

# TensorFlow and Neural networks

**ML** *Meetup*

+ Meetups



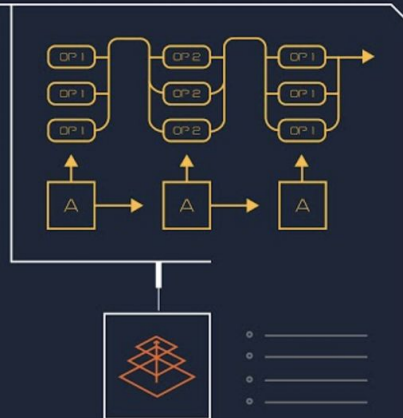
# Content!

1. Tensorflow basis
2. Logistic regression
3. Fully connected layers
4. Convolutional layers
5. Keras
6. Other approaches





# TensorFlow





Slides and code available  
at

[https://github.com/jcvasquezc/TF\\_pereira](https://github.com/jcvasquezc/TF_pereira)



# TensorFlow basis

## Setup

```
pip install tensorflow
```

If does not work:

```
sudo apt-get install python-pip python-dev python-virtualenv
```

```
pip install tensorflow
```



- CPUs
- GPUs
- desktop,
- Server
- Mobile device with a single API.

# TensorFlow basis

## Setup

```
import tensorflow as  
  
hello = tf.constant('Hello, TensorFlow!')  
sess = tf.Session()  
print(sess.run(hello))
```



- ⊙ CPUs
- ⊙ GPUs
- ⊙ desktop,  
Server
- ⊙ Mobile device with a single API.

# TensorFlow basis

- ◎ Open source software library for numerical computation using data flow graphs.
  - Nodes = represent mathematical operations,
  - tensors = The central unit of data.

```
node1 = tf.constant(3.0, dtype=tf.float32)
node2 = tf.constant(4.0) # also tf.float32 implicitly
```

```
sess = tf.Session()
node3 = tf.add(node1, node2)
sess.run(node3)
```



- ◎ CPUs
- ◎ GPUs
- ◎ desktop,
- ◎ Server
- ◎ Mobile device with a single API.



