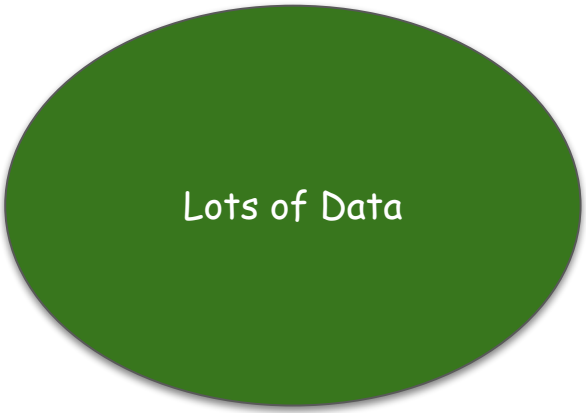
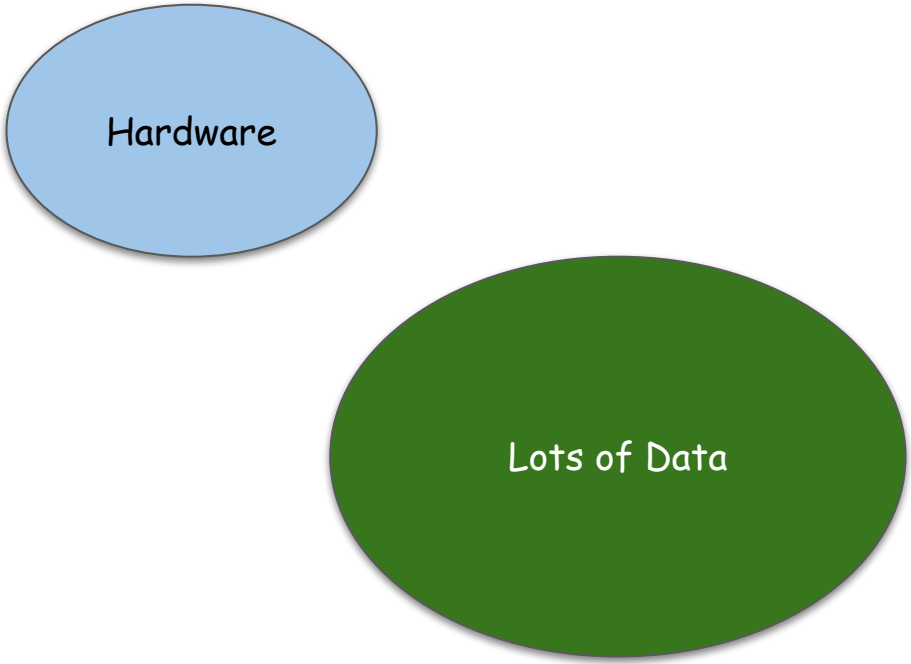
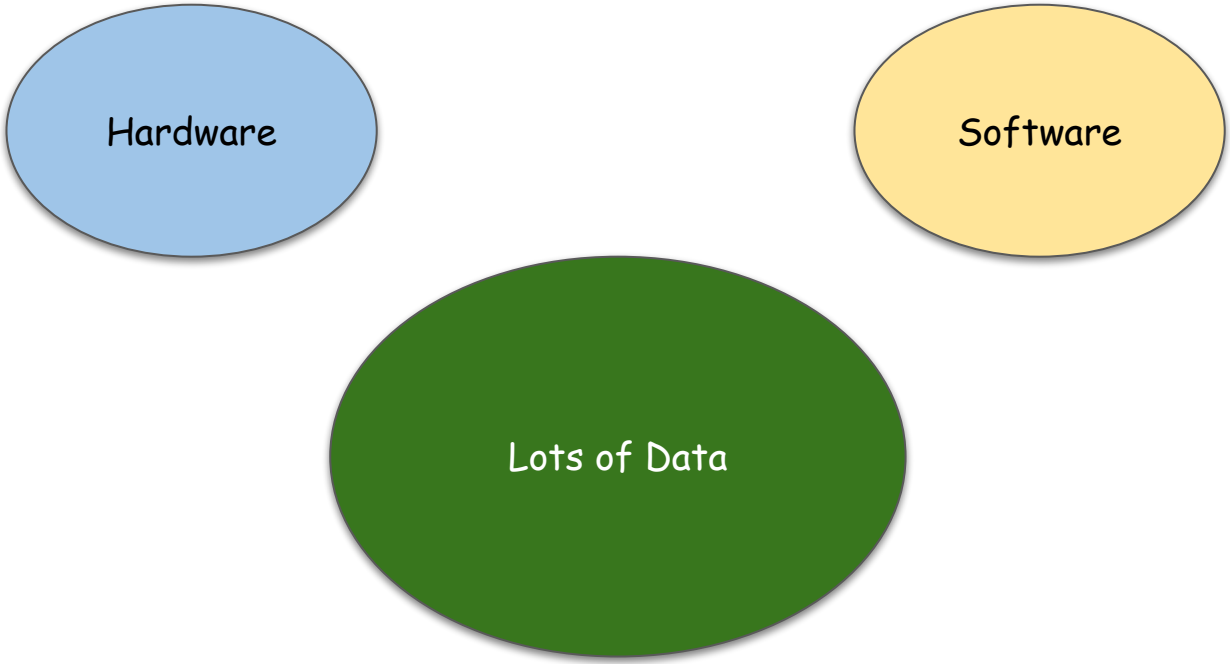
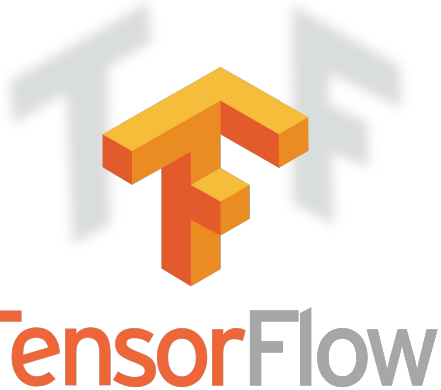
What do we need for

Machine Learning


Lots of Data


Hardware
Lots of Data


Hardware
Software
Lots of Data



Open Source platform for Deep Learning

by Google

TensorFlow

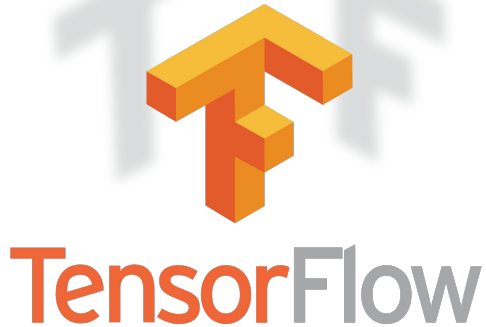
- **Supported Platforms**
  - Linux / Ubuntu
  - Windows
  - Mac OS

TensorFlow

- **Supported Languages**
  - Python
  - C++
  - Java (Limited)
  - R (WIP)

tensorflow.org/install

Additional Packages



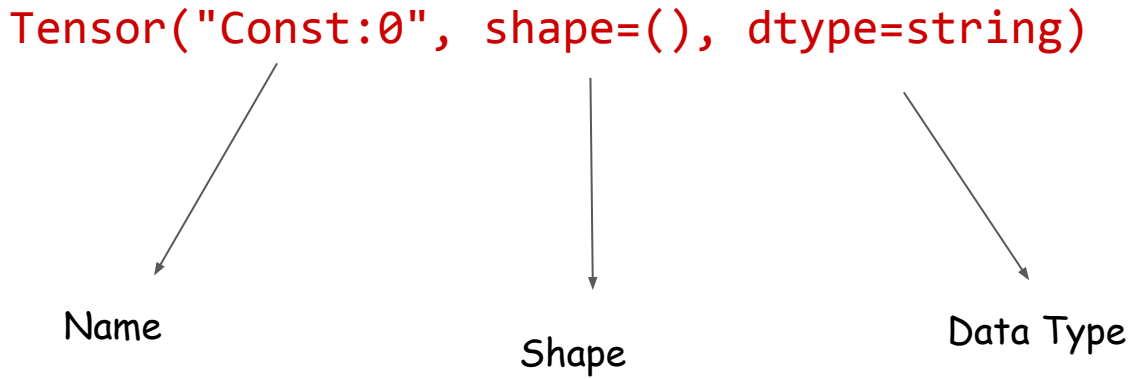
Let’s start with… Hello World :)

```
import tensorflow as tf
hello = tf.constant('Hello World')
print (hello)
```

