Dikshita Kambri

118A2044

BE EXTC A2

EXPERIMENT 8

ARTIFICIAL NEURAL NETWORK

EXPERIMENT 8

ARTIFICIAL NEURAL NETWORK

AIM: Implement artificial neural network

APPARATUS: python

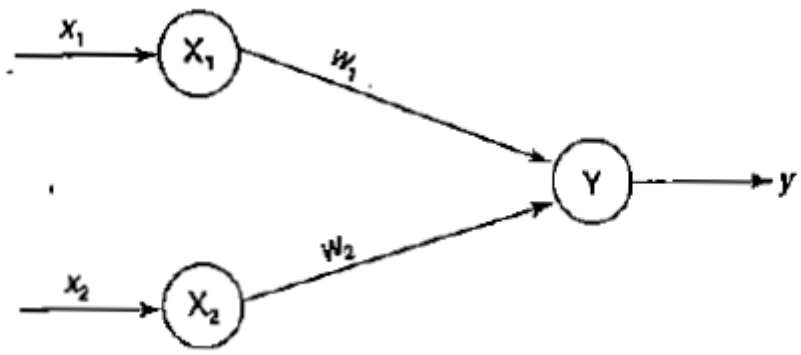
THEORY:

Artificial Neural Network (ANN) - An Introduction

Fundamental Concept

Neural networks are those information processing systems, which are constructed and implemented to model the human brain. The main objective of the neural network research is to develop a computational device for modelling the brain to perform various computational tasks at a faster rate than the traditional systems. ANN performs various tasks such as pattern matching and classification, optimization function, approximation, vector quantization and data clustering. These tasks are very difficult for traditional computers which are faster in algorithmic computational task Therefore for implementation of ANN high speed digital computers are used, which makes the simulation of neural processes feasible.

An artificial neural network (ANN) is an efficient information processing system which resembles in characteristics with a biological neural network. ANNs possess large number of highly interconnected processing elements called nodes or *units* or *neuron,* which usually operate in parallel and are configured in regular architectures. Each neuron is connected with the other by a connection link. Each connection link is associated with weights which contain information about the input signal. This information is used by the neuron net to solve a Particular problem. ANNs' collective behaviour is characterized by their ability to learn, recall and generalize training process or data similar to that of a human brain. They have the capability to model networks as found in the brain. Thus, the ANN processing elements are called *neurons* or *artificial neurons.*



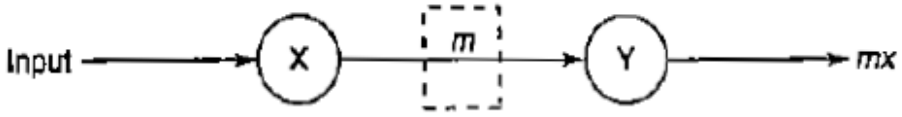
It should be noted that each neuron has an internal state of its own. This internal stare is called the *activation* or *activity* level of neuron, which is the function of the inputs the neuron receives. The activation signal of a neuron is transmitted to other neurons. A neuron can send only one signal at a time, which can be transmitted to several other neurons.
To depict the basic operation of a neural net, consider a set of neurons, say X1 and X2, transmitting signals to another neuron Y. Here X1 and X2 are input neurons, which transmit signals and Y is the output neuron, which receives signals. Input neurons X1 and *X2* are connected to the output neuron Y, over a weighted interconnection links (W, and W2) as shown in F..igure above.

For the above simple neuron net architecture, the net input has to be calculated in the following way:

Yin= +x1w1 +x2w2
where x1 and x2 are the activations of the input neurons X1 and X2, i.e., the output of input signals. The output *y* of the output neuron Y can be obtained by applying activations over the net input, i.e. the function of the net input:

Output= Function (net input calculated)

The function to be applied over the net input is called activation function. There are various activation functions, which will be discussed in the forthcoming sections. The above calculation of the net input is similar to the calculation of output of a pure linear straight line equation (y = *mx).* The neural net of a pure linear equation is as shown in Figure below.



Here, to obtain the output *y,* the slope *m* is directly multiplied with the input signal. This is a linear equation. Thus, when slope and input are linearly varied, the output *is* also linearly varied. This shows that the weight involved in ANN is equivalent to the slope of the linear straight line.

PROGRAM:

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)
fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashio
n_mnist.load_data()
```

```
train_images.shape
class_names = ['top','trouser', 'pullover', 'dress', 'coat', 'san
dal', 'shirt','sneaker','bat', 'boot']
len(train_labels)
train_labels
test_images.shape
len(test_labels)
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(True)
plt.show()
train_images = train_images / 255.0
test_images = test_images / 255.0
plt.figure()
for i in range(36):
  plt.subplot(6,6,i+1)
  plt.grid(False)
  plt.imshow(train_images[i], cmap=plt.cm.binary)
  plt.xlabel(class_names[train_labels[i]])
plt.show
model = keras.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(128, activation=tf.nn.relu),
  keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=10)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy', test_acc)
predictions = model.predict(test_images)
predictions[1]
np.argmax(predictions[1])
test_labels[1]
```

OUTPUT:

```
[18] 1875/1875 [==============================] - 3s 2ms/step - loss: 0.2654 - accuracy: 0.9026
     Epoch 8/10
     1875/1875 [==============================] - 3s 2ms/step - loss: 0.2557 - accuracy: 0.9058
     Epoch 9/10
     1875/1875 [==============================] - 3s 2ms/step - loss: 0.2457 - accuracy: 0.9086
     Epoch 10/10
     1875/1875 [==============================] - 3s 2ms/step - loss: 0.2355 - accuracy: 0.9125
     <keras.callbacks.History at 0x7f30071aa950>

[19] test_loss, test_acc = model.evaluate(test_images, test_labels)
     print('Test accuracy', test_acc)

     313/313 [==============================] - 1s 1ms/step - loss: 0.3360 - accuracy: 0.8834
     Test accuracy 0.883400022983551

[20] predictions = model.predict(test_images)

[21] predictions[1]

     array([7.6723018e-06, 1.5034504e-13, 9.9980658e-01, 2.8633853e-12,
            1.3059399e-04, 1.7528971e-13, 5.5155982e-05, 3.4881966e-19,
            7.5130603e-11, 2.0955417e-15], dtype=float32)

[22] np.argmax(predictions[1])

     2

[23] test_labels[1]

     2
```

CONCLUSION:

Successfully implemented ANN for image classification using mnis dataset with test accuracy of 0.833.