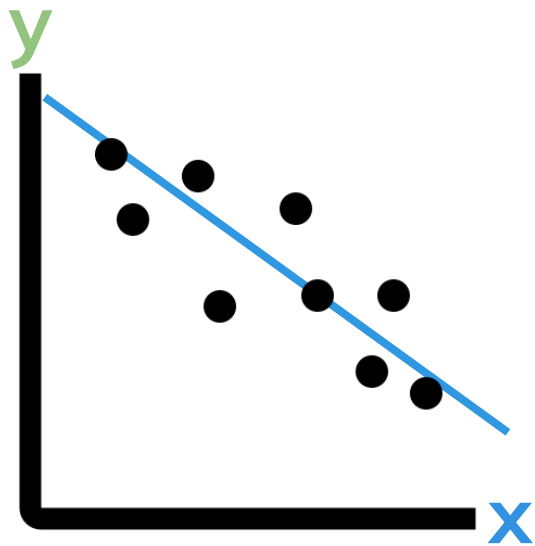
# Demo 1



# Linear model

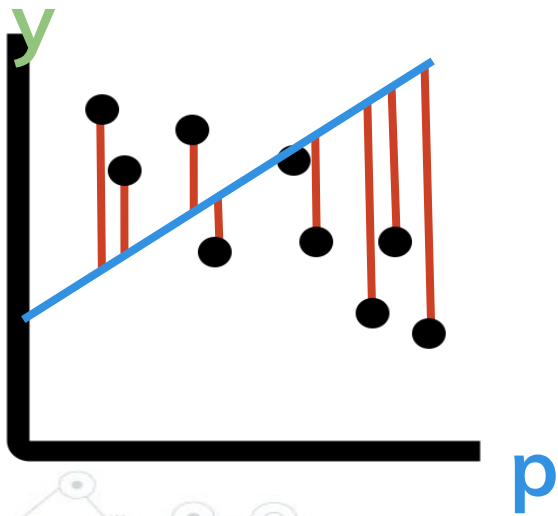
$$y=W^*x+b$$

## Regression



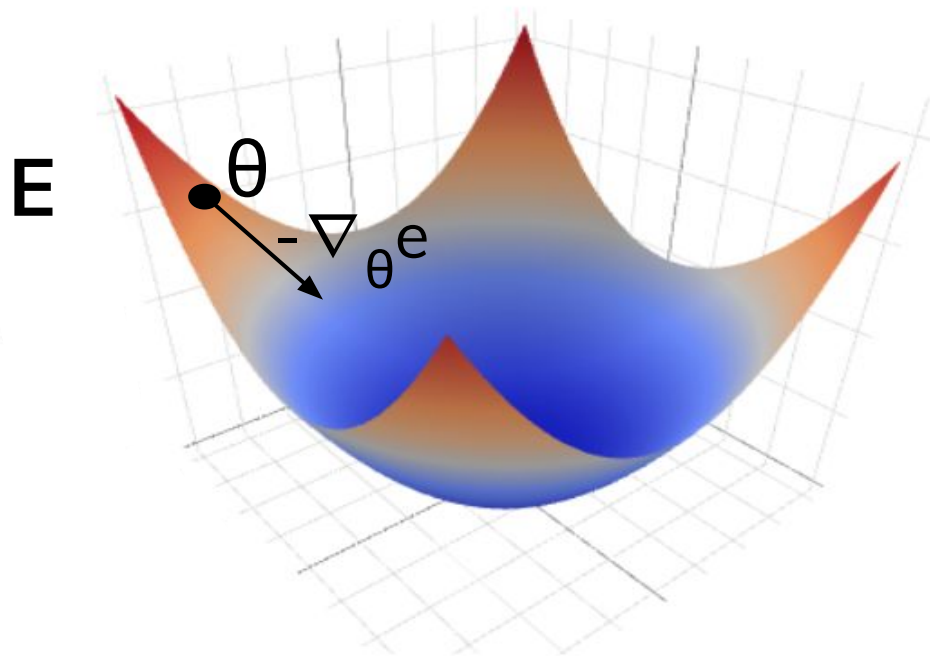