Not so simple :)

```
Tensor("Const:0", shape=(), dtype=string)
```

what is a ‘tensor’


Tensor("Const:0", shape=(), dtype=string)
Name
Shape
Data Type

Name

```
import tensorflow as tf
hello = tf.constant("Hello World",name="my_tensor")
print (hello)
```

Name

```
import tensorflow as tf
hello = tf.constant("Hello World",name="my_tensor")
print (hello)
```

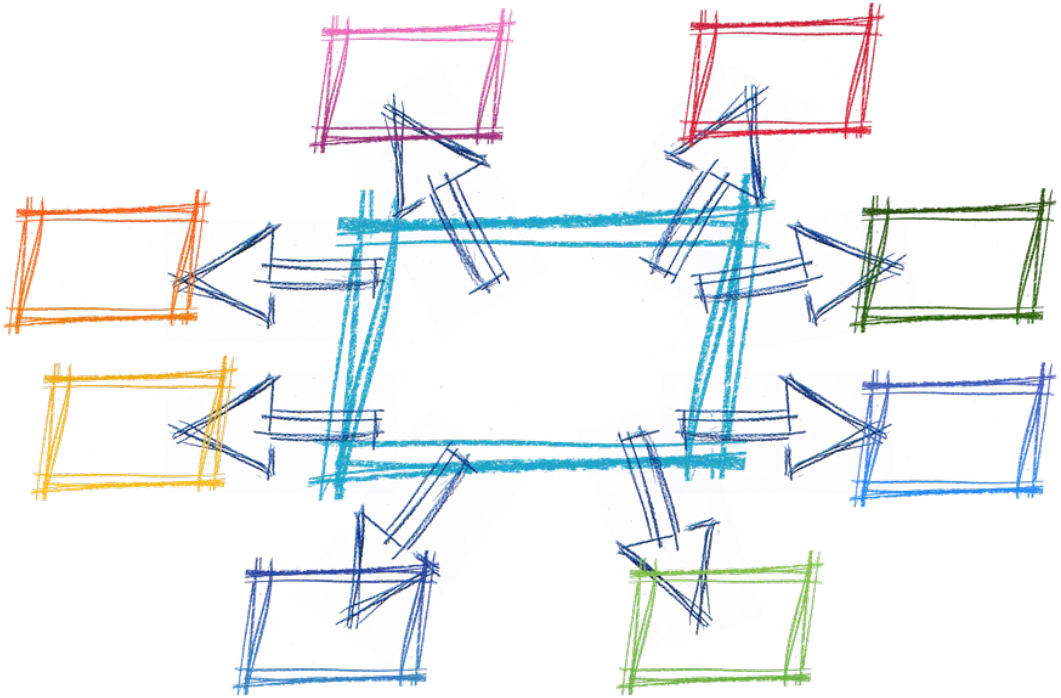
```
Tensor("my_tensor:0", shape=(), dtype=string)
```

Shape

```
import tensorflow as tf
hello = tf.constant("Hello",name="my_tensor",shape=[6,2])
print (hello)
```

```
Tensor("my_tensor:0", shape=(6,2), dtype=string)
```

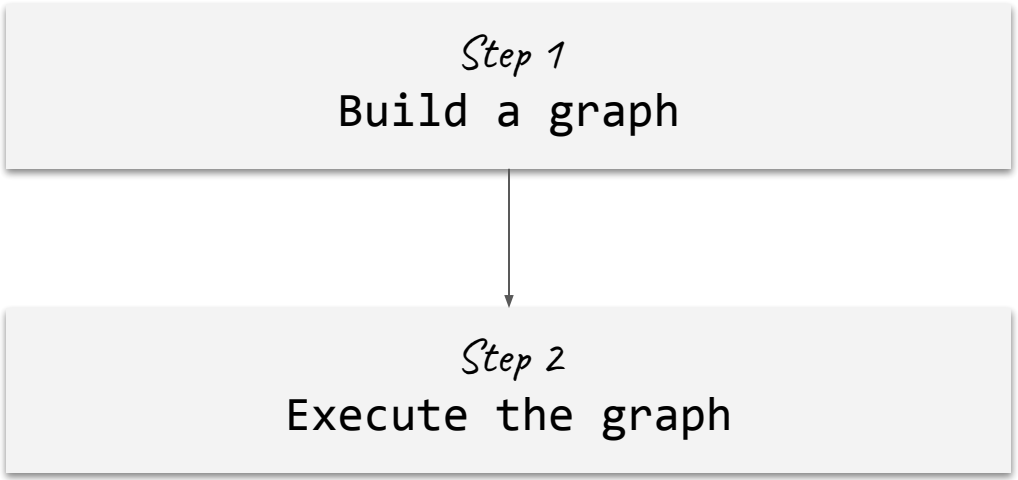
How do we print Hello World ?



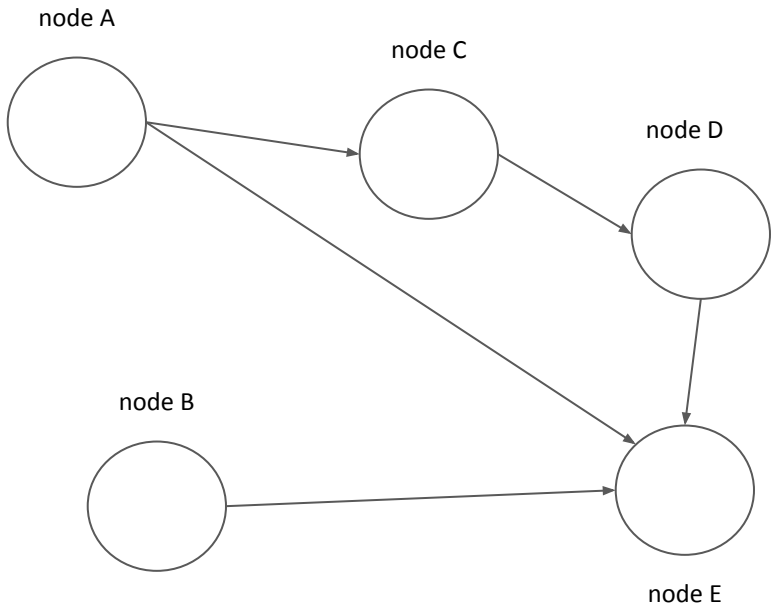
Understanding

'Computational Graph'

Computational Graph



What is a Graph?

