Getting up and running with TensorFlow

5 min

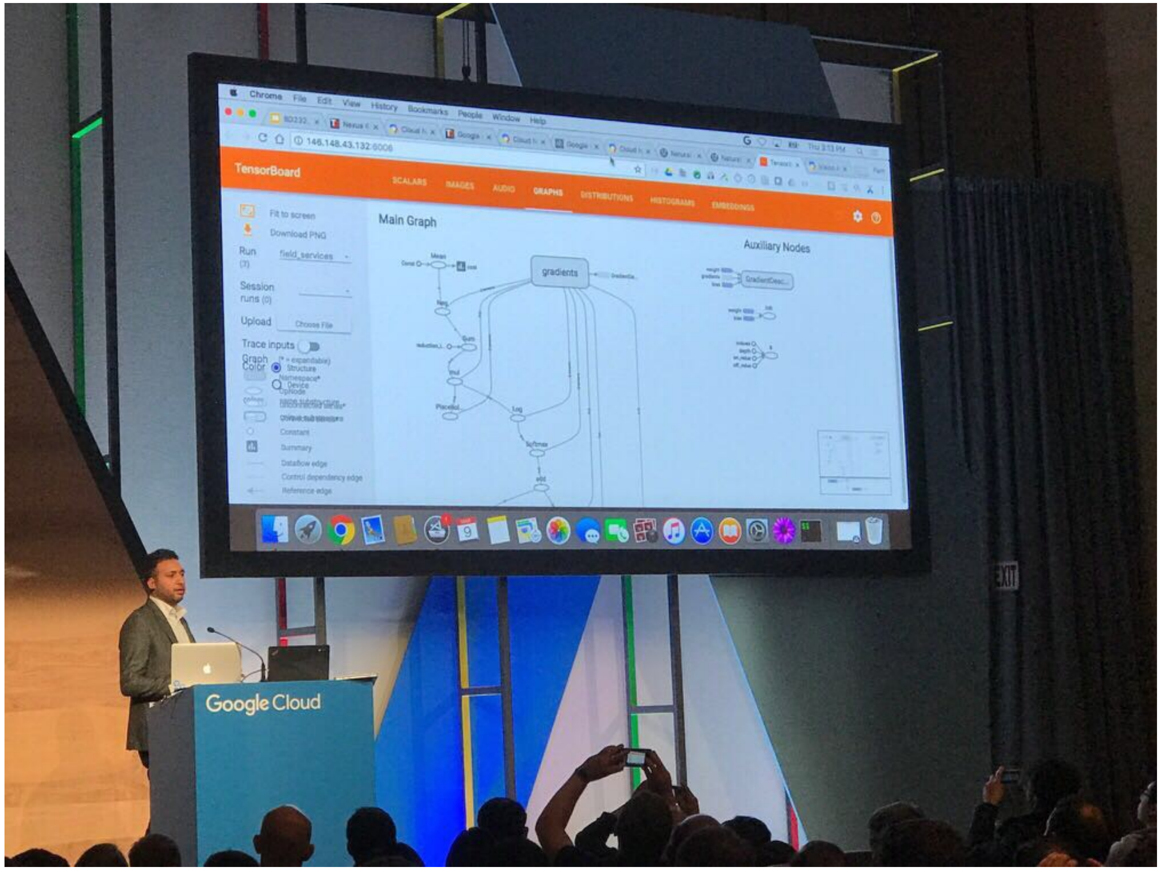
GETTING TO KNOW

Pre-requisites

- Python 2.7 or 3+
- Pandas, numpy and matplotlib
- Tensorflow for python
  - In a virtualenv

