Basics of Machine Learning Using TensorFlow

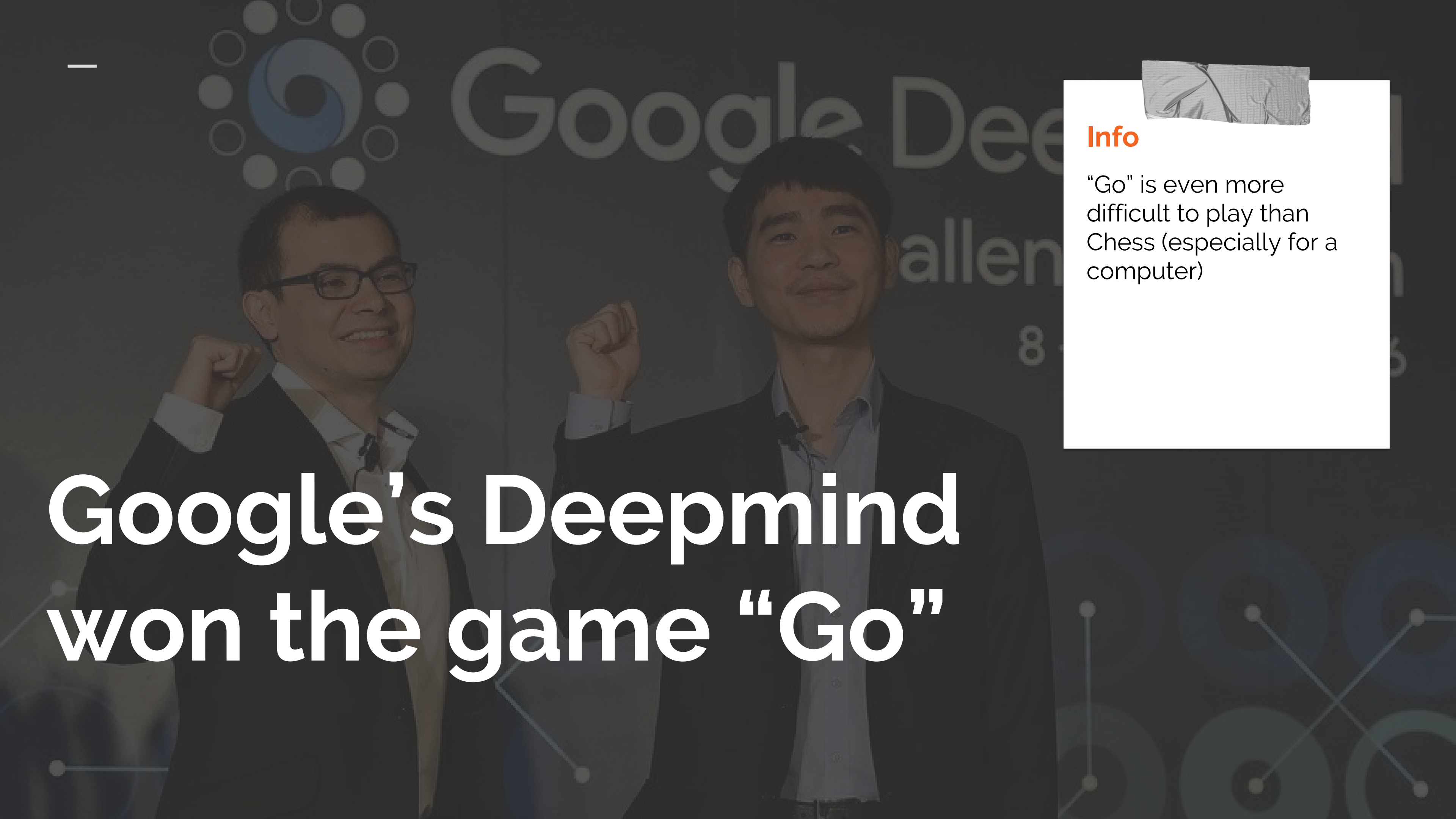
Piyush Bhopalka

Why so serious?