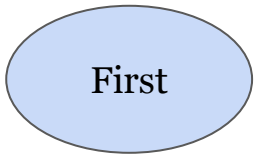


Connected nodes

Building Graph

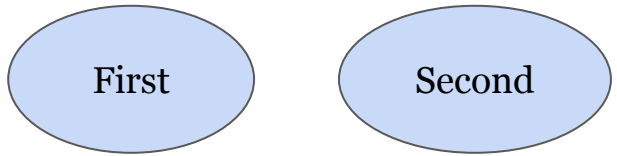
$$c = a + b$$

```
import tensorflow as tf
a = tf.constant([1.2],name='First',dtype= tf.float32)
```



```
import tensorflow as tf
a = tf.constant([1.2],name='First',dtype= tf.float32)
```

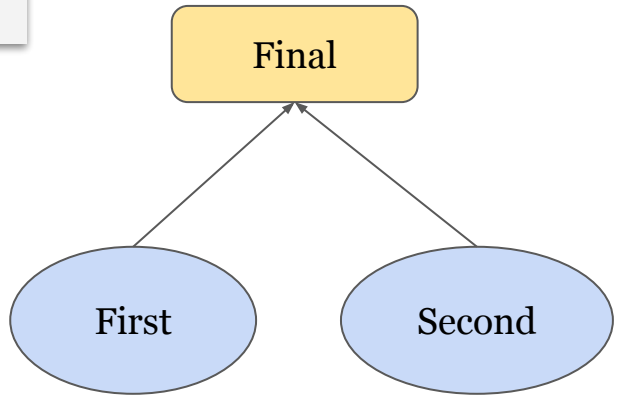
```
b = tf.constant([3.4],name='Second',dtype=tf.float32)
```



```
import tensorflow as tf
a = tf.constant([1.2],name='First',dtype= tf.float32)
```

```
b = tf.constant([3.4],name='Second',dtype=tf.float32)
```

```
C = tf.add(a,b,name='Final')
```

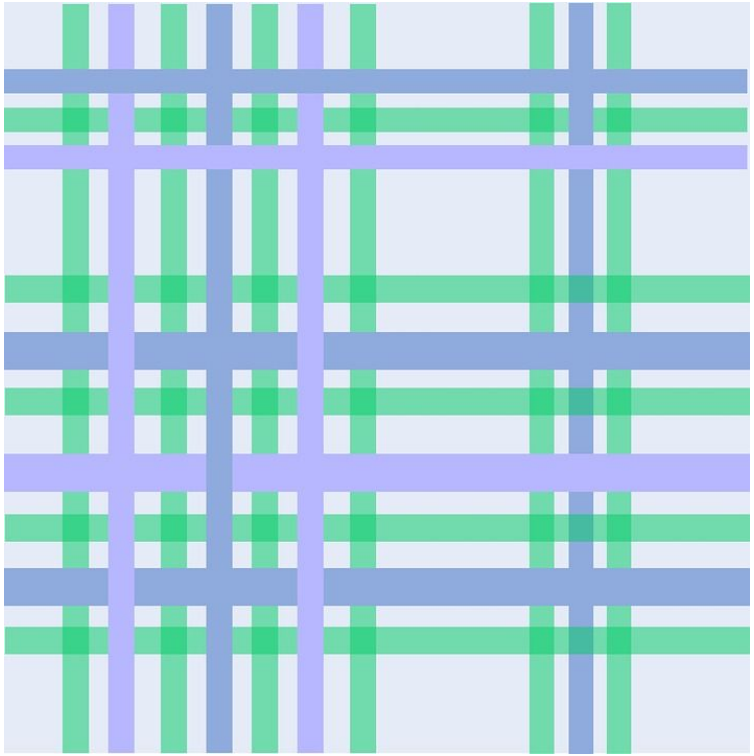


Executing Graph

c = a + b

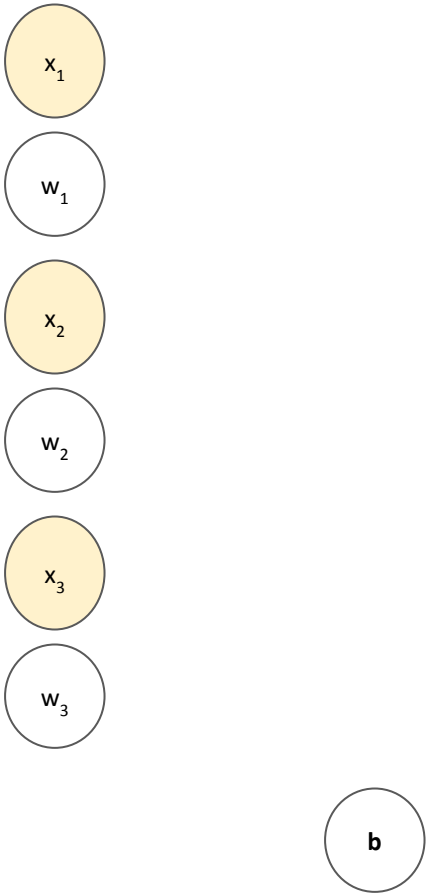
```
With tf.Session() as sess:

    print(sess.run(c))
```

