AD18511 – DEEP LEARNING LABORATORY

DATE:

EX NO: 4

SOFTMAX_REGRESSION.

AIM:

To write a program to build a softmax regression model using tensorflow.

DESCRIPTION:

- Softmax Regression or multimodal logistic regression is a generalization of logistic regression to the case where we want to handle multiple classes.
- Softmax Regression allows us to handle

yi ∈ {1,2,3,.....,k} where k is the number of classes

$$Pi= \frac{e^{wiTx}}{\sum_{i=1}^{k} e^{wiTx}}$$

where $1 \leq i < k$

- Softmax function is the extension of sigmoid function in multiclass.

PROGRAM:

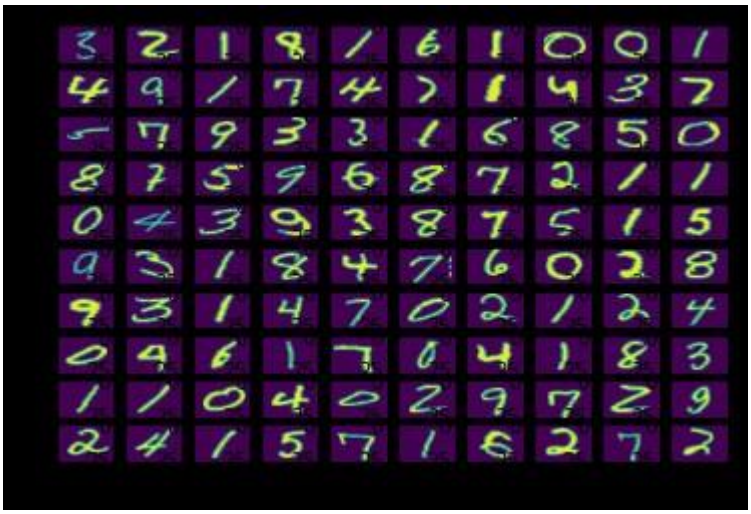
```
import tensorflow as tf
import tensorflow.compat.v1 as tf1
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

(X_train,Y_train),\
(X_val,Y_val)=tf.keras.datasets.mnist.load_data()
print("shape of features matrix:",X_train.shape)
print("shape of target matrix:",Y_train.shape)
```

OUTPUT:

```
shape of features matrix: (60000, 28, 28)
shape of target matrix: (60000,)
```

```
fig,ax=plt.subplots(10,10)
for i in range(10):
        for j in range(10):
        k=np.random.randint(0,X_train.shape[0])
        ax[i][j].imshow(X_train[k].reshape(28,28),aspect='auto')
plt.show()
```

OUTPUT:



```
num_features=784
num_labels=10
learning_rate=0.05
batch_size=128
num_steps=5001
train_dataset=X_train.reshape(-1,784)
train_labels=pd.get_dummies(Y_train).values
valid_dataset=X_val.reshape(-1,784)
valid_labels=pd.get_dummies(Y_val).values

graph=tf.Graph()
with graph.as_default():
        tf_train_dataset=tf1.placeholder(tf.float32,shape=(batch_size,num_features))
        tf_train_labels=tf1.placeholder(tf.float32,shape=(batch_size,num_labels))
        tf_valid_dataset=tf.constant(valid_dataset)
        weights=tf.Variable(tf.random.truncated_normal([num_features,num_labels]))
        biases=tf.Variable(tf.zeros([num_labels]))
        logits=tf.matmul(tf_train_dataset,weights)+biases
        loss=tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=tf_train_labels,logits=logits))
        optimizer=tf1.train.GradientDescentOptimizer(learning_rate).minimize(loss)
        train_prediction=tf.nn.softmax(logits)
        tf_valid_dataset=tf.cast(tf_valid_dataset,tf.float32)
        valid_prediction=tf.nn.softmax(tf.matmul(tf_valid_dataset,weights)+biases)
def accuracy(predictions,labels):
        correctly_predicted=np.sum(np.argmax(predictions,1)==np.argmax(labels,1))
        acc=(100.0*correctly_predicted)/predictions.shape[0]
        return acc
with tf1.Session(graph=graph) as session:
        tf1.global_variables_initializer().run()
        print("Initialized")
        for step in range(num_steps):
        offset = np.random.randint(0, train_labels.shape[0] - batch_size - 1)
        batch_data = train_dataset[offset:(offset + batch_size), :]
        batch_labels = train_labels[offset:(offset + batch_size), :]
        feed_dict = {tf_train_dataset: batch_data,tf_train_labels: batch_labels}
        _, l, predictions = session.run([optimizer, loss, train_prediction],feed_dict=feed_dict)
```

```
if (step % 500 == 0):
print("Minibatch loss at step {0}: {1}".format(step, l))
print("Minibatch accuracy: {:.1f}%".format(accuracy(predictions, batch_labels)))
print("Validation accuracy: {:.1f}%".format(accuracy(valid_prediction.eval(), valid_labels)))
```

OUTPUT:

```
Initialized
Minibatch loss at step 0: 3228.845703125
Minibatch accuracy: 14.1%
Validation accuracy: 30.5%
Minibatch loss at step 500: 470.5828857421875
Minibatch accuracy: 88.3%
Validation accuracy: 88.5%
Minibatch loss at step 1000: 810.6234741210938
Minibatch accuracy: 83.6%
Validation accuracy: 82.7%
Minibatch loss at step 1500: 217.19741821289062
Minibatch accuracy: 93.8%
Validation accuracy: 90.8%
Minibatch loss at step 2000: 400.3426818847656
Minibatch accuracy: 88.3%
Validation accuracy: 90.1%
Minibatch loss at step 2500: 587.1039428710938
Minibatch accuracy: 89.8%
Validation accuracy: 87.5%
Minibatch loss at step 3000: 793.615966796875
Minibatch accuracy: 85.9%
Validation accuracy: 88.4%
Minibatch loss at step 3500: 742.802978515625
Minibatch accuracy: 85.9%
Validation accuracy: 86.7%
Minibatch loss at step 4000: 181.66873168945312
Minibatch accuracy: 94.5%
Validation accuracy: 89.6%
Minibatch loss at step 4500: 692.54296875
Minibatch accuracy: 86.7%
Validation accuracy: 89.5%
Minibatch loss at step 5000: 200.63148498535156
Minibatch accuracy: 94.5%
Validation accuracy: 91.0%
```

RESULT:

Hence, the implementation of softmax regression model was done successfully using tensor flow.