Machine Learning

Jarvis by Mark Zuckerberg

Can do a lot of household works, takes care of security



Google’s Deepmind won the game “Go”

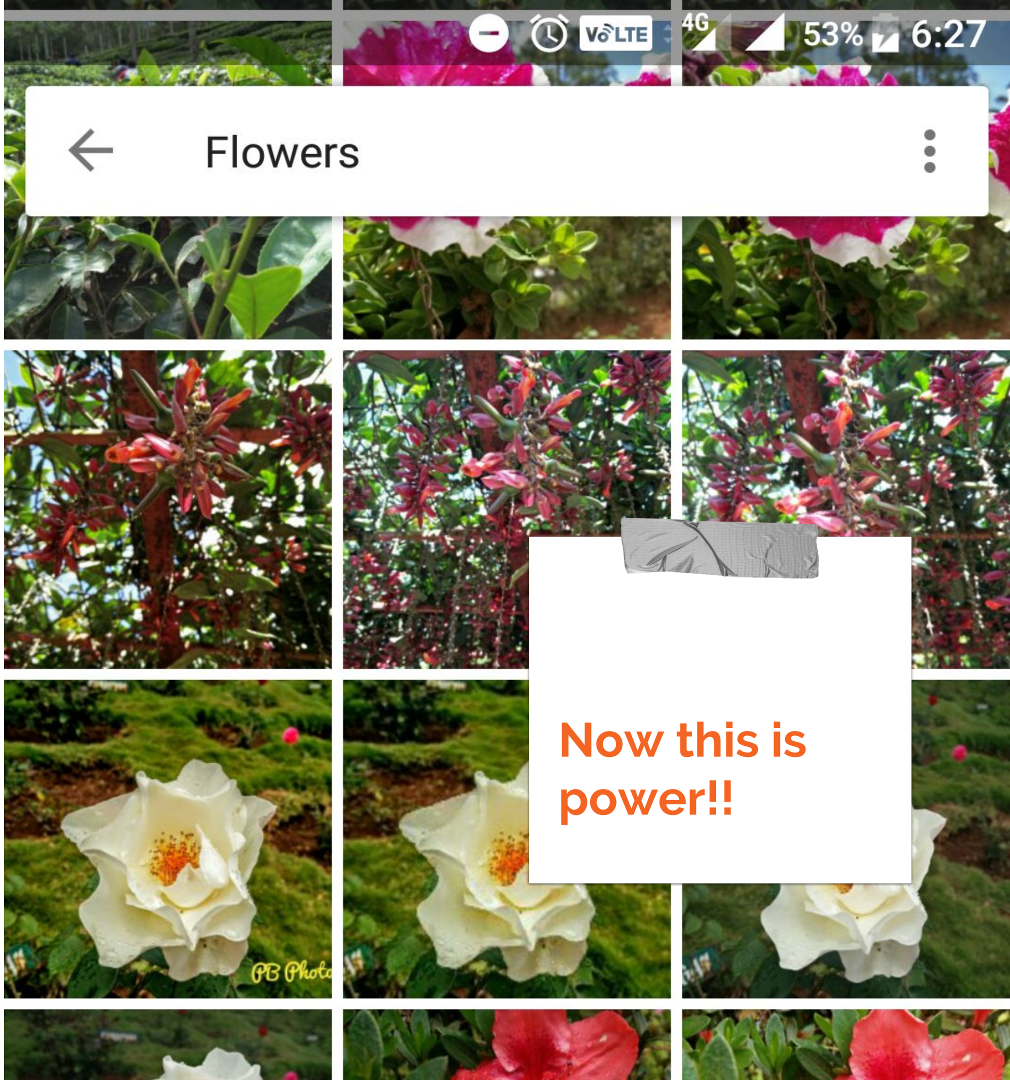
Info

“Go” is even more difficult to play than Chess (especially for a computer)

Google Photos

Search for something like “flowers” and it shows you all images you have ever clicked that has flowers :)

This is from my personal gallery :P



Now this is power!!

So, what is Machine Learning?

