Implementation of Deep Learning Algorithm with Perceptron using TenzorFlow Library

Arshiya Begum, Farheen Fatima and Asfia Sabahath

***Abstract*—In recent years, Deep Learning, Machine Learning, and Artificial Intelligence are highly focused concepts of data science. Deep learning has achieved success in the field of Computer Vision, Speech and Audio Processing, and Natural Language Processing. It has the strong learning ability that can improve utilization of datasets for the feature extraction compared to traditional Machine Learning Algorithm. Perceptron is the essential building block for creating a deep Neural Network. The perceptron model is the more general computational model. It analyzes the unsupervised data, making it a valuable tool for data analytics. A key task of this paper is to develop and analyze learning algorithm. It begins with deep learning with perceptron and how to apply it using TensorFlow to solve various issues. The main part of this paper is to make perceptron learning algorithm well behaved with non-separable training datasets. This type of algorithm is suitable for Machine Learning, Deep Learning, Pattern Recognition, and Connectionist Expert System.**

***Index Terms*—Deep Learning, Machine Learning, Perceptron Learning algorithm, TensorFlow.**

I. Introduction

IN recent year's advancement in technology, there has been a lot of focus on adaptive learning which is more fuelled by machine learning and deep learning. It is one of the effective methods to predict data analytics with a large amount of sample data. It also helps in recognizing hidden pattern or features through historical learning and trends in data. Models are created based on the past result gathered from training data, It studies the past observation results to make precise predictions. Deep learning technique could likewise be uutilized to upgrade the online learning environment for the educators to more readily assess the learning process [1]. Conventional machine-learning techniques were restricted in their abilities, for decades Machine learning algorithm is trained using training data sets to create a model. When the new input data is introduced to the Machine Learning Algorithm (MLA), it makes predictions based on the applied model.

Ms.Arshiya Begum and Asfia Sabahath are with Computer Science Department of King Khalid University, Abha, KSA.(e-mail: arshiyabegum@ymail.com, asfia.saba@gmail.com)

Ms.Farheen Fatima is a Data Analyst in Accenture Private Ltd., Hyderabad, India (e-mail: fatima.farheen711@gmail.com).

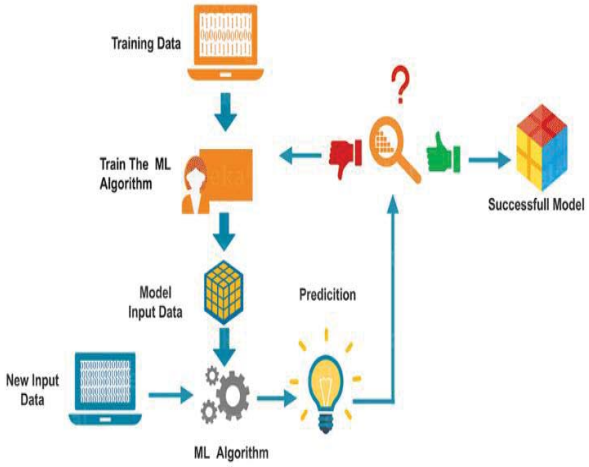


Fig. 1. Machine Learning Model

The prediction is evaluated for accuracy if accuracy is acceptable, the MLA is displayed otherwise MLA is trained again with an augmented training data sets. An ML incorporates the following four steps:

1. Feature extraction is the basis for prediction
2. Select the suitable machine learning algorithm
3. Train and evaluate model performance
4. Use the trained model to predict the obscure data

The steps of Deep Learning(DL) is similar to Machine Learning(ML) as shown in Fig. 1, Unlike the machine learning method, its feature extraction is automated rather than manual[2]. DL is a subset of ML where similar MLA are used to train deep neural networks to achieve better accuracy while predicting the required result. DL is the only method to overcome the challenges of feature extraction. The three different types of ML/DL methods are:

A. Supervised learning

The most common form of ML/DL is supervised learning. Presume that, we want to build a model that classifies the object based on their pattern. At first, we collect large data set similar to jigsaw puzzle where the one large object is divided into number of small images and labeled each with its category. Once the output is produced it is then exposed to the machine to obtained desired output in the form of vector score [3].

978-1-5386-7595-3/19/$31.00 ©2019 IEEE

The error between the output and the desired pattern is calculated using its objective function and if the attained result is not accurate the machine modifies the internal adjustable parameter to reduce the error, these parameters are called as weights [4].The supervised learning algorithm analyses the training data set and use the predicted result to map the new instances [5-10].

B. Unsupervised Learning

A Self-organizing neural network is learned using an unsupervised learning algorithm to identify the hidden pattern of the unlabeled data [6]. Here the sample data is unlabeled therefore the accuracy of data also cannot be predicted. Unsupervised learning organizes the data without any errors in the desired output pattern [2].

C. Semi-Supervised Learning

It uses a large dataset of unlabeled data and can reduce the efforts of labeling data while achieving high accuracy [2].

The rest of this paper is organized as below, Section II about Neural Network and Section III about how to implement Neural network in perceptron Learning.In section IV concludes the paper.