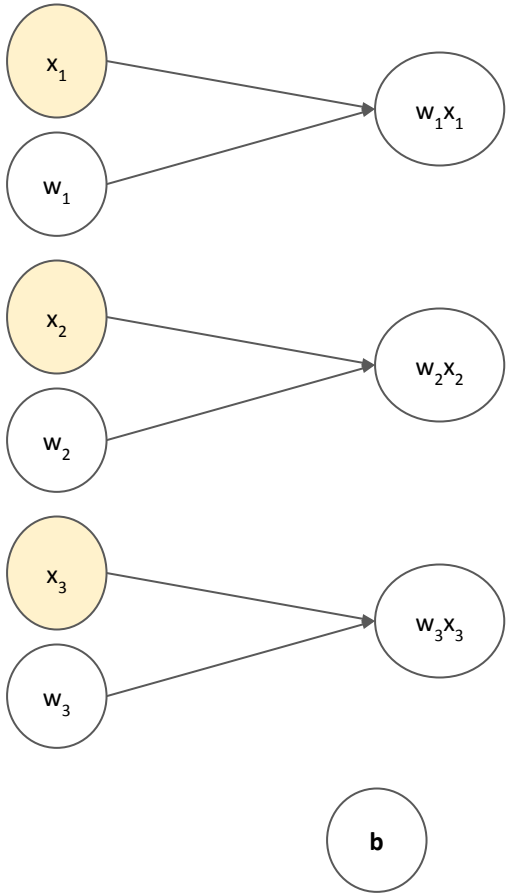
4.6

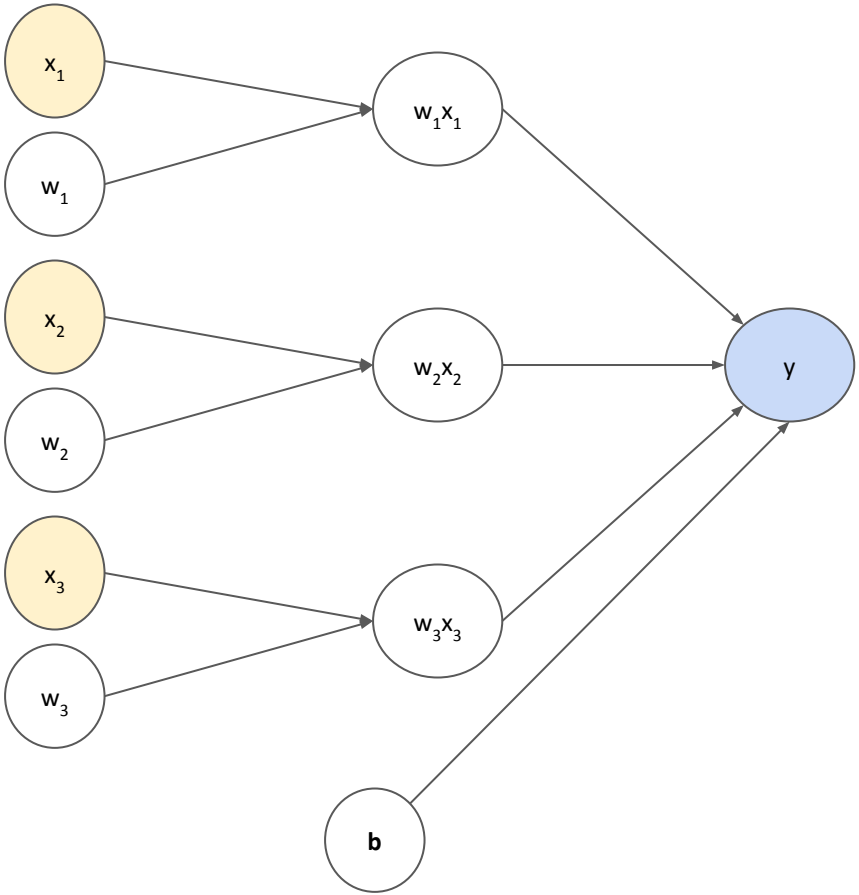


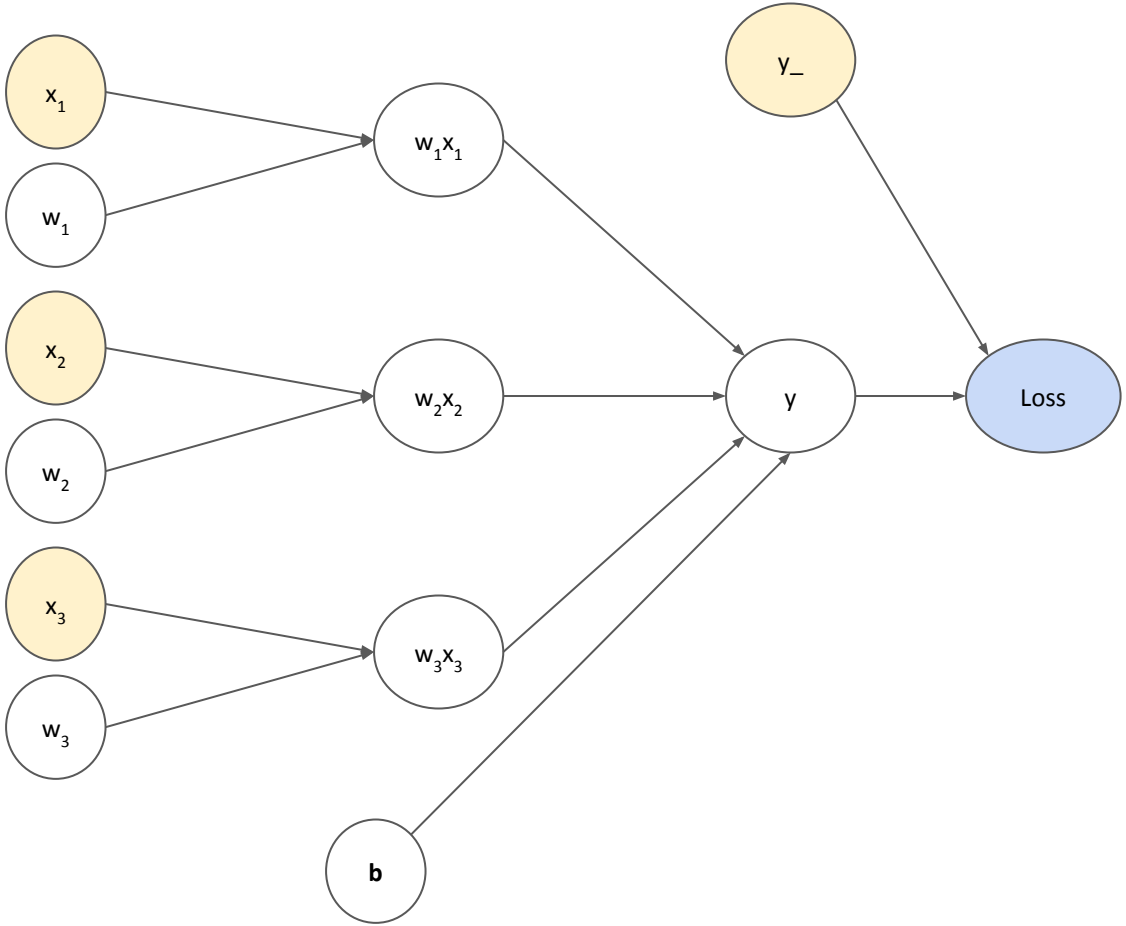
Rivisting
Linear Regression

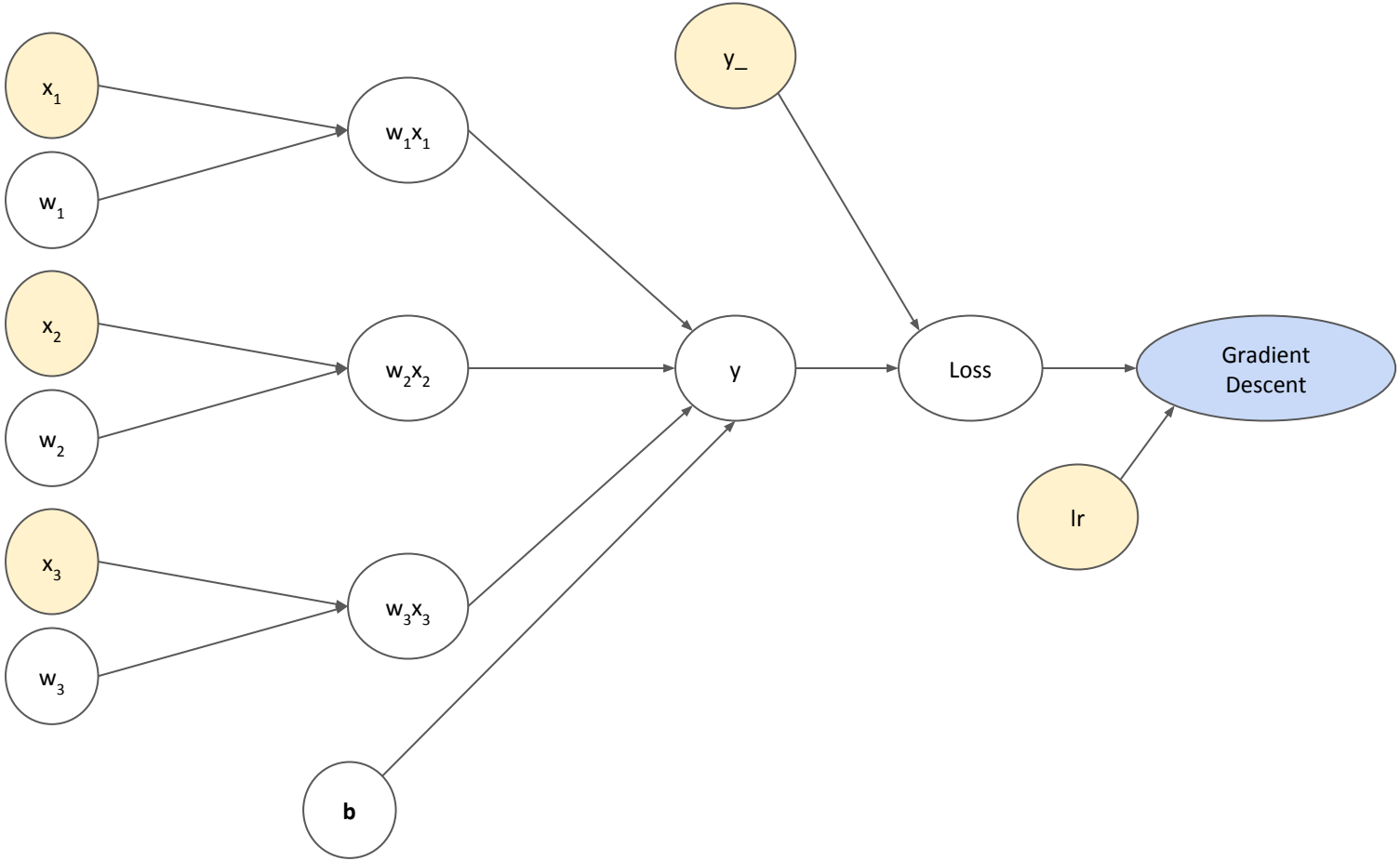
$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