The crust of Linear Algebra stuffed with Multivariable Calculus

Cost function

Logistic regression:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Neural network:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

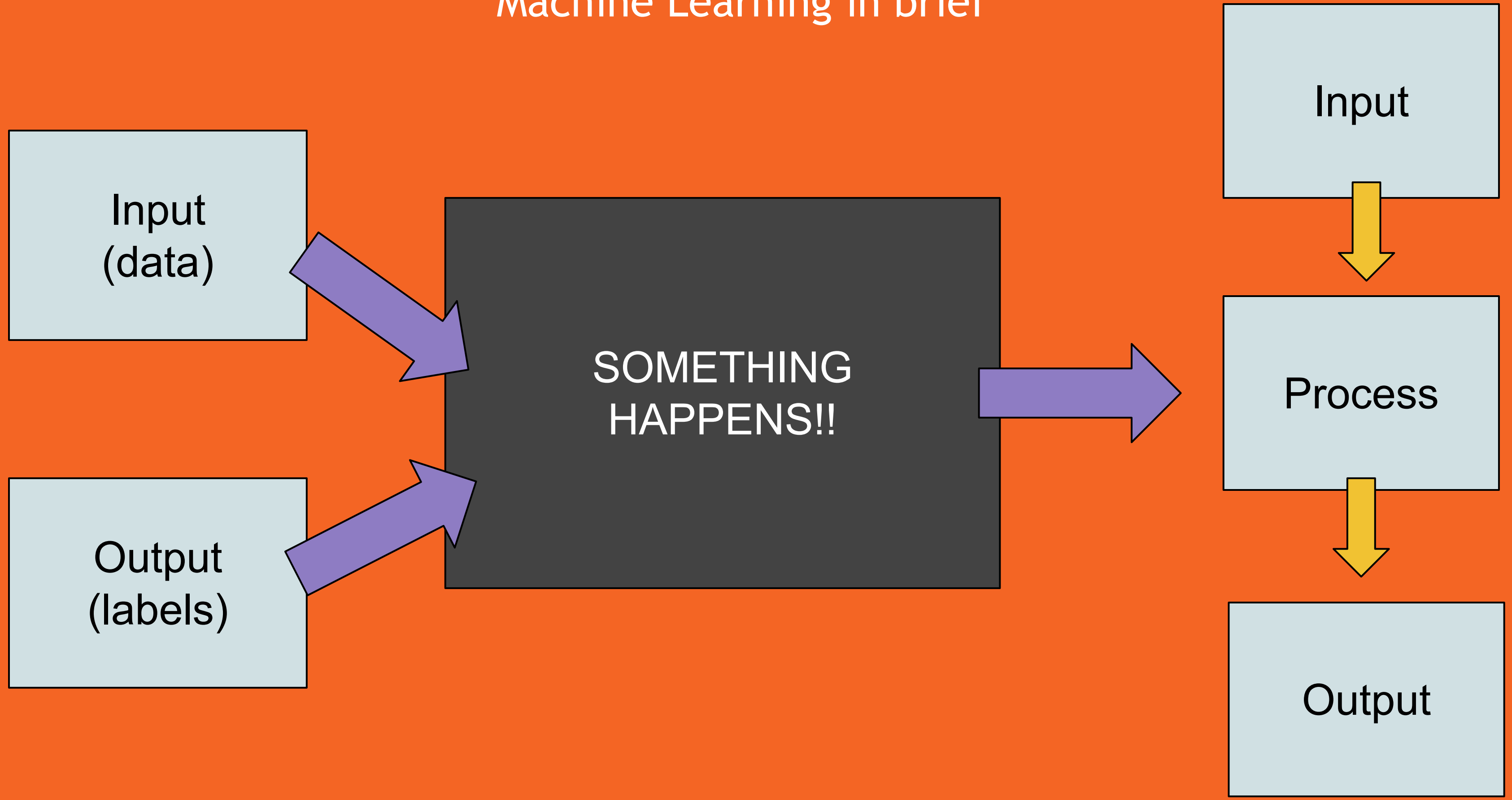
$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1-y_k^{(i)}) \log(1-(h_\Theta(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

Andrew Ng

operator	eigenvectors	eigenvalues
$\sigma_X \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}, \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$	± 1
$\sigma_Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\frac{1}{\sqrt{2}}\begin{pmatrix} -i \\ 1 \end{pmatrix}, \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -i \end{pmatrix}$	± 1
$\sigma_Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}$	± 1

Machine Learning in brief



I was just kidding...

You don’t need to be a Ph.D in Maths

Somebody who actually did Ph.D has done most of this stuff for us.

God bless Doctorates! :)

So, what is Tensorflow?

