TensorFlow

BY - ANUJ SACHIN KADU
UNDER THE GUIDANCE OF :
1)DR. KIRUTHIKA M.
2)MRS. ARCHANA S.

Flow of Contents

Definition

TensorFlow is an open source deep learning framework developed by google which allows us to build and train neural networks for various tasks like :

- Image & speech recognition
- NLP
- Financial prediction models

TensorFlow supports:

- Low-level neural network building with tensors. (e.g., with tf.Variable, GradientTape)
- High-level APIs like Keras for faster model development.

We will learn how to use both of these methods with a problem statement

Sample problem statement

Finance Use Case : Loan default prediction model

Input Features : Age , Income , Credit Score , Loan Amount , Loan Term and Past Defaults

Sample table :

	age	income	credit_score	loan_amount	loan_term	past_defaults	will_default
0	59	77761	848	841507	12	1	1
1	49	141178	651	519164	36	0	0
2	35	56368	611	753814	12	3	1
3	28	26783	358	78365	24	0	1
4	41	131530	334	515982	36	0	1

Manual Implementation - 1

Manually inputting the model weights

Weights

Features

```
W1 = tf.Variable([
    [0.05, 0.1, 0.2, 0.0, 0.1, 0.3, 0.0, 0.15],
    [0.0, 0.0, 0.1, 0.2, 0.0, 0.2, 0.1, 0.1],
    [0.2, 0.2, 0.0, 0.1, 0.3, 0.0, 0.1, 0.0],
    [0.1, 0.0, 0.0, 0.3, 0.1, 0.0, 0.2, 0.05],
    [0.05, 0.1, 0.1, 0.1, 0.05, 0.1, 0.0, 0.0],
    [0.3, 0.2, 0.1, 0.0, 0.0, 0.1, 0.3, 0.2]
, dtype=tf.float32)
```

Linear transformation

$Z = \sum XW + B$

X : DATA
W : WEIGHTS
B : BIAS VECTOR
Z: PRE ACTIVATION

Weights : Control how much each input feature controls output.
Bias Vector :

Manual Implementation - 2

But this Linear Transformation has an issue :
It is linear , can only be used for regression , not complex patterns.

Activation Function

Mathematical operation applied after each linear transformation to make it non linear.

ReLU	Most widely used in DL and has capablity of choosing which neuron to use for training
Leaky ReLU	fixes issues of ReLU where neurons output 0 in a loop
Sigmoid	It is for output and best for displaying binary classification output.
Softmax	Used in multi-class classification

Manual Implementation - 3

After Activation our model is technically ready to use. But for later purpose when we will need to train the model , we use Loss function

Loss Function

It tells the model how wrong its predictions are.

Binary CrossEntropy	Binary Classification(Comparison of predicted with actual output)
Categorical CrossEntropy	Multi Class classification(Compares probablity distribution of classes encoded in one hot encoding.)
Mean Square Error	Regression
KL Divergence	Probablistic(Compares 2 probablity distributions often in generative tasks like LLM)

Manual Implementation - 4

While training the model , along with Loss function we also need **Optimizer** and **Backpropogation**.

Backpropogation

Chain of mathematical engines which computes how each weight affect final loss.

Optimizer

Algorithm that adjusts model's weights so that loss gets smaller.

SGD	Small and simple with constant learning rate.
SGDM	memory of past gradients helping escape local minima
RMS	Adjusts learning rate per weight by using RMSE.

Height of the hill → Loss. Optimizer's job is to go to the bottom of the hill using gradients.

Gradient : Change of loss upon changing the weights.

ADAM(Adaptive Moment estimation)
Combines all three to become the best.

Using TensorFlow libraries

We saw how grilling and time consuming it is to build a small neural model on our own. Here comes in picture Keras , TensorFlow's high-level API. It simplifies deep learning development by abstracting the lower-level mechanics while still letting us customize when needed.

As we can see , most of our functions which had to be manually written are simply abstracted and presented as functions in Keras , boosting productivity and ease of use.

These modules help us build complex models revolving around our usecase in fintech

Step	Manual TensorFlow	Keras Shortcut
Define Layers/Weights	`tf.Variable` , `matmul` , `relu`	`Dense(..., activation=...)`
Activation Functions	`tf.nn.relu` , `sigmoid`	Declared inside layer definition
Forward Pass	Manually compute `Z1` , `A1` , ...	`model(X)`
Loss Calculation	Manual binary cross-entropy	`BinaryCrossentropy()` or `'binary_crossentropy'`
Backpropagation	`GradientTape` , apply gradients	Done inside `model.fit()`
Optimizer	`optimizer.apply_gradients()`	`optimizer='adam'`
Accuracy Calculation	`tf.equal` , `reduce_mean`	`metrics=['accuracy']`
Prediction on New Sample	Full forward pass manually	`model(X_new)`

Conclusion

HENCE WE HAVE LEARNT AND IMPLEMENTED TENSORFLOW FROM SCRATCH AS WELL AS USED ITS LIBRARIES TO SHOW US HOW QUICKLY WE CAN BUILD NEURAL MODELS TO USE IN VARIOUS FINANCIAL USE CASES.