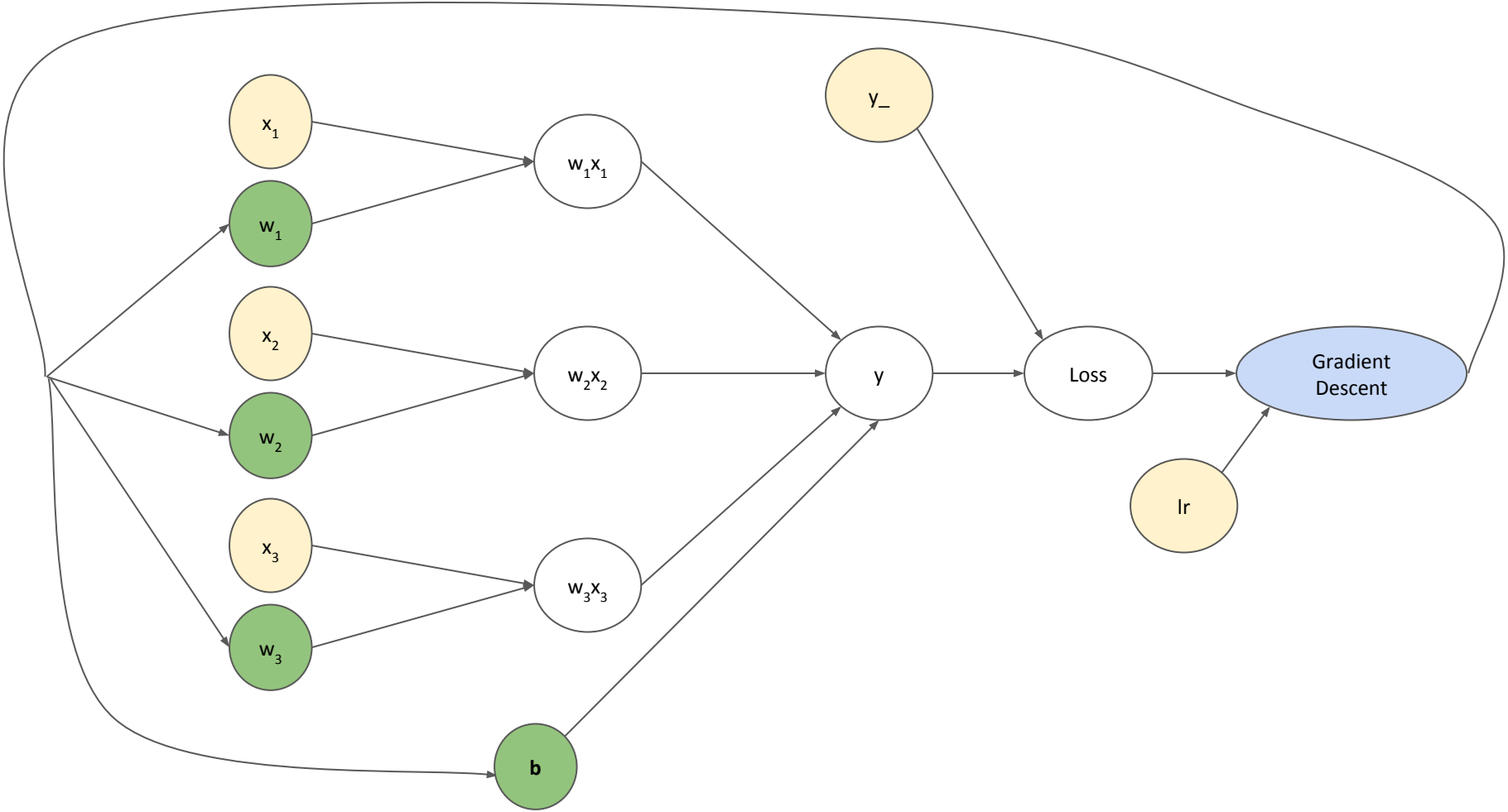

x_1
w_1
x_2
w_2
x_3
w_3
b


x_1
w_1
w_1x_1
x_2
w_2
w_2x_2
x_3
w_3
w_3x_3
b


x_1
w_1
x_2
w_2
x_3
w_3
w_1x_1
w_2x_2
w_3x_3
b
y

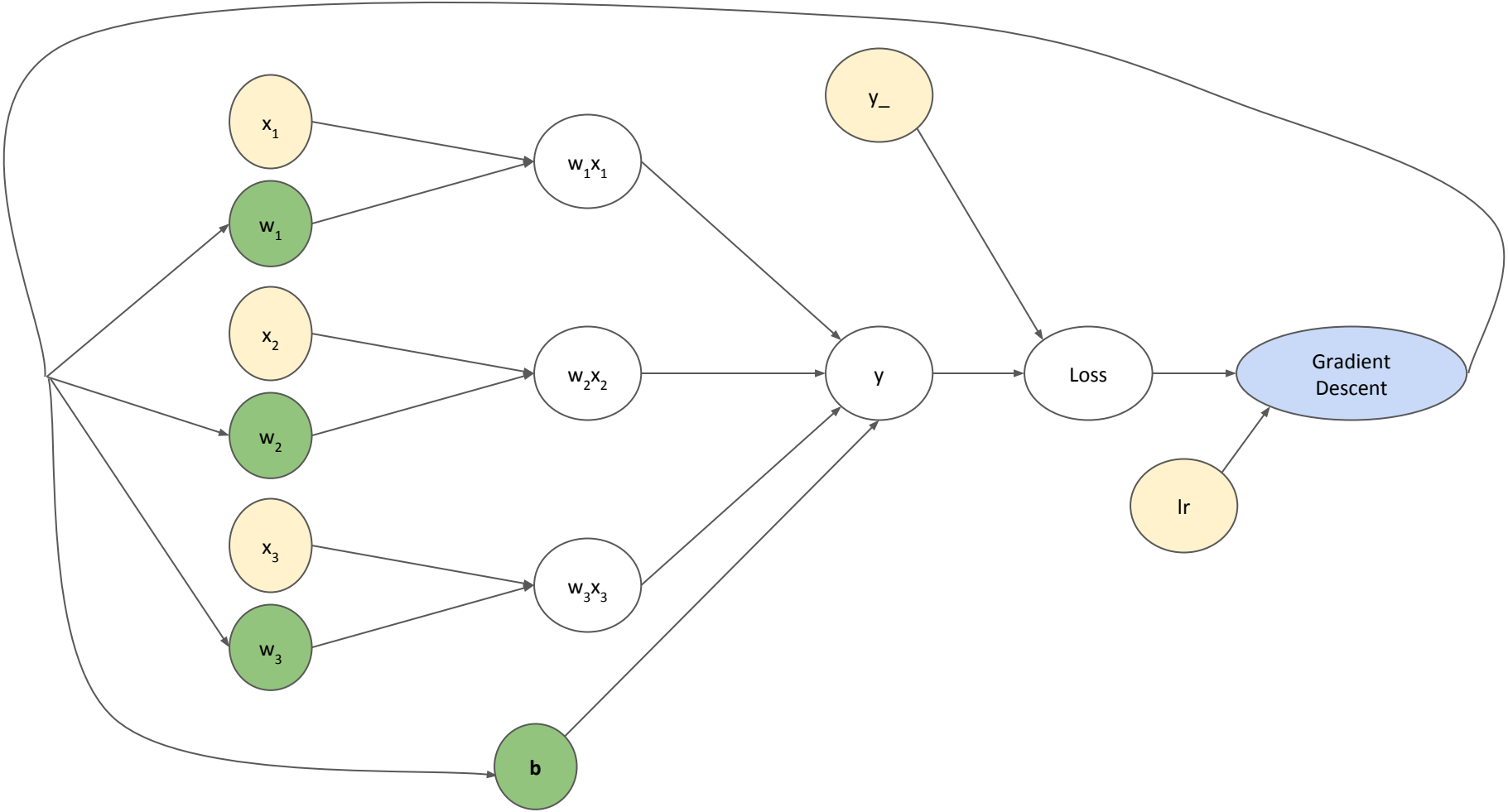

x1
w1
x2
w2
x3
w3
b
w1x1
w2x2
w3x3
y
y_
Loss


x1
w1
x2
w2
x3
w3
b
w1x1
w2x2
w3x3
y_
y
Loss
Gradient Descent
lr


x1
w1
x2
w2
x3
w3
b
w1x1
w2x2
w3x3
y_
y
Loss
Gradient Descent
lr

Some key Tensor Ops

- tf.constant(...)
  - Value can NOT change during graph execution.
- tf.Variable(...)
  - Value can change during graph execution.
- tf.placeholder(...)
  - Provide data during during execution.