A library that does most of the math for you :)

Thank you Google :)

Machine learning is all about getting W's and b's

$$X * W + b = Y$$

Input *Weights* *Bias* *Output vector*

Find out error and tune your matrix accordingly

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 0 \\ 6 & 0 & 0 & 7 \end{bmatrix} \cdot \begin{bmatrix} 2 \\ 5 \\ 1 \\ 8 \end{bmatrix} = \begin{bmatrix} 4 \\ 47 \\ 5 \\ 68 \end{bmatrix} \qquad \begin{bmatrix} 4 \\ 81 \\ 5 \\ 68 \end{bmatrix}$$

Matrix initialized randomly

Input vector

Output vector

Actual Output vector

Is everything in numbers?

- In computers, everything can be represented by numbers
  - Images -> Matrix
  - Text -> ASCII Number format
- So calculations can be done to do interesting stuffs

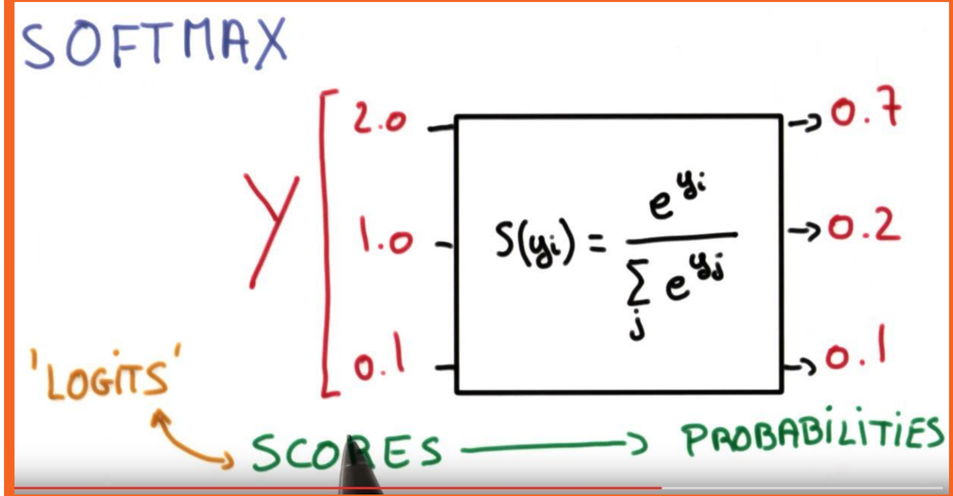
2. Writing in Tensorflow

```
import tensorflow as tf

sess = tf.Session()

x = tf.placeholder(tf.float32,[4, 1])
W = tf.Variable(tf.zeros([4,4])

Y = tf.matmul(W, x)

sess.run(
 tf.global_variables_initializer())

sess.run(Y, feed_dict = {x: <someInput>})
```

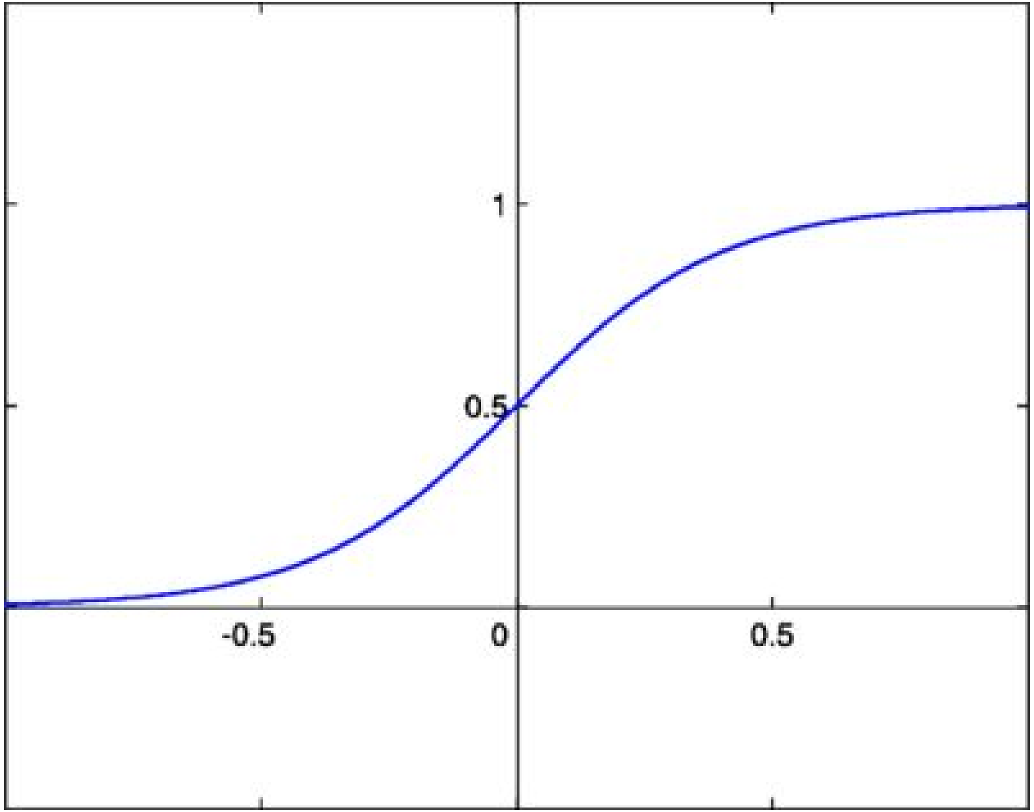
You can also randomize your weights...

```
import tensorflow as tf

... Session and x declarations...

W =
tf.Variable(tf.truncated_normal([4,4],
stddev=0.1))

... The other commands ...
```

