

Learning Tensorflow for Developing Machine Learning Models

Sukanta Dey

Ph.D. Candidate,
Dept of CSE, IIT Guwahati,
Guwahati – 781039, India

10th August 2020

Workshop on “*Signal Processing and Machine Learning*”

Dept of ECE, Assam Don Bosco University

Guwahati – 782402, Assam, India

About Me

- BE in ETC, AEC (2014)
- M.Tech-PhD in CSE, IIT Guwahati (2014 to Present)
- Research Interest: Machine Learning, VLSI CAD, Hardware Security.
- Email: sukanta.dey@iitg.ac.in
- Email: sukantadey12@gmail.com
- Website: <https://thesukantadey.github.io/>
- Code and slide available at:
https://github.com/thesukantadey/TF_ADBU

Overview

- Introduction to TensorFlow
- Intro to ML Regression & Classification
- Intro to Neural Network
- Hands-on Codes on Google Colab

What's TensorFlow™?

- Open source software library for numerical computation using data flow graphs
- Originally developed by Google Brain Team to conduct machine learning and deep neural networks research
- General enough to be applicable in a wide variety of other domains as well

Why TensorFlow™?

● tensorflow tutorial
Search term

● torch tutorial
Search term

● theano tutorial
Search term

⋮

● Caffe tutorial
Search term

● CNTK tutorial
Search term

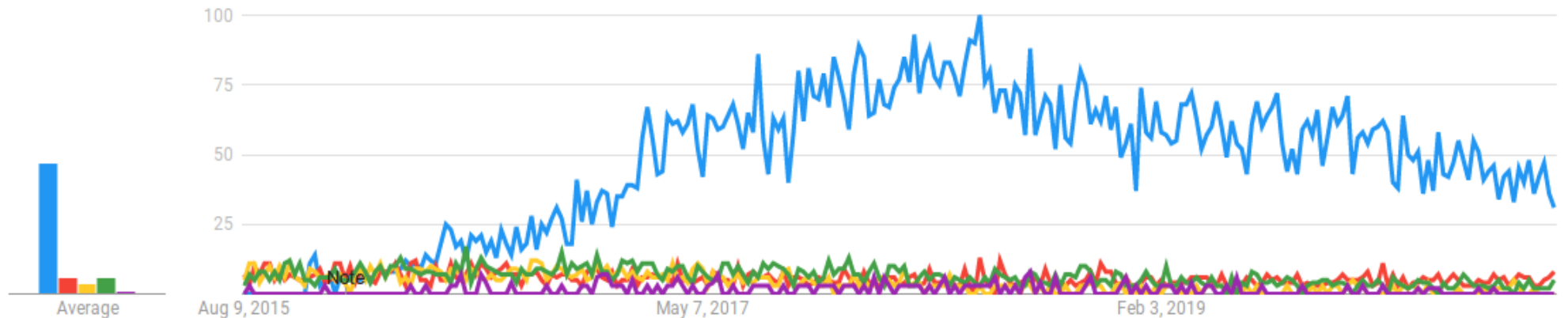
Worldwide ▼

Past 5 years ▼

All categories ▼

Web Search ▼

Interest over time ⓘ



What's TensorFlow™?

- Python API
- Portability: deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API
- Flexibility: from Raspberry Pi, Android, Windows, iOS, Linux to server farms
- Visualization (TensorBoard is da bomb)
- Checkpoints (for managing experiments)
- Auto-differentiation autodiff (no more taking derivatives by hand. Yay)
- Large community (> 10,000 commits and > 3000 TF-related repos in 1 year)
- Awesome projects already using TensorFlow

Companies using Tensorflow

- Google
- OpenAI
- DeepMind
- Snapchat
- Uber
- Airbus
- eBay
- Dropbox
- A bunch of startups