## How to find $W$ and $b$ ?

Error function: Mean square error

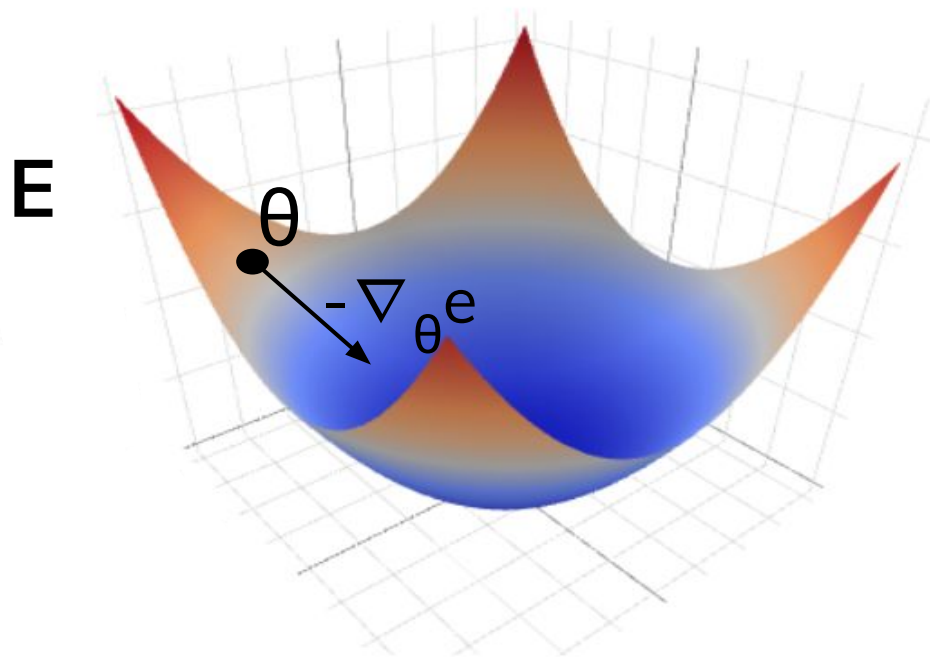


$$E = \frac{\sum (p - y)^2}{N}$$

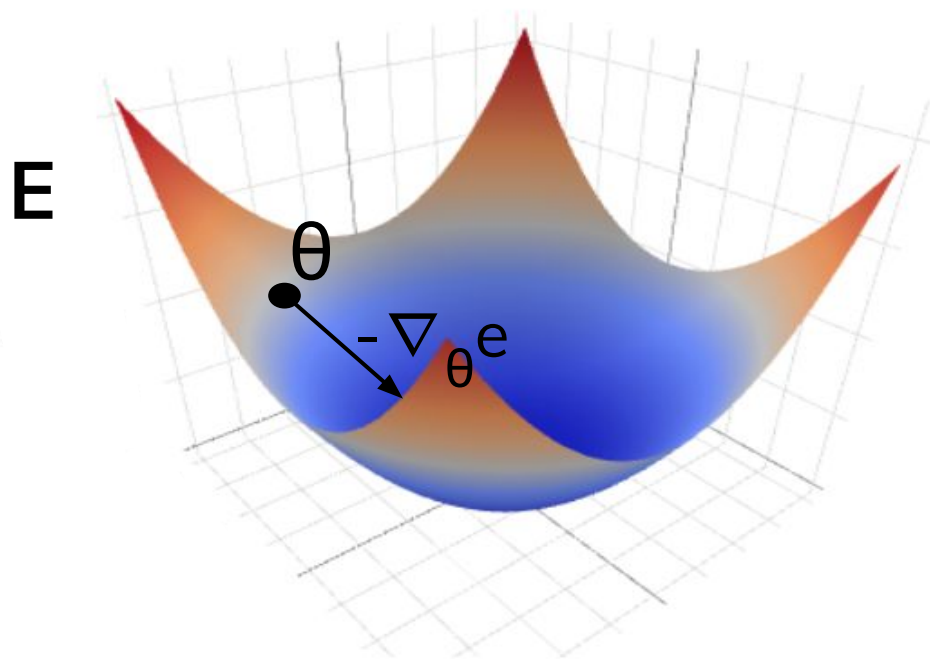
How to find  $\mathbf{W}$  and  $\mathbf{b}$ ?