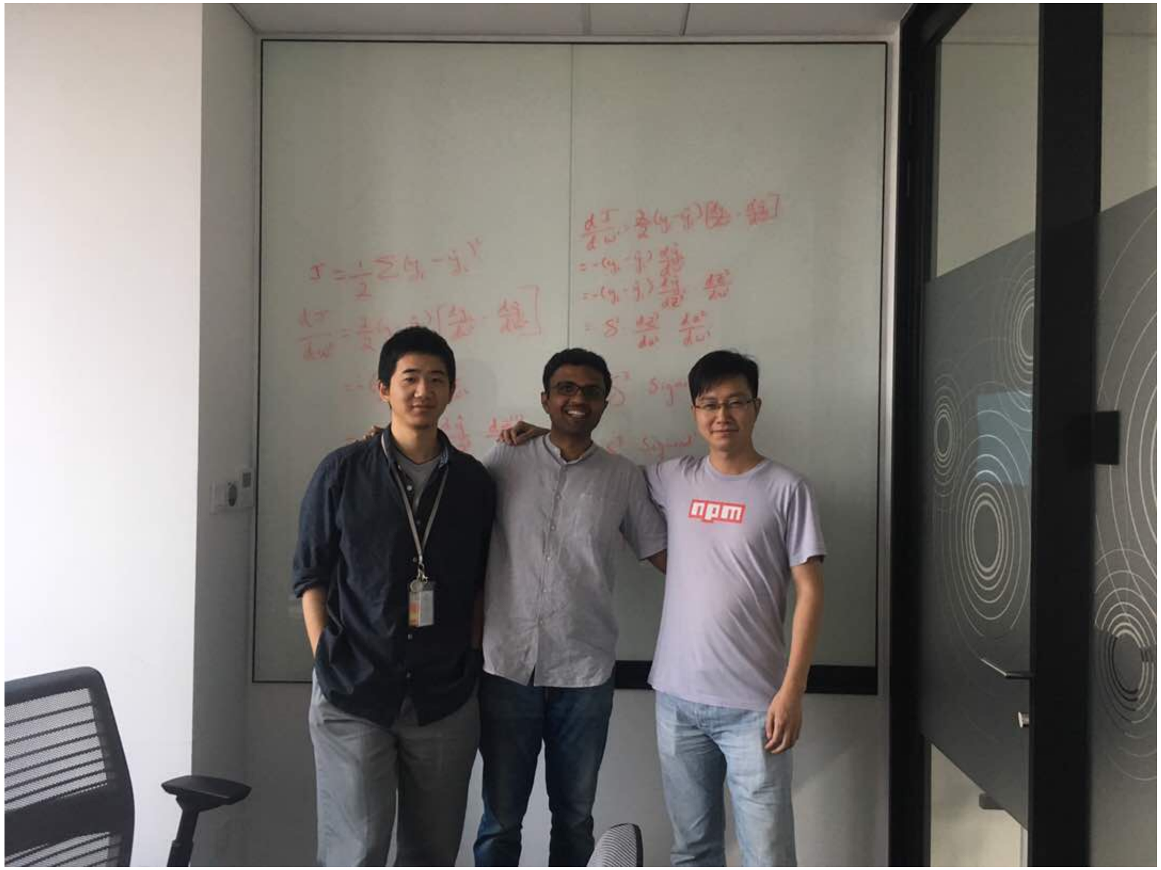
Myself

- Niranjan Salimath
- CMU ’09
- Startups
  - www.gethaggle.com
  - www.hirepirates.com
- Venture fund
  - www.latticefund.com



Implementing Google cloud for clients

Google cloud conference – March ‘17



PWC partners

Shanghai – June ‘17

Saturday math sessions

15 min

WHAT IS A TENSOR?

Why the name TensorFlow?

- A deep-learning library which lets you manipulate Tensors
- Every deep learning problem can be boiled down to manipulating Tensors

Definition

- Multilinear maps from vector spaces to real numbers
- Can represent a Scalar, Vector or a matrix
- Easiest to think of it as an n-d array in numpy
- A partially defined computation that will *eventually* produce a value

Shape

- Number of dimensions and size of each dimension
  - Dimensions are the number of indices you need to access each element
  - Array dimension NOT vector dimension
- Shapes might be fully-known or partially-known
- i.e. an operation with fully-known input will produce tensor of fully-known shape

Examples

```
In [23]: c = tf.constant([[1.0,2.0],[3.0,4.0]])

In [24]: c.shape
Out[24]: TensorShape([Dimension(2), Dimension(2)])

In [25]: print(c.shape)
(2, 2)
```

```
In [27]: x = tf.placeholder(tf.float32, [None, 784])

In [28]: x.shape
Out[28]: TensorShape([Dimension(None), Dimension(784)])

In [29]: print(x.shape)
(?, 784)
```

Rank

- Number of dimensions
- A.K.A, order, degree or n-dimension
- NOT the same as rank of a matrix
  - Number of dimensions in the output of a linear transformation

Examples

```
In [38]: scalar = tf.Variable(987, tf.int16)

In [39]: vector = tf.Variable([1.0, 2.0], tf.float32)

In [40]: matrix = tf.Variable([[1,2],[3,4]], tf.int16)

In [41]: n_tensor = tf.Variable([[[1,2],[3,4]],[[5,6],[7,8]]], tf.int16)

In [42]: sess = tf.Session()

In [43]: rank = tf.rank(n_tensor)

In [44]: sess.run(rank)
Out[44]: 3
```