TensorFlow online tutorials

- <https://www.tensorflow.org/tutorials/>

The screenshot shows the TensorFlow website's tutorial section. The browser's address bar displays <https://www.tensorflow.org/tutorials/>. The page features a navigation bar with links for 'Install', 'Learn', 'API', 'Resources', 'Community', and 'More'. Below this, the 'TensorFlow Core' section is active, with tabs for 'Overview', 'Tutorials', 'Guide', and 'TF 1'. The left sidebar lists various tutorial categories, including 'TensorFlow tutorials', 'Quickstart for beginners', 'Quickstart for experts', 'BEGINNER', 'ML basics with Keras', 'Load and preprocess data', 'Estimator', 'ADVANCED', 'Customization', 'Distributed training', 'Images', and 'Text'. The main content area is titled 'For beginners' and explains that the tutorials are Jupyter notebooks running in Google Colab. It provides a 'Beginner quickstart' link, a 'Keras basics' link, and a 'Load data' link, each with a brief description of the content.

TensorFlow

Install Learn API Resources Community More Search English GitHub Sig

TensorFlow Core

Overview Tutorials Guide TF 1

TensorFlow tutorials

Quickstart for beginners

Quickstart for experts

BEGINNER

ML basics with Keras

Load and preprocess data

Estimator

ADVANCED

Customization

Distributed training

Images

Text

The TensorFlow tutorials are written as Jupyter notebooks and run directly in Google Colab—a hosted notebook environment that requires no setup. Click the *Run in Google Colab* button.

For beginners

The best place to start is with the user-friendly Keras sequential API. Build models by plugging together building blocks. After these tutorials, read the [Keras guide](#).

Beginner quickstart

This "Hello, World!" notebook shows the Keras Sequential API and `model.fit`.

Keras basics

This notebook collection demonstrates basic machine learning tasks using Keras.

Load data

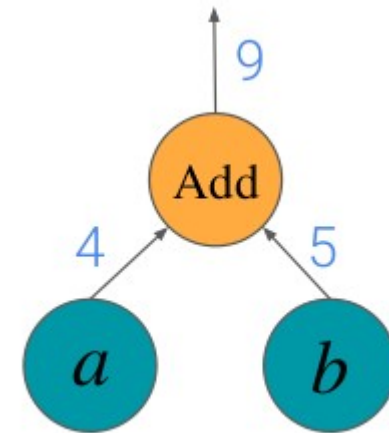
These tutorials use `tf.data` to load various data formats and build input pipelines.

Basic Code Structure

- View functions as computational graphs
- First build a computational graph, and then use a session to execute operations in the graph
- This is the basic approach, there is also a dynamic approach implemented in the recently introduced eager mode