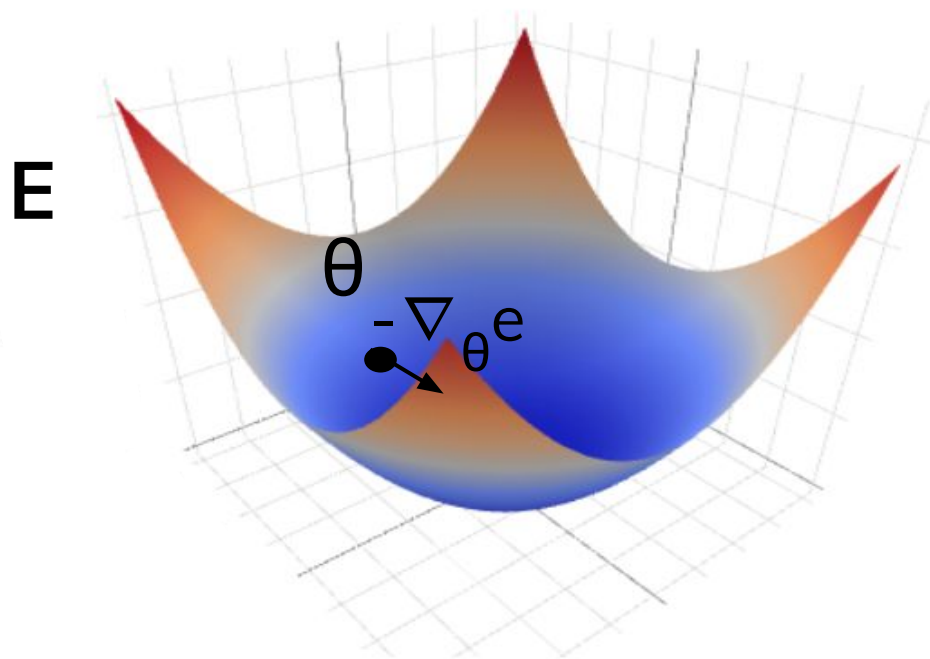
How to find  $\mathbf{W}$  and  $\mathbf{b}$ ?



How to find  $\mathbf{W}$  and  $\mathbf{b}$ ?

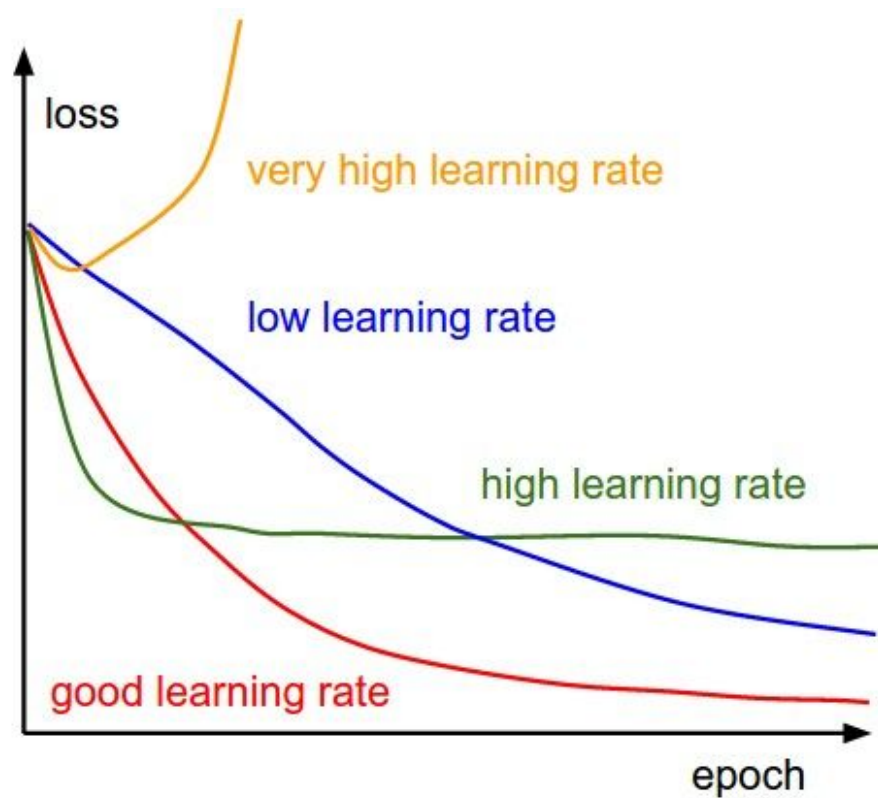


How to find  $W$  and  $b$ ?