II. Neural Network

The Neural Network is divided into two main broad streams as shown in Fig. 2. Linear Separable problem: It shows in Fig. 2(a) the datasets are classified into two basic categories or sets using a single line. Non-Linear Separable problem: It shows in Fig. 2(b) It is the datasets contains multiple categories or sets and require a non-linear line to separate them in respective sets.

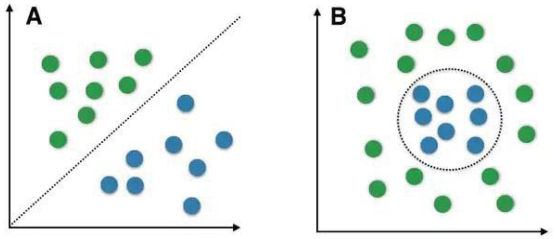


Fig. 2. (a).Linear and (b).Non-Linear Separable Problem

A. Perceptron Learning Algorithm

A Perceptron is a method for learning linear threshold device classifiers [5]. It computes the weighted sum of all the coordinates of the vector pattern .The produced output is +1 or -1 which lies below or above the threshold if larger or smaller than the threshold [6]. In the present study, the perceptron divides the input data sets into two sets. Set A lies under the decision line when Inputs are having Output as zero though Set B lies over the decision line when Inputs are having Output as one.Function of weights, Input, and Bias are mathematically represented as.

$$f(x) = wx + b \tag{1}$$

Where w =weight vector, x =Input Vector, b =Bias and $f(x)$ = Transfer functions.

Each input received from the perceptron has been weighted dependent on the measure of its contribution for obtaining final output in eq. (1) Bias enables us to move the decision line with the goal that can best separate the input into two classes.

B. Implementation of AND Gate in Perceptron Learning Algorithm

Below we have implemented the following code using Phython 3.7 using Anaconda navigator, where we create variable t_in and t_out vectors to store input, output, and bias.

```
In [ ]: #importing libraries
        import tensorflow as tf
        #Input
        t_in = [[1., 1.,1], [1., 0,1], [0, 1.,1], [0, 0,1]]
        #output
        t_out = [[1.], [0], [0], [0]]
```

For three inputs (Input 1, Input 2, Bias) three weight values are defined for each input. For these inputs tensor variable of 3*1 vectors for weight will be initialized with random values. In TensorFlow, variables update the learning procedure by changing their weights.

```
#weight variable   is initialized using random_normal()
w = tf.Variable(tf.random_normal([3, 1], seed=12))

#placeholder for input ( ) and output ( )
a = tf.placeholder(tf.float32,[None,3])
b = tf.placeholder(tf.float32,[None,1])

#calculate output
output = tf.nn.relu(tf.matmul(a,w))

#calculating error by squared loss (ls)
ls = tf.reduce_sum(tf.square(output -b))
```

In the above code, Input and output of the AND Gate are given the placeholders as a and b respectively.The input received by perceptron is multiplied by the respective weights and summed together. The summed value is fed to activation function to obtain the final output as shown in Fig. 3.

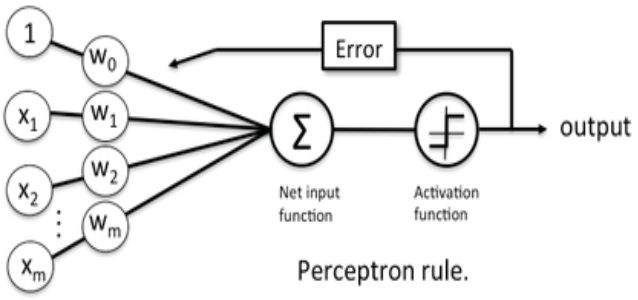


Fig. 3. AND Gate and Activation Function

```
#Minimize loss using GradientDescentOptimizer as learning rate 0.01
optimizer = tf.train.GradientDescentOptimizer(0.01)
     train = optimizer.minimize(ls)
     init = tf.global_variables_initializer()
     sess = tf.Session()
     sess.run(init)
      sess.run(train, {a:t_in,b:t_out})
      cost = sess.run(ls,feed_dict={a:t_in,b:t_out})
      print('Epoch--',i,'--loss--',cost)
```

To calculate the error value for perceptron output and the desired output, minimize TensorFlow error where it provides optimizers that gradually change each variable weight and bias in the successive iteration. Perceptron is trained to update weights and bias values in the successive iteration in order to minimize the error or loss.