Basic Code Structure

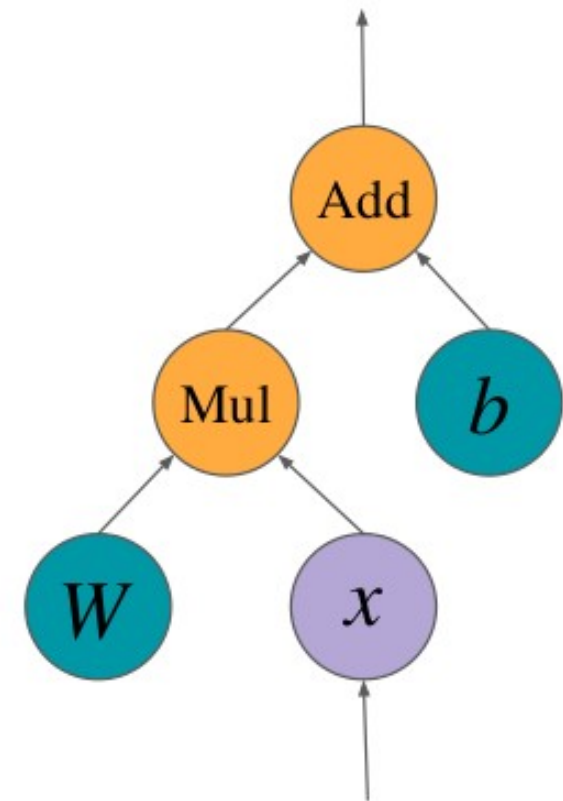
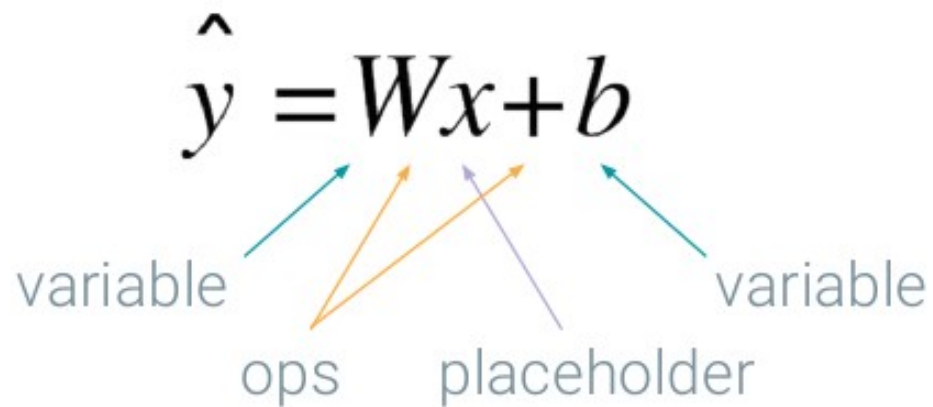
- Nodes are operators (ops), variables, and constants
- Edges are tensors
 - 0-d is a scalar
 - 1-d is a vector
 - 2-d is a matrix
- TensorFlow = Tensor + Flow = Data + Flow



Basic Code Structure

- Constants are fixed value tensors - not trainable
- Variables are tensors initialized in a session – trainable
- Placeholders are tensors of values that are unknown during the graph construction, but passed as input during a session
- Ops are functions on tensors

Basic Code Structure - Graphs



Basic Code Structure - Sessions

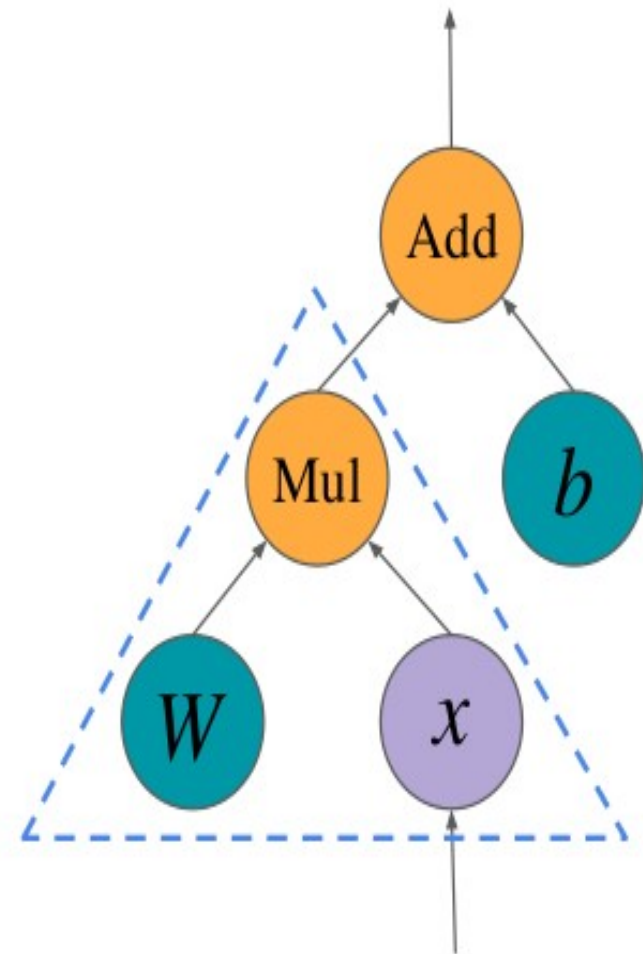
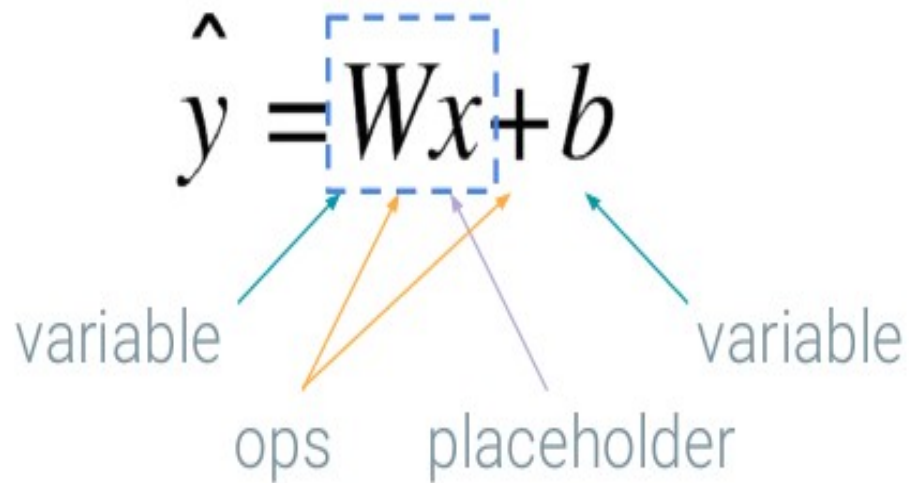
- Session is the runtime environment of a graph, where operations are executed, and tensors are evaluated

```
>>> import tensorflow as tf
>>> a = tf.constant(4)
>>> b = tf.constant(5)
>>> add_op = tf.add(a, b)
>>> print(add_op)
Tensor("Add:0", shape=(), dtype=int32)
```

```
>>> import tensorflow as tf
>>> a = tf.constant(4)
>>> b = tf.constant(5)
>>> add_op = tf.add(a, b)
>>> with tf.Session() as session:
...     print(session.run(add_op))
...
9
```