And so on...

## How to find $W$ and $b$ ?



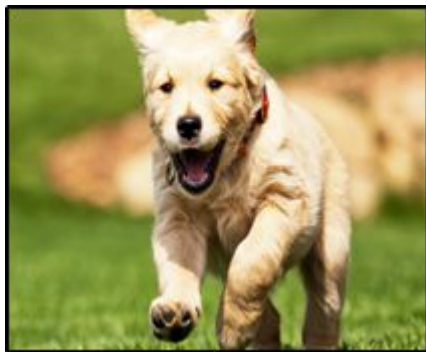
# Demo 2



# Logistic regression

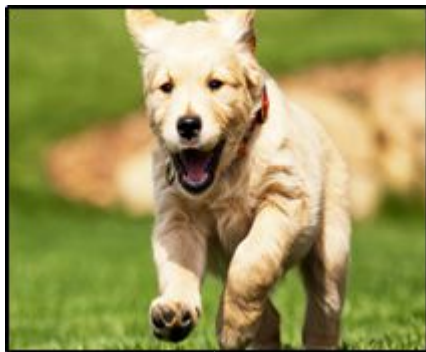
$$\mathbf{W} \cdot \mathbf{X} + b = y$$

# Logistic regression



$$W \cdot X + b = y$$

# Logistic regression

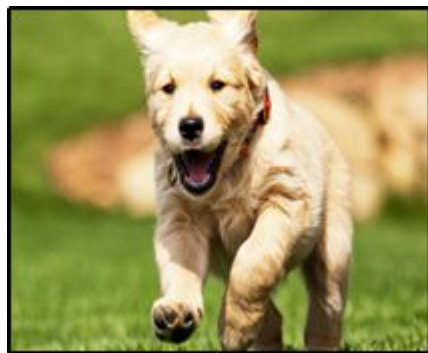


$$W \cdot X + b = y$$

golden

chinese

# Logistic regression



$$W \cdot X + b = y$$

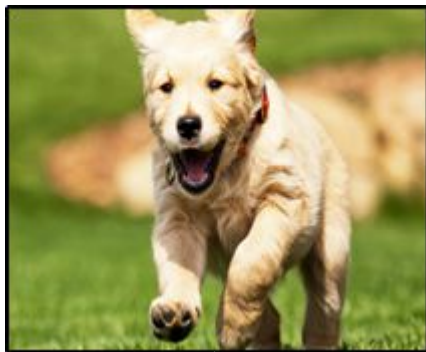
Weights

bias

golden

chinese

# Logistic regression

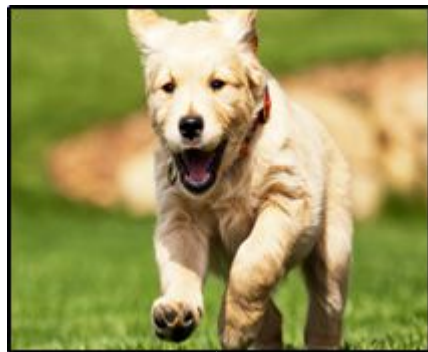


$$W \cdot X + b = y = \begin{bmatrix} 2.0 \\ 0.5 \end{bmatrix}$$

golden

chinese

# Logistic regression



$$W \cdot X + b = y = \begin{bmatrix} 2.0 \\ 0.5 \end{bmatrix}$$

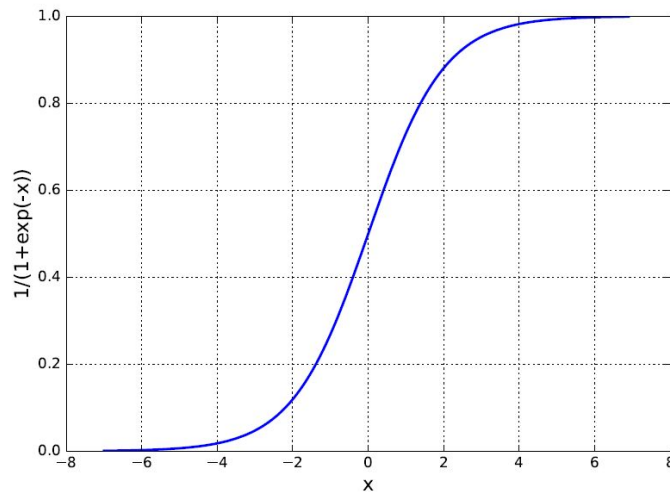
golden ✓  
 $P(X) = 0.99$

chinese ✗  
 $P(X) = 0.05$

# Logistic regression

$$y = \begin{bmatrix} 2.0 \\ 0.1 \end{bmatrix}$$

$$p(y) = \frac{1}{1 + e^{(-y)}}$$



$$p = \begin{bmatrix} 0.99 \\ 0.05 \end{bmatrix}$$

# Logistic regression

- + Easy to train
- + Free adjustable hyper-parameters
- Could be very simple
- Only for linearly-separable classes



## How to find $W$ and $b$ ?

Answer:

Gradient descendent algorithm

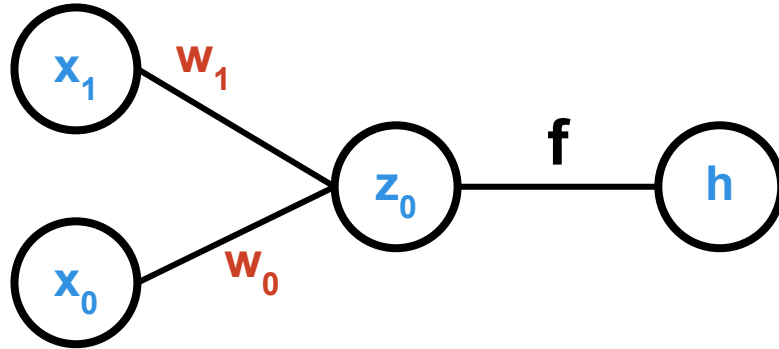
1.  $p = f(x)$  #logistic function
2.  $e = E(p, y; \theta)$  #get error
3.  $\Delta = \nabla_{\theta} e$  #error's gradient / derivative
4.  $\theta := \theta - \alpha \Delta$  #lower error moving against gradient
5. repeat