C. Activation Functions

The activation function is applied to the perceptron's output. Activation function is given below to perform linear classification on the AND gate linear perceptron in eq. (2) and (3) and input dataset.

$$z = x_1 w_1 + x_2 w_2 + \ldots\ldots\ldots\ldots + x_n w_n \qquad (2)$$

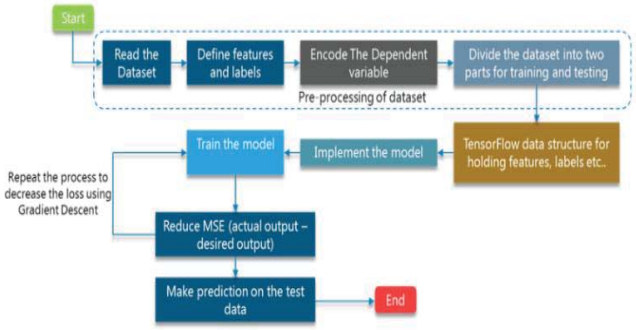
$$y = f(z) \qquad (3)$$

Where $\boldsymbol{f(z)}$=Activation Function, $\boldsymbol{y}$= Final Output

III. Use Case: Sonar Data using Single Layer Perceptron

As a testbed we have taken dataset containing data of 208 patterns obtained by bouncing sonar signals at different angles and under different conditions from a metal cylinder (naval mine) and rock. Our aim is therefore to construct a model that can predict whether the object is a naval mine or a rock. A brief walk-through of all SONAR data set linear classification steps using single Layer Perceptron is shown in Fig. 4.

In the perceptron learning algorithm input and output variable with respect to AND Gate are defined explicitly.



X1	X2	y
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 4. SONAR dataset using Single Layer Perceptron

```
#Trained model on test subset
pred_y = sess.run(b, feed_dict={a: test_x})

#calculate the correct predictions
correct_prediction = tf.equal(tf.argmax(pred_y,1), tf.argmax(test_y,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print(&amp;amp;amp;quot;Accuracy: &amp;amp;amp;quot;,sess.run(accuracy))
```

An accuracy of 83.84% is achieved during this process. A graph in Fig. 5 shows reduced cost or error in successive epochs in a graph where Cost vs Number of Epochs are plotted.

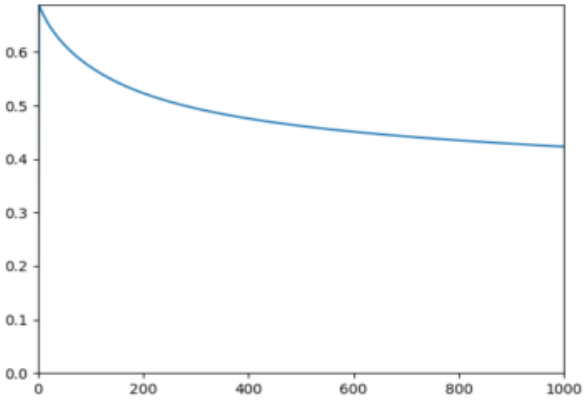


Fig. 5. Cost vs Number of Epochs

But, in use cases such as SONAR, will be provided with the raw data sets that has to read and preprocess to train the model. It is divided into two data sets during preprocess of data.

Training Subset: Used to train a model

Testing subset: Used to validate trained model

The model is trained and the cost or error is calculated using Cross Entropy Mean squared error method. The model is trained in successive epoch by calculating the cost of weight and bias to minimize the error. The trained model calculates the error or cost to attained accurate predicted output.

IV. Conclusion

In this paper, we have learnt what is perceptron and how the perceptron learning algorithm is implemented using TensorFlow library. Also, we discussed how a perceptron can be used as a linear classifier and how to implement AND Gate using perceptron. We have observed that the Perceptron learning algorithm guarantees the performance depending on a notion of "separability" of training datasets. The solution describes the convergence property and could be applied to other models too in the nonlinear separable problem.

REFERENCES

[1] O.R. Zaiane,"Web usage mining for better web-based learning environment", in:Prof. of Conference on Advanced Technology for Education, Banff, AB, June2001, pp. 60–64.

[2] Yang xin, Lingshuang Kong, Zhi Liu, (member, IEEE), Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao,Haixia Hou, and Chunhua Wang , "Machine Learning and Deep Learning Methods for Cyber security",IEEE Access,Vol. 6 page 35365 – 35381(15 May 2018).21

[3] Yann LeCun, Yoshua Bengio & Geoffrey Hinton,"Deep Learning",436 NATURE, Vol. 521, 28 May 2015.

[4] R. Sathya, Annamma Abraham, "Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification", (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No. 2, 2013.

[5] Kai-Yeung Siu, Amir Dembo, Thomas Kailath. "On the Perceptron Learning Algorithm on Data with High Precision", Journal Of Computer And System Sciences 48, 347-356 (1994).

[6] Nicol`o Cesa-Bianchi, Alex Conconi, and Claudio Gentile "A Second-Order Perceptron Algorithm",Siam J.Comput., Vol. 34, No. 3, pp. 640–668.

[7] Tang, J., Deng, C. and Huang, G.B., 2016. Extreme learning machine for multilayer perceptron. *IEEE transactions on neural networks and learning systems*, *27*(4), pp.809-821.

[8] Raiko, T., Valpola, H. and LeCun, Y., 2012, March. Deep learning made easier by linear transformations in perceptrons. In *Artificial intelligence and statistics* (pp. 924-932).

[9] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[10] Zhu, Z., Luo, P., Wang, X. and Tang, X., 2014. Multi-view perceptron: a deep model for learning face identity and view representations. In *Advances in Neural Information Processing Systems* (pp. 217-225).