- `a.eval()` is equivalent to `session.run(a)`, but in general, “eval” is limited to executions of a single op and ops that returns a value
- Upon op execution, only the subgraph required for calculating its value is evaluated

Basic Code Structure - Sessions



TensorFlow 1.x vs TensorFlow 2.x

- Only Static Computation graph.
- Heavyweight build-then-run was overkill for simple applications
- Low level API
- `tf.Session` for hard separation from Python
- Both dynamic and static supported
- Eager execution
- High level API Keras Integrated
- No sessions, just functions. `tf.function` decorator for advanced uses

Machine Learning

- **Machine Learning:** Intelligence demonstrated by machines using its previous experiences.
- Supervised Learning: Labelled data.
- Unsupervised Learning: Unlabelled data.

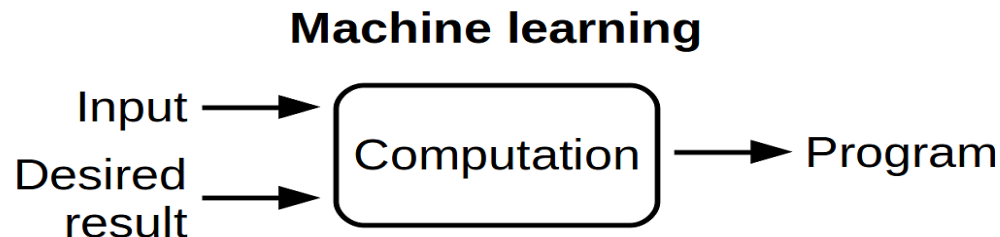
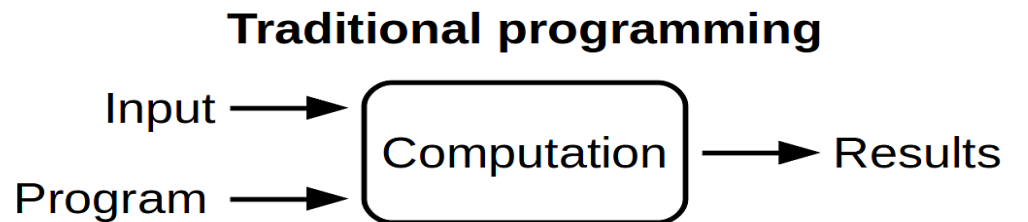