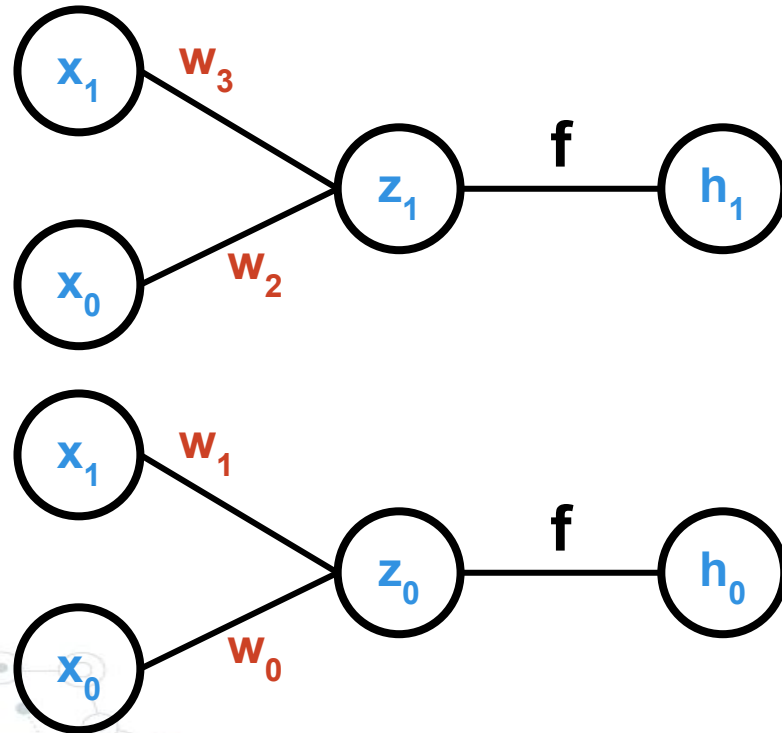
# Demo 3

# Neural Networks

$$h = f(w x + b)$$

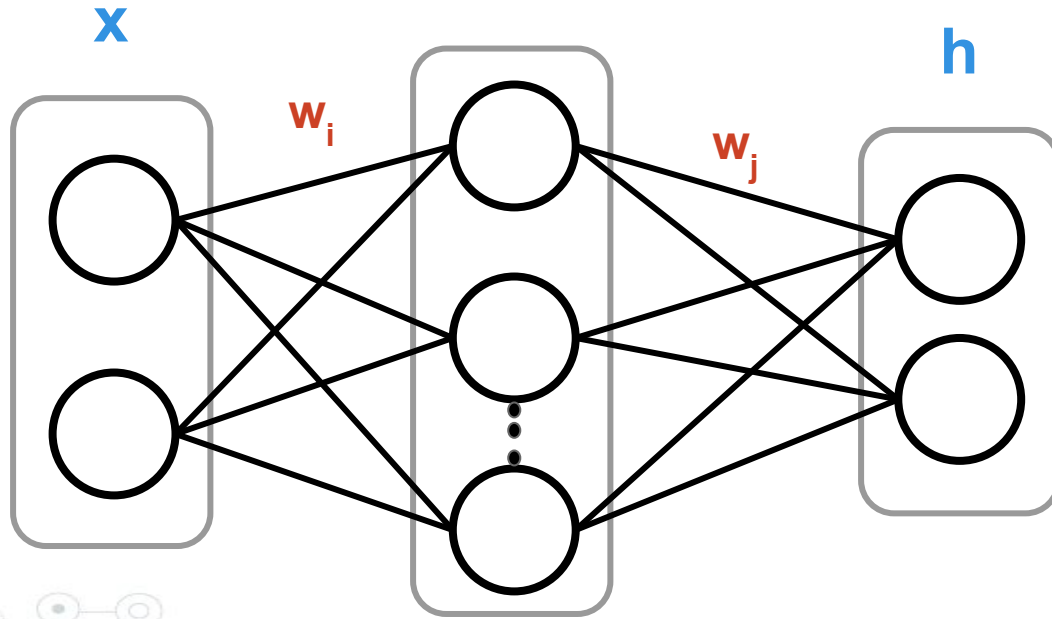


# Neural Networks

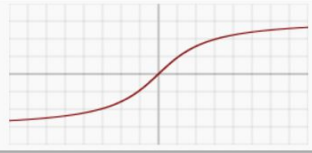
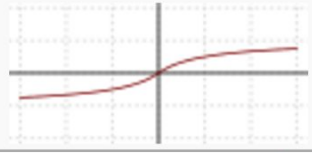
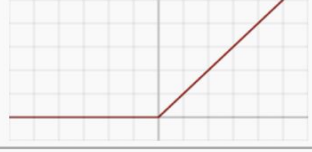



## Neural Networks

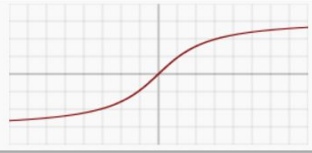
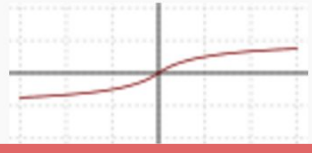


$$h = f(\mathbf{x}; \theta)$$



# Neural Networks

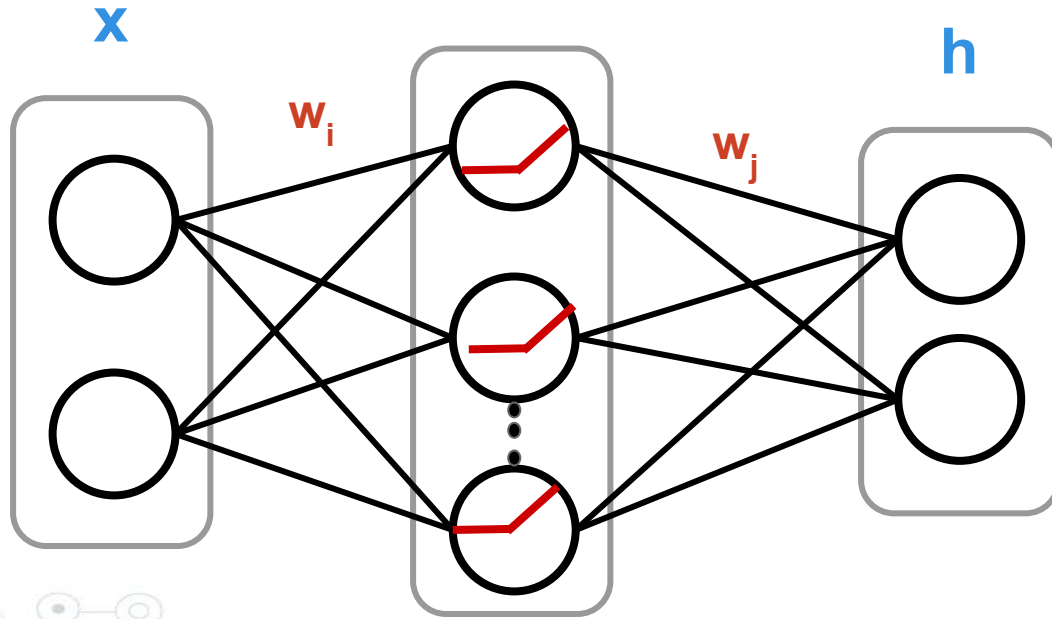
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign <a href="#">[7]</a> <a href="#">[8]</a>		$f(x) = \frac{x}{1 +  x }$
Rectified Linear Unit (ReLU) <a href="#">[9]</a>		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) <a href="#">[10]</a>		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