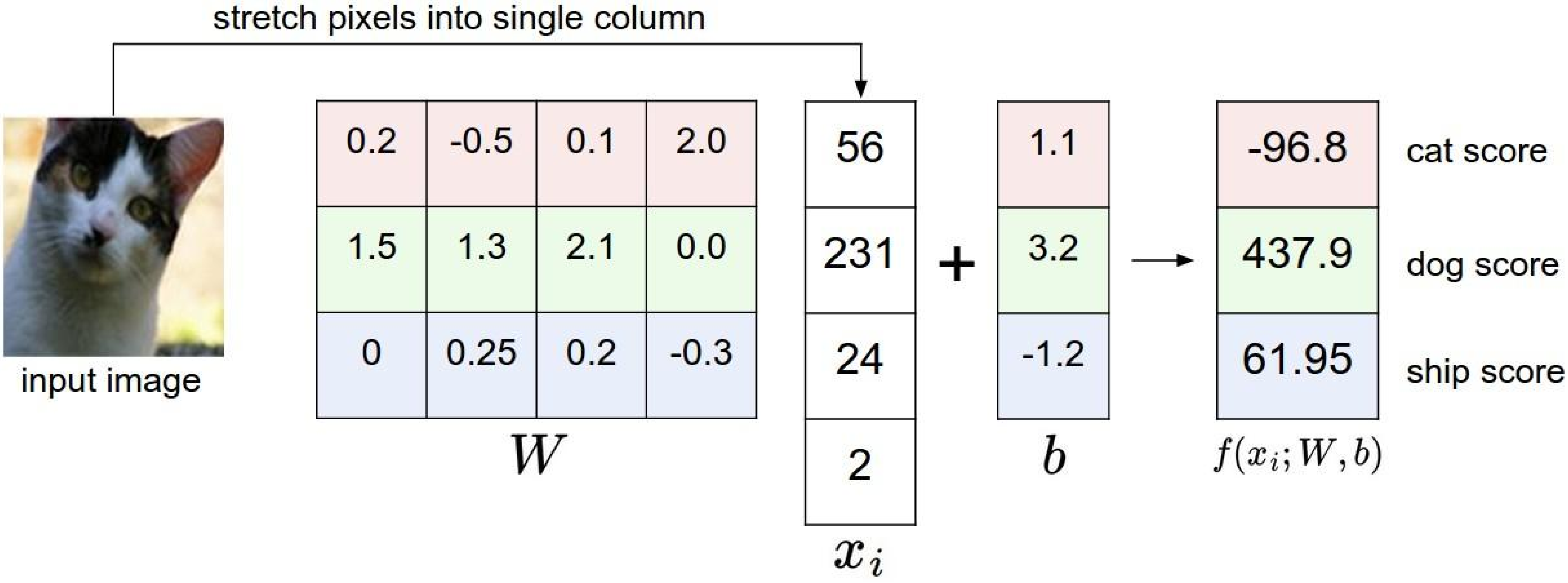
Some more theory...