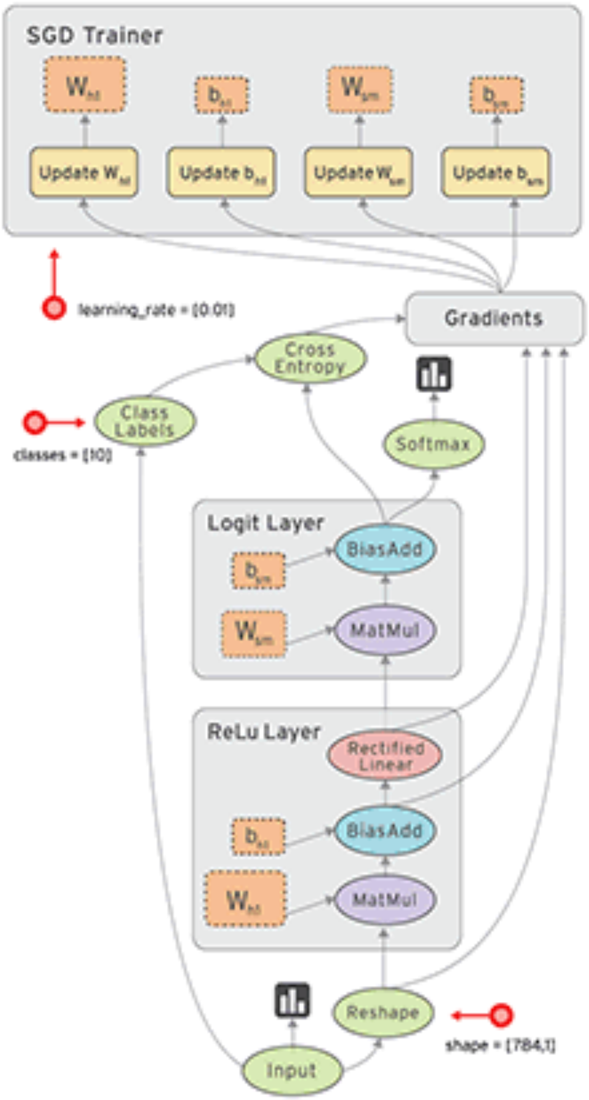
10 min

COMPUTATION GRAPHS

Definition

- All problems are represented by a graph
- Nodes represent operations
- Edges represent the flow of tensors
- Once the graph is defined, a TF session can run parts of the graph or the entire graph

Visualization



Exercise - 1

- ex1.py
- tf.convert_to_tensor
- tf.reduce_mean
- tf.reduce_sum

What is the value of the tensor??

```
In [49]: a = np.zeros((2,2))

In [50]: print(a)
[[ 0.  0.]
 [ 0.  0.]]

In [51]: np.sum(a, axis=1)
Out[51]: array([ 0.,  0.])
```

```
In [6]: tf.convert_to_tensor(tv_budget_x, dtype=np.float32, name="X_INPUT")
Out[6]: <tf.Tensor 'X_INPUT:0' shape=(200,) dtype=float32>
```

Session

- 2 basic steps to any TF program
  - Build the graph
  - Execute the graph
- A graph is executed within the context of a Session
- Connection between the client program and the C++ runtime
- Provides access to devices in the distributed TF runtime

Getting tensor value

```
In [15]: adv = pd.read_csv('Advertising.csv')

In [16]: tv_budget_x = adv.TV.tolist()

In [17]: x_tensor = tf.convert_to_tensor(tv_budget_x, dtype=np.float32, name="X_INPUT")

In [18]: print(x_tensor)
Tensor("X_INPUT_2:0", shape=(200,), dtype=float32)

In [19]: with tf.Session() as sess:
    ...:     res = sess.run(x_tensor)
    ...:     print(res)
```

Interactive Session

```
In [20]: adv = pd.read_csv('Advertising.csv')

In [21]: tv_budget_x = adv.TV.tolist()

In [22]: x_tensor = tf.convert_to_tensor(tv_budget_x, dtype=np.float32, name="X_INPUT")

In [23]: tf.InteractiveSession()
Out[23]: <tensorflow.python.client.session.InteractiveSession at 0x11021bfd0>

In [24]: x_tensor.eval()
Out[24]:
```

