[:, 2])
```

```
 [608 'Spain' 0 ... 0 1 112542.58]
 [502 'France' 0 ... 1 0 113931.57]
 ...
 [709 'France' 0 ... 0 1 42085.58]
 [772 'Germany' 1 ... 1 0 92888.52]
 [792 'France' 0 ... 1 0 38190.78]]
```


One Hot Encoding the "Geography" column

```
[8] from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import OneHotEncoder
    ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [1])], remainder='passthrough')
    X = np.array(ct.fit_transform(X))
```

```
[9] print(X)

[[1.0 0.0 0.0 ... 1 1 101348.88]
 [0.0 0.0 1.0 ... 0 1 112542.58]
 [1.0 0.0 0.0 ... 1 0 113931.57]
 ...
 [1.0 0.0 0.0 ... 0 1 42085.58]
 [0.0 1.0 0.0 ... 1 0 92888.52]]
```

4) Now, we split the dataset into the training set and test it. Feature scaling is absolutely compulsory for deep learning. Whenever you build an artificial neural network, you have to apply feature scaling. That's absolutely fundamental.

Splitting the dataset into the Training set and Test set

```
[10] from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Feature Scaling

```
[11] from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train = sc.fit_transform(X_train)
     X_test = sc.transform(X_test)
```

Note: *Data preprocessing face counts for 70 percent of the work of a* ***data scientist.***

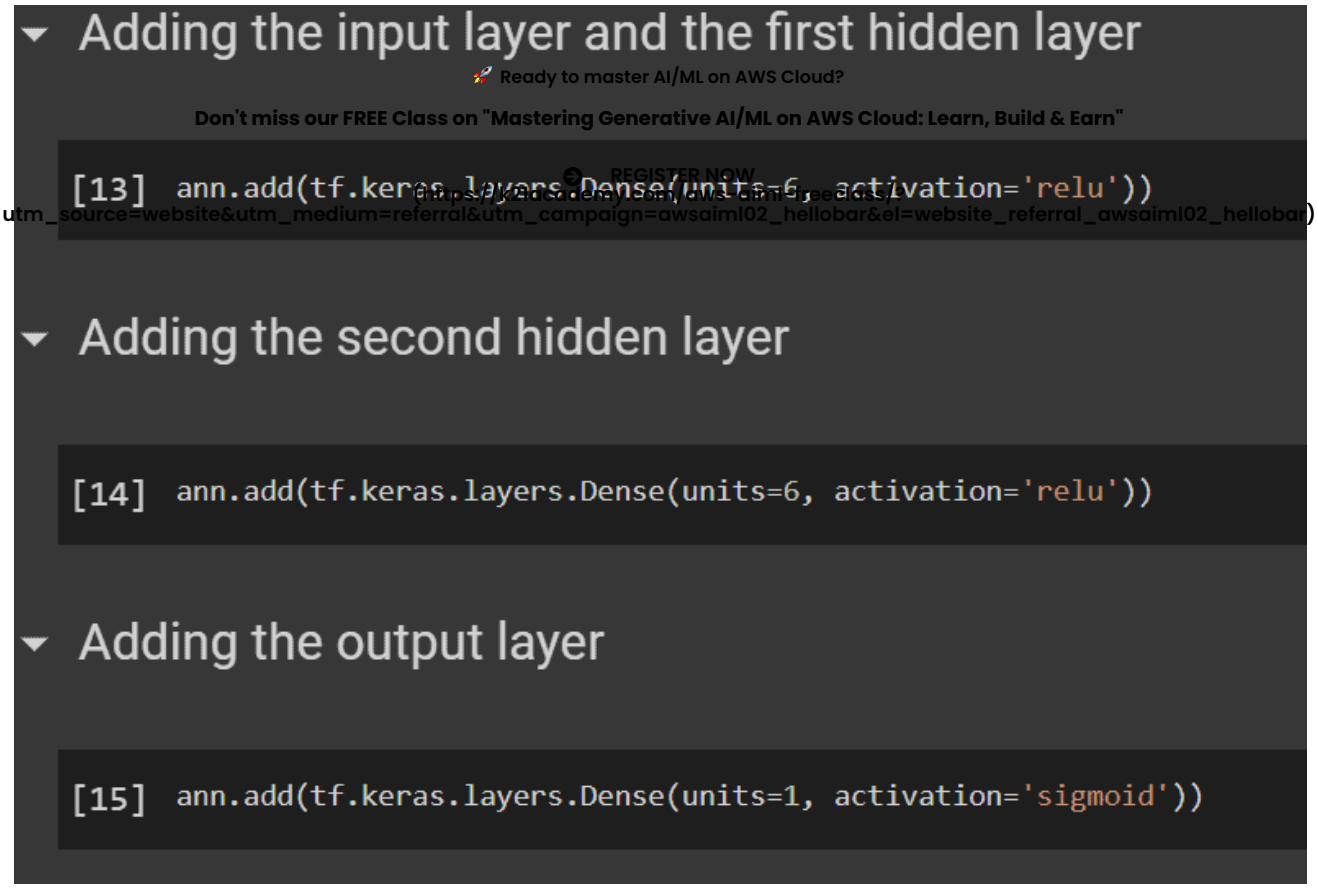
What are the pre-requisites and necessary libraries for implementing an ANN in Python?