Softmax Curve... (We all love curves, don't we?)



So how everything stacks up...



2. Writing the Softmax

```
import tensorflow as tf

...Session and variable declarations...

Y = tf.nn.softmax(tf.matmul(W, x))

...Session runs...
```

Now let me tell you how bad your model is...

Cross-Entropy

$$H_{y'}(y) = -\sum_i y'_i \log(y_i)$$

In TensorFlow

```
import tensorflow as tf

...Session and variable declarations...

...Session runs …

actual_y = tf.placeholder(tf.float32,
[None, 10])

cross_entropy = tf.reduce_mean(
     -tf.reduce_sum(actual_y * log(Y),
     reduction_indices=[1]))
```

Now lets see how that tuning of matrix works...

Training your model...

```
import tensorflow as tf

... All other code ...

train_step =
tf.train.GradientDescentOptimizer(0.5).
minimize(cross_entropy)
```

Training your model...

```
import tensorflow as tf

... All other code ...

train_step =
tf.train.GradientDescentOptimizer(0.5).
minimize(cross_entropy)
```

Compare the left

- **Learning rate**
  How fast it should learn from data. High value means it can generalize properly. On the left, it is **0.5**
- **Idea is to minimize the loss (cross_entropy)**
  Of course, it is the loss. You always minimize loss.
- **This will not give any output**
  Needs to be run in a session. Will do in sometime. Just Wait!!!

Evaluation of models??

-- Find the accuracy

Finding your accuracy...

```
import tensorflow as tf

... All other code ...

correct_prediction=
     tf.equal(tf.argmax(Y,1),
          tf.argmax(actual_y, 1))

accuracy = tf.reduce_mean(tf.cast(
     correct_prediction, tf.float32))
```

Finally...

```
import tensorflow as tf

... Finally all that code ...

print(sess.run(accuracy,
feed_dict={x:<input>,actual_y:<actual
output>}
```

Yayyyy!!!

You built your first Machine Learning Model.

Congratulations :D

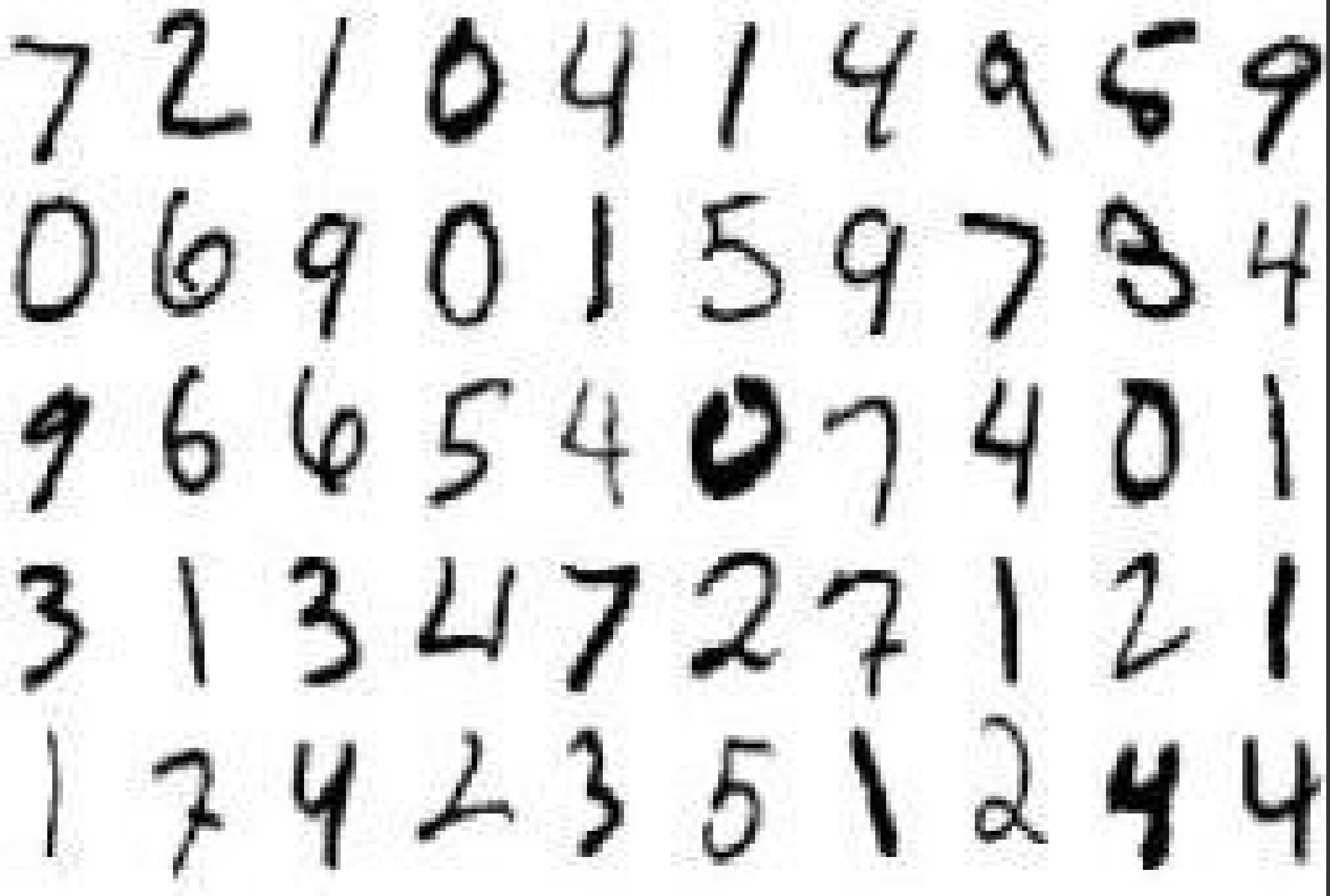
Ofcourse, you didn't run anything yet !