x1
w1
x2
w2
x3
w3
b
w1x1
w2x2
w3x3
y_
y
Loss
Gradient Descent
lr

Defining Tensors

- Input features (x_1, x_2, x_3)
  - tf.placeholder()

Defining Tensors

- Input features (x_1, x_2, x_3)
  - tf.placeholder()
- Weights and Bias(w_1, w_2, w_3, b)
  - tf.Variable()

Defining Tensors

- Input features (x_1, x_2, x_3)
  - tf.placeholder()
- Weights and Bias(w_1, w_2, w_3, b)
  - tf.Variable()
- Actual output(y_)
  - tf.placeholder()

Boston Housing Prices

Exercise

Scenario

- **What needs to be done**
  - Build a Linear Regressor to predict Housing Prices for Boston
  - Use Tensorflow to build a Linear Regressor model

- **What is given**
  - Housing Prices data (506 examples)
  - 13 features and Price

Data Set Information:

Concerns housing values in suburbs of Boston.

Attribute Information:

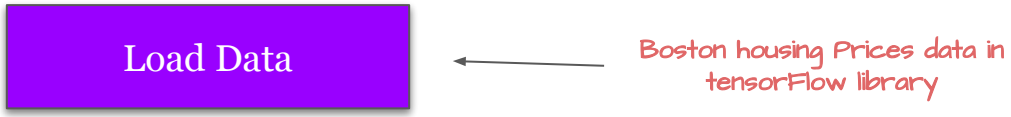
1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town
4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX: nitric oxides concentration (parts per 10 million)
6. RM: average number of rooms per dwelling
7. AGE: proportion of owner-occupied units built prior to 1940
8. DIS: weighted distances to five Boston employment centres
9. RAD: index of accessibility to radial highways
10. TAX: full-value property-tax rate per $10,000
11. PTRATIO: pupil-teacher ratio by town
12. B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT: % lower status of the population
14. MEDV: Median value of owner-occupied homes in $1000's

Build Boston Housing Model in TensorFlow

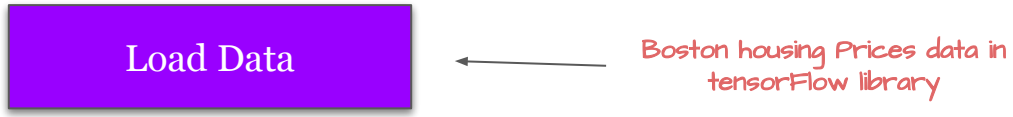
Steps to build a Program in TensorFlow

1

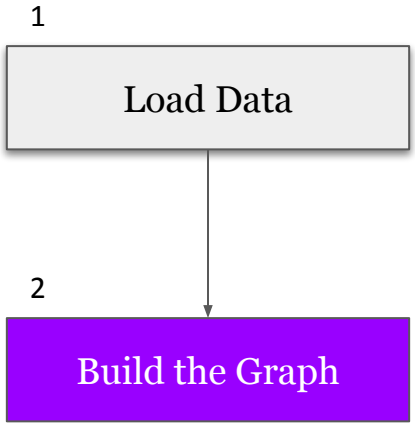
1


Load Data
Boston housing Prices data in tensorFlow library

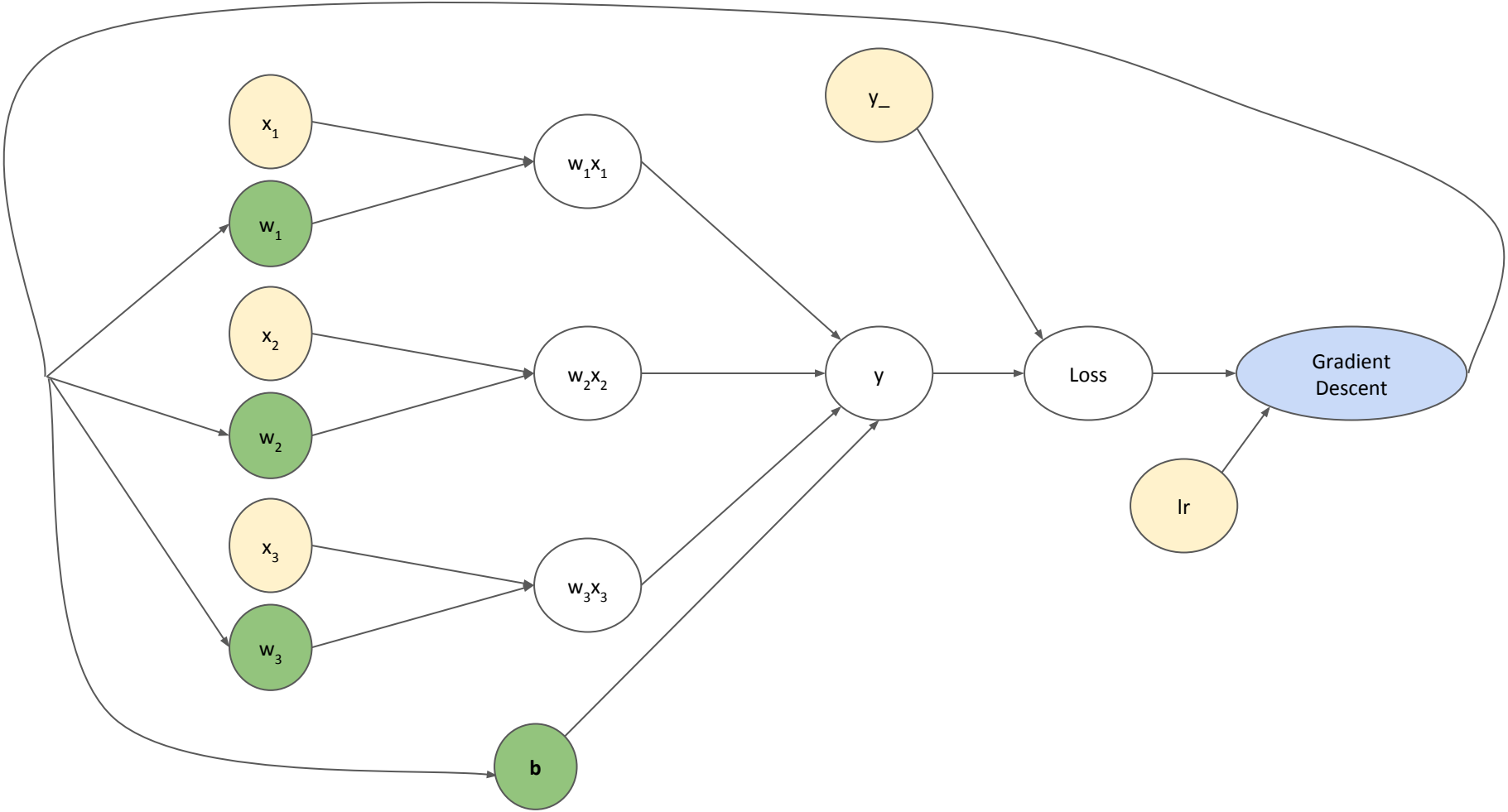
1




```
import tensorflow as tf
import numpy as np

#Take data from tensorflow library
from tensorflow.contrib.learn import datasets
boston = datasets.load_dataset('boston')

#Input Features and Actual Price
features = np.array(boston.data)
target = np.array(boston.target)
```



What are steps to build a graph?