Figure: Difference between Traditional programming and Machine Learning

Supervised Learning

- All data are labeled.
- For a set of input features there is output feature.
- Types of Supervised Learning tasks:
 - Regression (Linear Regression demo)
 - Classification (Logistic Regression demo)

Linear Regression

- In regression, the output variable is a continuous variable.
- Multiple input features or variables.
- One output feature or variable.
- For eg: Housing Price Prediction Problem.
- Input feature: House area, location, number of rooms etc.
- Output feature: Price of house.

Classification

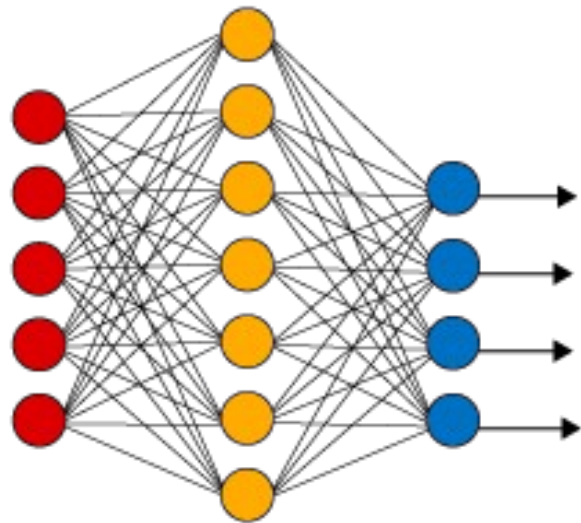
- In logistic regression, the output variable is discrete “Yes” or “No”.
- In multi-class regression, there can be multiple classes of output.

Traditional Machine Learning Models

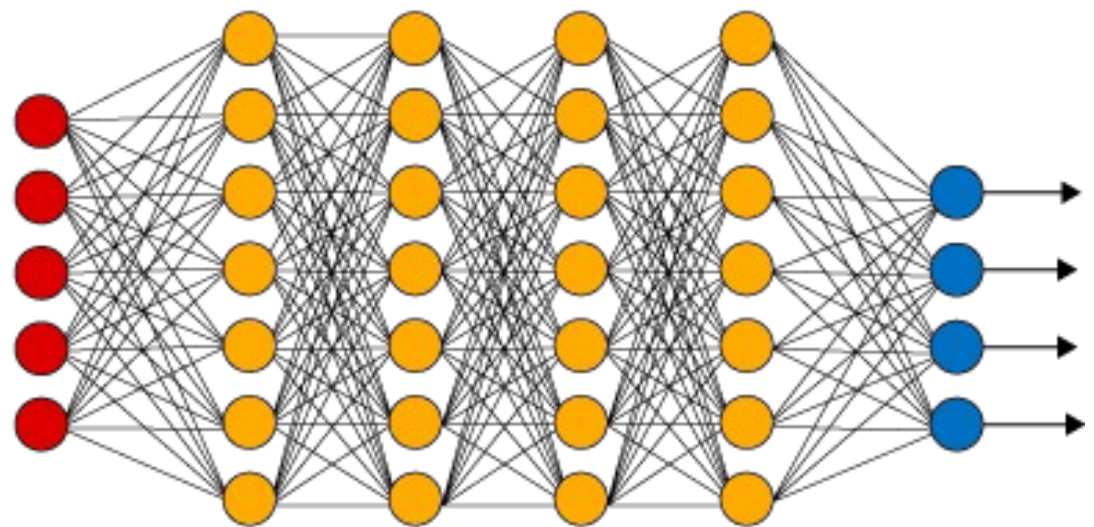
- SVM
- Decision Trees
- Random Forest
- Gaussian Process Regression
- For these models you can use sci-kit learn package
- Tensorflow is famous for creating deep learning models