Now, let's dive into an actual ML problem!

Info

There are several ML problems. Out of those, MNIST Character Recognition is considered like a "hello world"

MNIST Character Recognition



Something about MNIST

➔ **Input image size**

28 by 28 image (Size of x is 784)

➔ **Number of output labels**

Ten (from 0 to 9)

➔ **It will be a black and white image**

In RGB, number of channels is 3, in white, number of channels is 1

Lets download the mnist dataset

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
```

Now as usual, the variables

```
import tensorflow as tf

... Code for downloading data …

x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))

y = tf.nn.softmax(tf.matmul(x, W) + b)

actual_y = tf.placeholder(tf.float32, [None, 10])
```

Writing the cross-entropy and train_step

```
import tensorflow as tf

... Code for downloading data ...

... Variable Initializations ...

cross_entropy = tf.reduce_mean(-tf.reduce_sum(actual_y *
tf.log(y), reduction_indices=[1]))

train_step =
tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

Finding accuracy

```
import tensorflow as tf

... Code for downloading data ...

... Variable Initializations ...

... Code for cross-entropy and training step …

correct_prediction = tf.equal(tf.argmax(y,1),
tf.argmax(actual_y, 1))

accuracy = tf.reduce_mean(tf.cast(correct_prediction,
tf.float32))
```

That finished our computational graph

Initializing the session

```
import tensorflow as tf

... All the previous code ...

sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

Finally training ....

```
import tensorflow as tf

... All the previous code ...

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(1000):
    batch_X, batch_Y = mnist.train.next_batch(100)
    train_data = {x: batch_X, actual_y: batch_Y}

    sess.run(train_step, feed_dict = train_data)

    a = (sess.run([accuracy], feed_dict = train_data))
    if i % 100 == 0:
        print ("Step %d, Accuracy %g")%(i, a)
```

We finished our training!!!

Hurrah!

God bless us!

Hopefully, test will be good ;)

Testing ...

```
import tensorflow as tf

... All the previous code ...

sess = tf.Session()
sess.run(tf.global_variables_initializer())

... Training the model ...

test_data = {x: mnist.test.images, actual_y: mnist.test.labels}
a = (sess.run([accuracy], feed_dict=test_data))
print (“Step %d, Accuracy %g”)%(i, a)
```

That's it!

You have learnt Machine Learning :D

Some cool experiments!
Let's play!

(https://aiexperiments.withgoogle.com/)


Thank you