# Neural Networks

ArcTan		$f(x) = \tan^{-1}(x)$
Softsign <sup>[7] [8]</sup>		$f(x) = \frac{x}{1 +  x }$
Rectified Linear Unit (ReLU) <sup>[9]</sup>		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) <sup>[10]</sup>		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

## Neural Networks

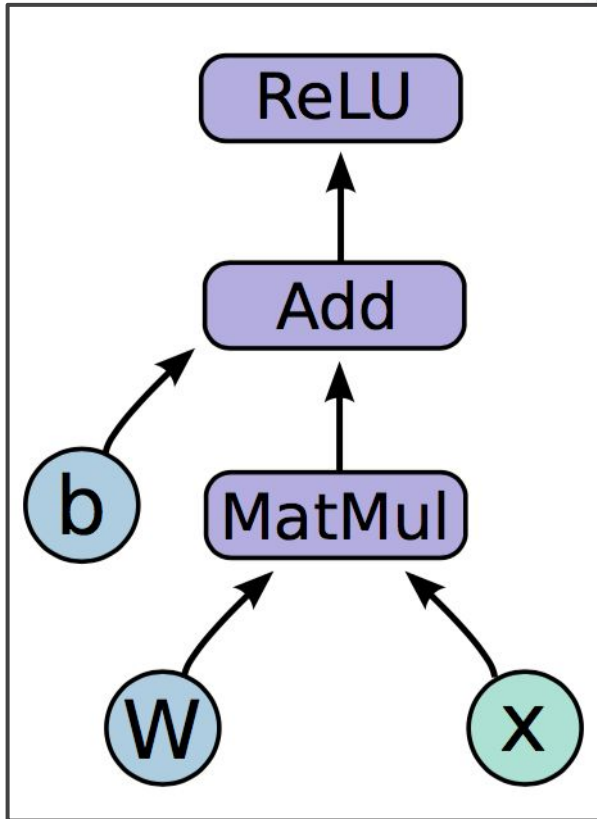
$$h = f(\mathbf{x}; \theta)$$





## Neural networks

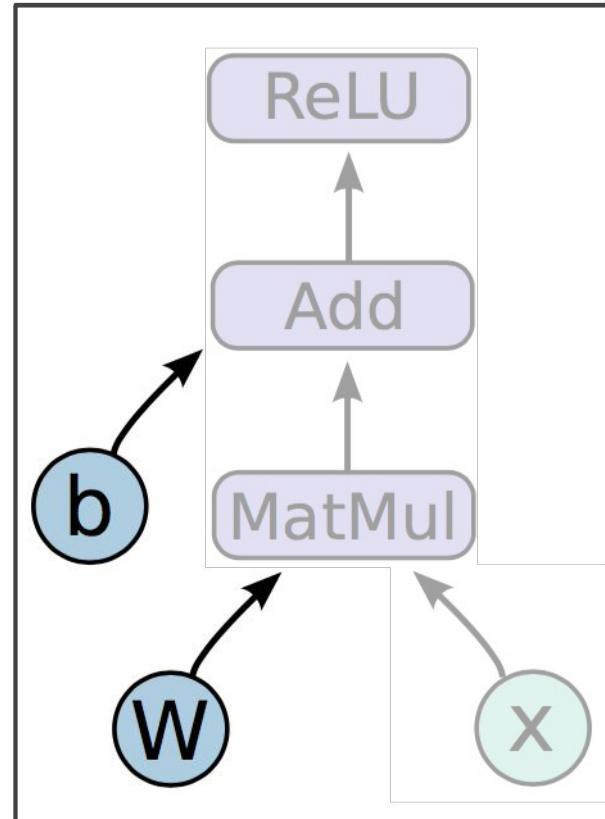
$$h_i = \text{ReLU}(Wx + b)$$



## Neural networks

$$h_i = \text{ReLU}(Wx + b)$$

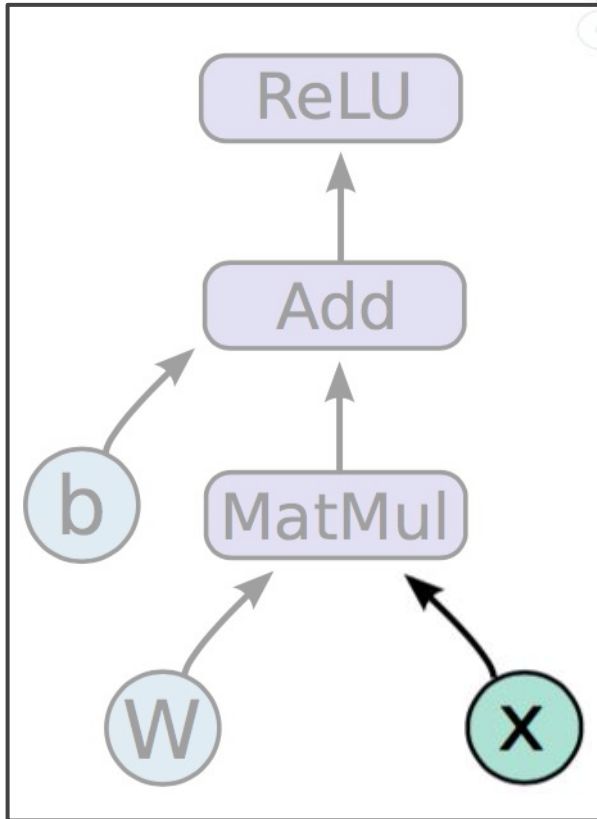
Parameters to fit  
(Variables)



## Neural networks

$$h_i = \text{ReLU}(Wx + b)$$

Input data  
(Placeholder)

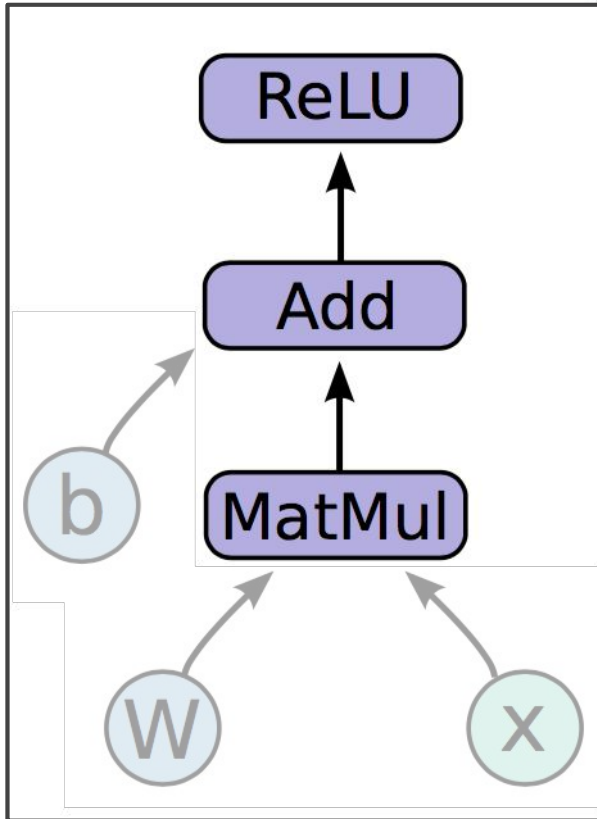


## Neural networks

$$h_i = \text{ReLU}(Wx + b)$$