Neural Networks

Simple Neural Network



Deep Learning Neural Network



● Input Layer

● Hidden Layer

● Output Layer

Constructing Neural Networks in TF

```
import tensorflow as tf
```

```
from tensorflow.keras.layers import Input, Dense, Activation, Dropout  
from tensorflow.keras.models import Model
```


Constructing Neural Networks in TF

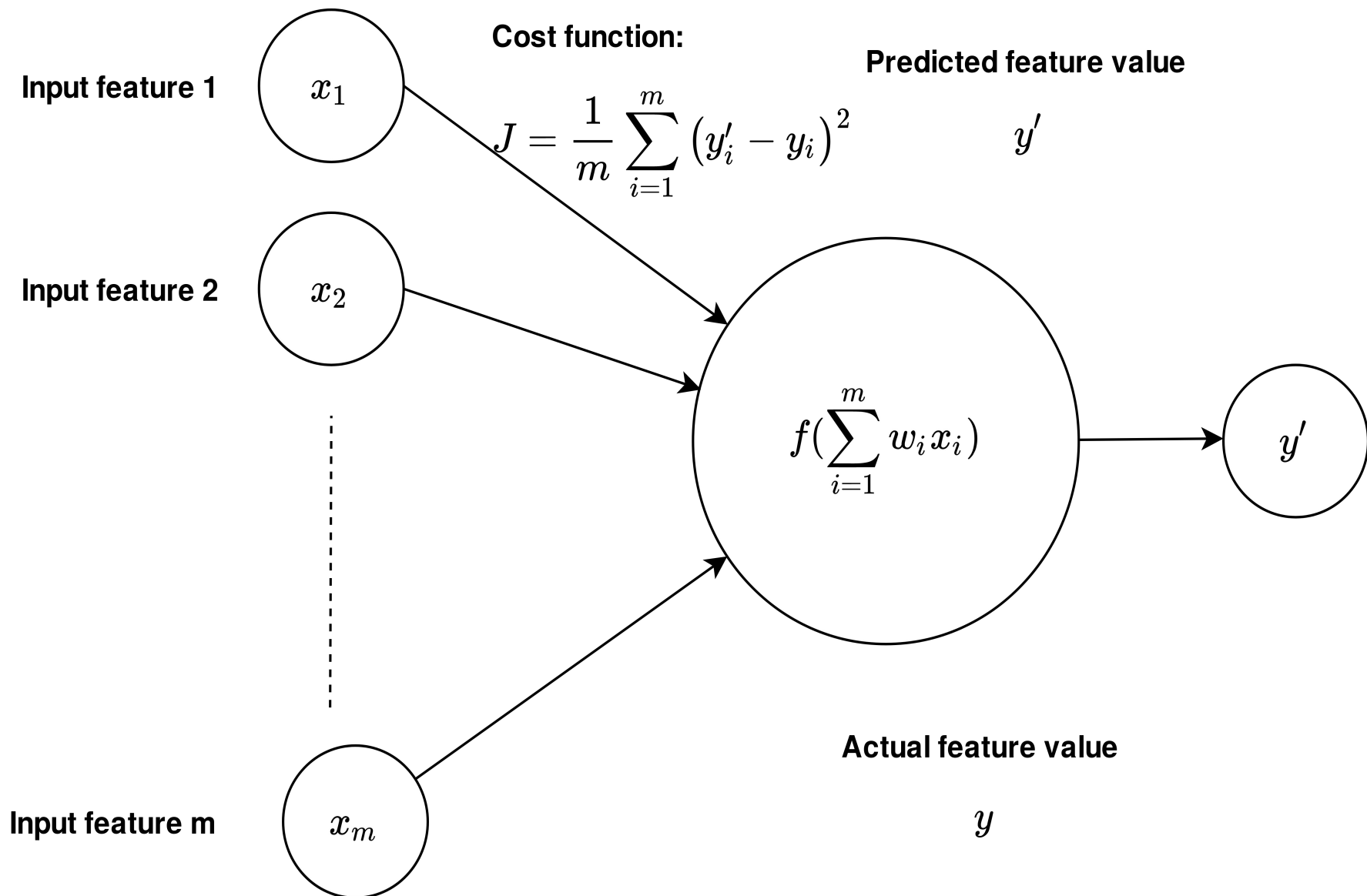
```
input_layer = Input(shape=(X.shape[1],))
dense_layer_1 = Dense(15, activation='relu')(input_layer)
dense_layer_2 = Dense(10, activation='relu')(dense_layer_1)
output = Dense(y.shape[1], activation='softmax')(dense_layer_2)

model = Model(inputs=input_layer, outputs=output)
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
```