x1
w1
x2
w2
x3
w3
b
w1x1
w2x2
w3x3
y_
y
Loss
Gradient Descent
lr

2.1

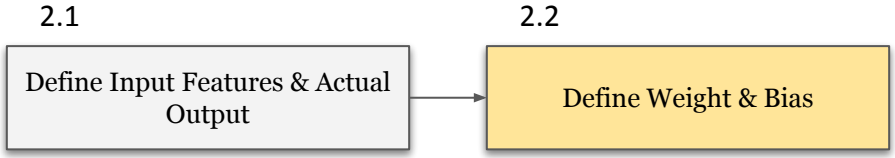
Define Input Features & Actual Output

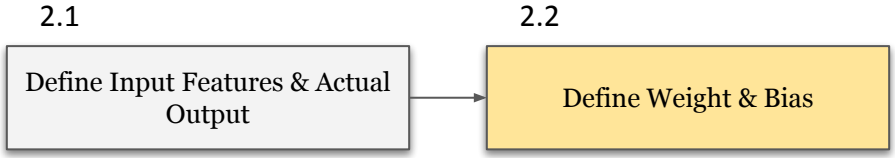
2.1

Define Input Features & Actual Output

What should be the value here?

```
x = tf.placeholder(shape=[None,?],dtype=tf.float32, name='x-input')

y_ = tf.placeholder(shape=[None,?],dtype=tf.float32, name='y-input')
```


2.1
Define Input Features & Actual Output
2.2
Define Weight & Bias



Number of features

Number of output

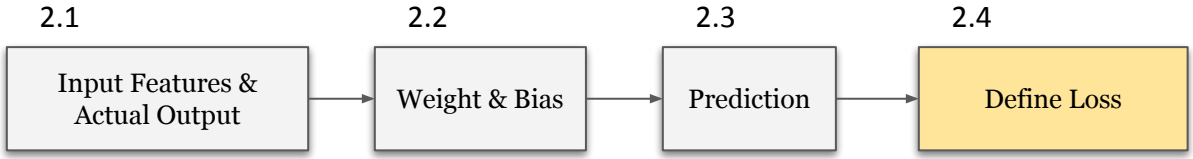
```
W = tf.Variable(tf.zeros(shape=[?,1]), name="Weights")

b = tf.Variable(tf.zeros(shape=[1]),name="Bias")
```


```
y = tf.add(tf.matmul(x,W),b,name='output')
```

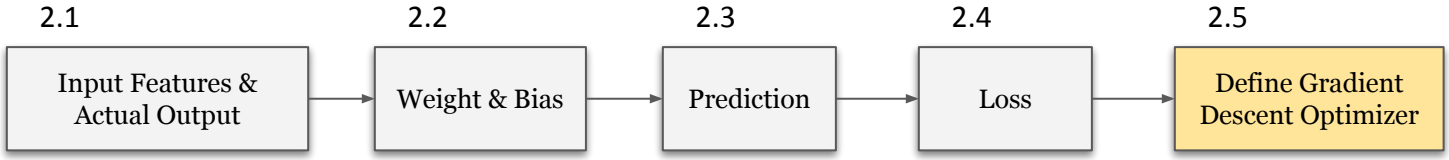
Shape of x?

Shape of W?

Shape of y?



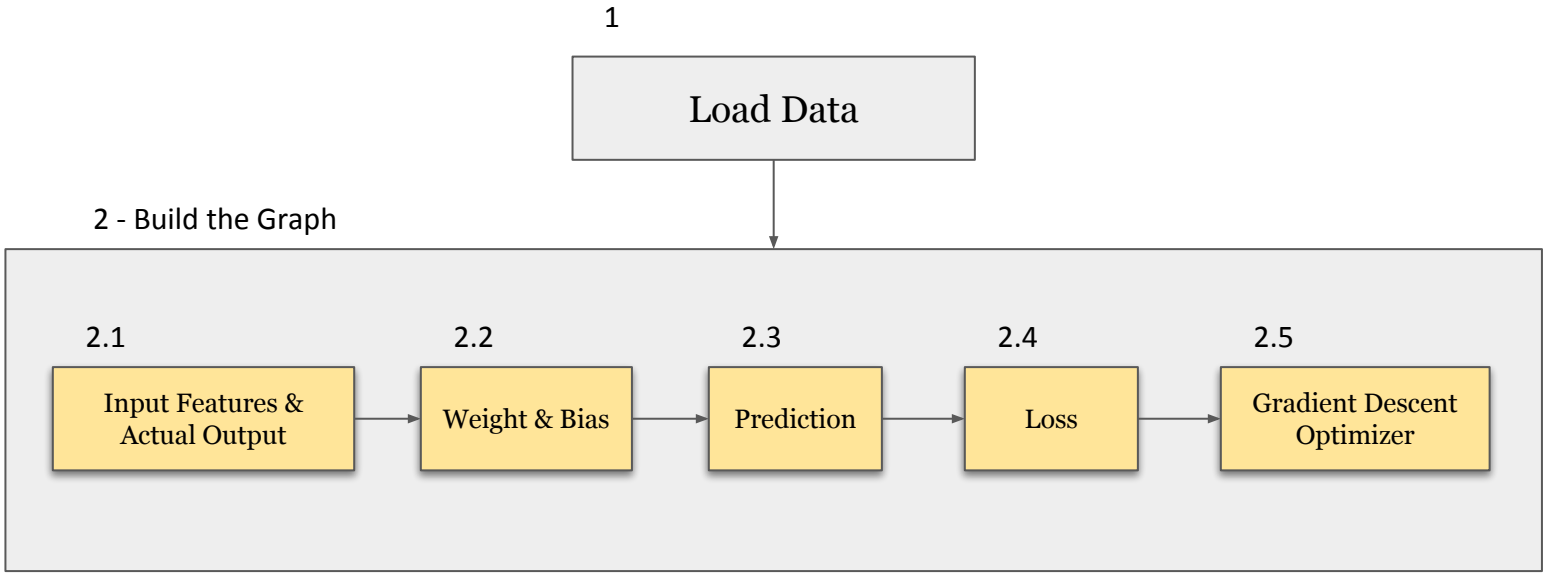

```
loss = tf.reduce_mean(tf.square(y-y_),name='Loss')
```