30 min

NON-ML LINEAR REGRESSION

SVLR

- Model for SVLR is:

$$Y = \beta_1 * X + \beta_0 + \epsilon$$

- Model params to be learned are:
  - Slope
  - Intercept
- Cost function is:

$$J = \sum_{i=1}^{i=n} (y_i - y_p)^2/2$$

Estimating model params

- ith residual
- Residual sum of squares

$$RSS = (e_1)^2 + (e_2)^2 + (e_3)^2 + +(e_n)^2$$

- Values of slope and intercept which minimize this:

$$\beta_1 = \sum_{i=1}^{n} (y_i - y_a) * (x_i - x_a) / \sum_{i=1}^{n} (x_i - x_a)^2$$

$$\beta_0 = y_a - \beta_1 * x_a$$

20 min

TENSORBOARD COMPUTATION GRAPH

Visualization

- Computation graph
- Quantitative metrics about the execution of your graph
- Works by reading event files with summary data
- Graph nodes are annotated with summary operations

Just 1 line of code

- writer = tf.summary.FileWriter('./graphs', sess.graph)

- tensorboard --logdir=./graphs

Making it pretty

- Named operations
- Named scopes
  - with tf.name_scope(”foo")

35 min

ML LINEAR REGRESSION

Tensor types

- Constant
  - Self explanatory, seen it already
- Variable
  - Holds values which are updated during training
- Placeholder
  - Usually used as the input tensor to start training
  - Kinda saw the need when we used convert_to_tensor
  - Values fed from feed_dict

Variable initialization

```
In [5]: w1 = tf.ones((2,2))

In [6]: w2 = tf.Variable(tf.zeros((2,2)))

In [7]: with tf.Session() as sess:
   ...:     print(sess.run(w1))
   ...:     sess.run(tf.global_variables_initializer())
   ...:     print(sess.run(w2))
   ...:
```

SVLR - Refresher

- Model for SVLR is:

$$Y = \beta_1 * X + \beta_0 + \epsilon$$

- Model params to be learned are:
  - Slope
  - Intercept
- Cost function is:

$$J = \sum_{i=1}^{i=n} (y_i - y_p)^2 / 2$$

Gradient Descent

- Every model has an error or cost function – J
- J is a function of model parameters
- We differentiate J w.r.t model parameters to reach the least value
- Value of model parameters <u>where error is least</u> is the <u>learned values</u> of the parameters
- 2 types
  - Stochastic and batch

25 min

TENSORBOARD
VISUALISING TRAINING

Visualization - Refresher

- Computation graph
- Quantitative metrics about the execution of your graph
- Works by reading event files with summary data
- Graph nodes are annotated with summary operations

Steps

- Annotate a graph node
  - tf.summary.scalar(”foo", bar)
  - tf.summary.histogram(”foo", bar)
- Merge all annotations
  - tf.summary.merge_all()
- Run merge operation
- Add summary to file writer
  - writer.add_summary()

1 hour

BUILDING A NEURAL NETWORK

Activation functions

- You have an existing function, but you want to scale its values between 0&1
- In the linear case: $$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$
- Called the "logistic function" or "Sigmoid"
- "Odds" is given by: $$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}.$$
- Log-odds or logits: $$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0+\beta_1 X.$$

Cost function for logistic regression

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})).$$

- Called cross entropy
  - a = 1/1 + tf.exp(-tf.add(tf.multiply(x, W), b))
  - tf.reduce_mean(-(y * tf.log(a) + (1 - y) * tf.log(1 - a)))
- What is log(0)?
- (Quick code walk thru of logistic regresssion)

NN basics

- 2 building blocks
  - Synapse: Connects neurons
  - Neuron: Performs a very simple function
- Synapse:
  - 1 input 1 output
  - Multiplies its input by weight
- Neuron
  - Multiple input 1 output
  - Adds its inputs and applies an activation function

NN parameters

- Non learned
  - Input layer
    - Depends on input
  - Output layer
    - 1
  - Hidden layer
    - Depends on how ‘deep’ we want to go
- Learned
  - Synapse weights

Traversing

- Forward propogation
  - Move from input layer to output layer
  - Generate a predicted value for Y
  - Assume values for weights the first time
- Back propagation
  - Move from output layer to input layer
  - Generate small changes to weight values
  - Subtract these small changes from previous weights
- Repeat till happy!

Finally, TensorFlow’s HelloWorld!

- https://www.tensorflow.org/get_started/mnist/beginners