Layers

- Dense
- Conv1D
- Conv2D
- AveragePooling1D

How Deep Neural Networks Work?



How Deep Neural Networks Work?

Forward propagation:

- Loss function is calculated
- Activation added non-linearity.

Backward propagation:

- Loss is minimized.
- Weight updated.

Dataset

Reading Dataset:

```
url = 'https://raw.githubusercontent.com/thesukantadey/TF_ADBU/master/data/car_evaluation.csv'

cols = ['price', 'maint', 'doors', 'persons', 'lug_capacity', 'safety', 'output']
cars = pd.read_csv(url, names=cols)
```

Splitting Training and Test dataset:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

Encoding, Scaling, Normalization

Encoding:

- Integer Encoding
- One-hot encoding

red,	green,	blue
1,	0,	0
0,	1,	0
0,	0,	1

Scaling/Normalization:

- Standard Scaler
- MixMax Scaler

Activation Functions

- The activation function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.
- It can be as simple as a step function that turns the neuron output on and off, depending on a rule or threshold.
- Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.

Activation Functions

- Rectified Linear Unit (ReLU)
- Sigmoid
- tanh
- Softmax (used in last layer)
- Exponential
- Leaky ReLU

ReLU Activation

- Computationally efficient—allows the network to converge very quickly.
- Non-linear—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation.

Softmax Activation

- Softmax assigns decimal probabilities to each class in a multi-class problem.
- Softmax is implemented through a neural network layer just before the output layer.
- The Softmax layer must have the same number of nodes as the output layer.

Loss Function

- Mean-square Error
- Categorical Cross Entropy
- Binary Cross Entropy
- Hinge
- Huber
- Mean Absolute Error

Mean-square Error Loss Function

- Mean Square Error (MSE) is the most commonly used regression loss function.

$$MSE = \frac{\sum_{i=1}^n (y_i - y_i^p)^2}{n}$$

Categorical Cross Entropy Loss Function

- Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.
- Cross-entropy loss increases as the predicted probability diverges from the actual label.

Optimizer

- Gradient Descent
- Stochastic Gradient Descent
- Mini-Batch Gradient Descent
- Momentum
- Adagrad
- AdaDelta
- Adam

Adam is the best optimizers. If one wants to train the neural network in less time and more efficiently than Adam is the optimizer.

Training & Testing & Accuracy

Training:

```
history = model.fit(X_train, y_train, batch_size=8, epochs=50, verbose=1, validation_split=0.2)
```

Testing:

```
score = model.evaluate(X_test, y_test, verbose=1)
```

Epochs, Batch and Iterations

- Epochs: One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.
- Batch size: Total number of training examples present in a single batch.
- Iterations: It is the number of batches needed to complete one epoch.
- Let's say we have 2000 training examples that we are going to use.
- We can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch.

Hands-on Codes on Google Colab

- Go to: <https://colab.research.google.com/>
- New Notebook.
- This will create `.ipynb` file on your browser.
- This is a python interpreter similar to Jupyter notebook.
- This can be saved in your Google drive.
- You can choose Python 2 or 3.
- You can also use Google GPU or TPU.

Thank You