To implement an Artificial Neural Network (ANN) in Python, you need to install essential libraries like

programming is also recommended.

Building the ANN

- We will create our new variable, and which will be nothing else than the artificial neural network we're going to build.
- We will create that variable as an object of the ***sequential class***. Tensor Flow, which has a short T.F. from which we're going to call a ***Keras Library***. And from which we're going to call the ***models*** module.
- The next step, which is to add the input layer and the first **hidden layers**. we're gonna call one of the methods of the sequential class and that method is ***'add'***. A fully-connected neural network must be the ***rectifier activation function***.
- Output layer to be fully connected to that second hidden layer and therefore we're using again here the dense class but two parameters have to be changed. The value of that unit parameter here that we have to replace right now is actually one and activation function but a ***sigmoid activation function***.



Training the ANN

- Compiling the ANN, we first need to start from our ann object, and then from this object, we're going to call a new method, the method to compile an artificial neural network is simply the ***compile*** method.
- We have to enter three parameters. i.e. an ***optimizer, loss function, and matrix***.
- Training the ANN on the Training set, the method to train whatever machinery model is the fit method, which will take always the same parameters.
- The first one is ***X train*** for, you know, the matrix of features of the training set, then ***Y train*** for the dependent variable vector of the training set. And then first ***batch size*** and final parameter is a number of ***epochs***.

Making the Predictions and Evaluating the Model

- we need to call the predict method, the predicted probability is larger than 0.5 so that if Y pred is between 0 and 0.5, then this new Y pred will become 0 because this won't be true.
- And if Y pred is larger than 0.5, then this will be true. we'll get therefore the final predicted binary outcomes one.
- We get an accuracy of over 86 percent. it means that out of 100 customers 86 were predicted correctly to either stay in or leave the bank.

How can you save an artificial neural network model in Python, and what file format is used for this purpose?

To save an artificial neural network model in Python, you can use libraries like Keras or TensorFlow. The model is typically saved in the HDF5 format with the h5 extension, which stores the model architecture, weights, and optimizer states. Use model.save (model.h5) to save it.

What are some frequently asked questions about neural networks in Python?

In Python, some frequently asked questions about neural networks include: How do I implement a neural network in Python? What libraries should I use for neural networks in Python? How can I optimize the performance of a neural network? How do I avoid overfitting in my model? These questions focus on practical implementation and optimization strategies.

Neural Network Python Libraries

Neural networks in Python are powered by several powerful libraries that make building and training models easier. Popular libraries include **TensorFlow** and **Keras**, which provide high-level APIs for building complex neural networks with ease. **PyTorch** offers dynamic computation graphs, ideal for research and experimentation. **Theano** is another deep learning library, known for its efficient computation of large-scale neural networks. **Scikit-learn** complements these libraries for simpler models and preprocessing tasks. Together, these libraries streamline the process of creating artificial neural networks for applications like image recognition, NLP, and more.

A **Neural Network** is a model inspired by the human brain, designed to recognize patterns and make predictions. In **Python**, **TensorFlow** simplifies building and training these networks for various tasks like image recognition, NLP, and more. With TensorFlow, you can define layers, activation functions, and optimize learning through backpropagation. It supports both **Deep Learning** models (e.g., CNNs, RNNs) and simpler architectures, providing flexibility for beginners and experts alike. Perfect for exploring **AI** and solving complex problems with data.

Artificial Neural Network in Python

Artificial Neural Networks (ANNs) are a class of machine learning algorithms inspired by the human brain, capable of learning from data patterns. In Python, ANNs can be implemented using libraries like TensorFlow, Keras, or PyTorch. These networks consist of layers of neurons that process input data through weights and activation functions. By training the network on labeled data, it learns to predict outputs for new, unseen inputs. ANNs are widely used in image recognition, natural language processing, and speech analysis, driving innovations in AI and automation.

Related References

- **Visit our YouTube channel "K21Academy" (https://www.youtube.com/channel/UCRgn-SAQZa4l41zLH03LoOQ?sub_confirmation=1)**
- **Join Our Generative AI Whatsapp Community (https://chat.whatsapp.com/ID6SCsi6jPnKUiJpmYgq5B)**
- **Decision Tree Algorithm Introduction** (https://k21academy.com/datascience/decision-tree-algorithm/)
- **Natural Language Processing with Python** (https://k21academy.com/datascience/machine-learning/natural-language-processing/)
- **An Introduction to Reinforcement Learning** (https://k21academy.com/datascience/machine-learning/reinforcement-learning/)
- **Data Science And Machine Learning: Hands-On Labs With Python** (https://k21academy.com/datascience/hands-on-data-science-and-machine-learning-labs-with-python/)

Next Task: Enhance Your Azure AI/ML Skills

Ready to elevate your **Azure AI/ML expertise**? Join our **free class** and gain **hands-on experience** with **expert guidance**.