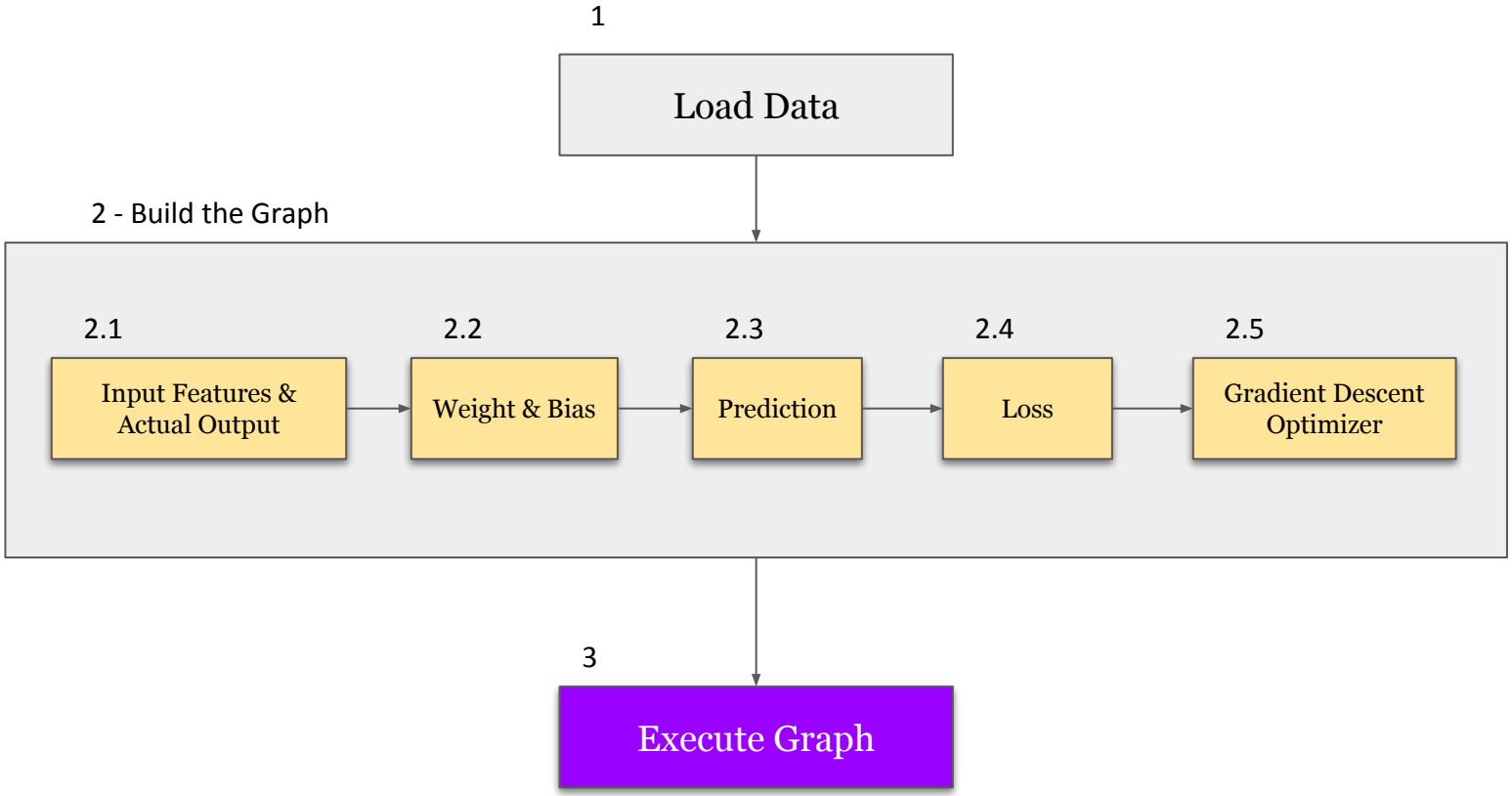


Learning rate

```
train_op = tf.train.GradientDescentOptimizer(0.03).minimize(loss)
```

We are trying to minimize loss

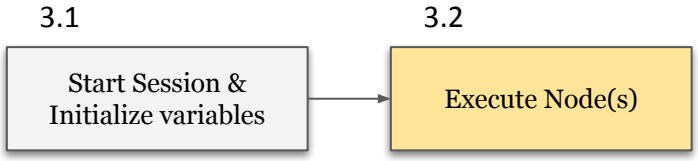

1
Load Data
2 - Build the Graph
2.1
Input Features & Actual Output
2.2
Weight & Bias
2.3
Prediction
2.4
Loss
2.5
Gradient Descent Optimizer


1
Load Data
2 - Build the Graph
2.1
Input Features & Actual Output
2.2
Weight & Bias
2.3
Prediction
2.4
Loss
2.5
Gradient Descent Optimizer
3
Execute Graph

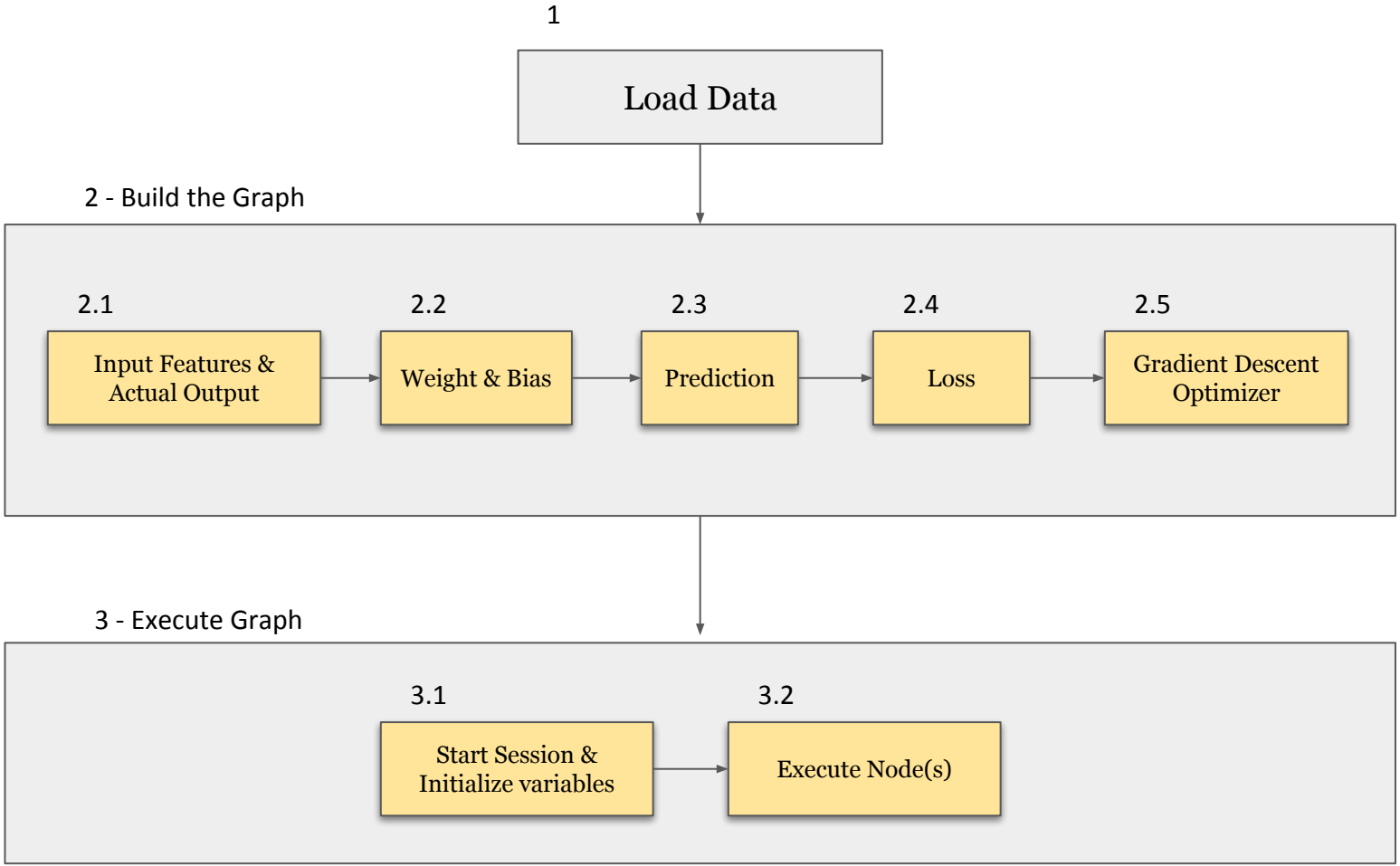
3.1

Start Session &
Initialize variables

```
with tf.Session() as sess:
    # variables need to be initialized before we can use them
    sess.run(tf.global_variables_initializer())
```




```
training_epochs = 1000 #number of iterations


for epoch in range(training_epochs):
    #Calculate train_op and loss
    _, train_loss = sess.run([train_op,loss],feed_dict={x:features, y_:prices})

    if epoch % 100 == 0:
        print ('Training loss at step: ', epoch, ' is ', train_loss)
```


1
Load Data
2 - Build the Graph
2.1
Input Features & Actual Output
2.2
Weight & Bias
2.3
Prediction
2.4
Loss
2.5
Gradient Descent Optimizer
3 - Execute Graph
3.1
Start Session & Initialize variables
3.2
Execute Node(s)

Data Normalization

$$X_i = (X_i - \text{mean})/(\text{max} - \text{min})$$