Mathematical  
operations:

MatMul, Add, ReLU



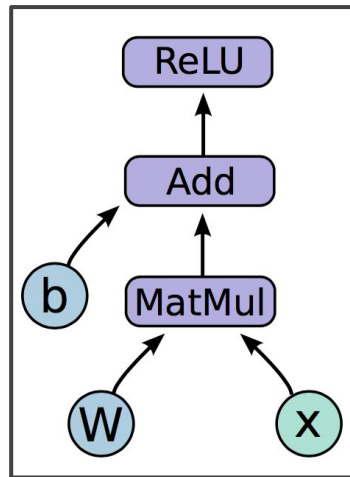
# Neural networks

In code, please!

```
import tensorflow as tf

1  b = tf.Variable(tf.zeros((10,)))  
   W = tf.Variable(tf.random_uniform((784, 10), -1, 1))  
  
2  x = tf.placeholder(tf.float32, (None, 784))  
3  h_i = tf.nn.relu(tf.matmul(x, W) + b)
```

$$h_i = \text{ReLU}(Wx + b)$$



# Neural networks

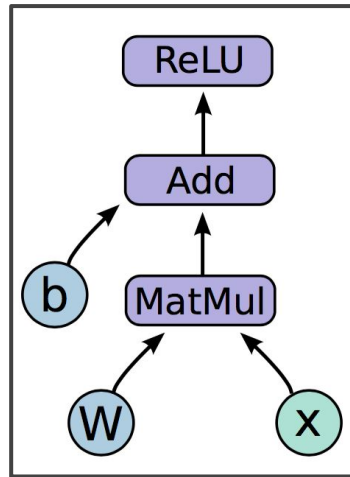
## How to run it?

$$h_i = \text{ReLU}(Wx + b)$$

```
import tensorflow as tf
```

```
1 b = tf.Variable(tf.zeros((10,)))  
  W = tf.Variable(tf.random_uniform((784, 10), -1, 1))  
  
2 x = tf.placeholder(tf.float32, (None, 784))  
3 h_i = tf.nn.relu(tf.matmul(x, W) + b)
```

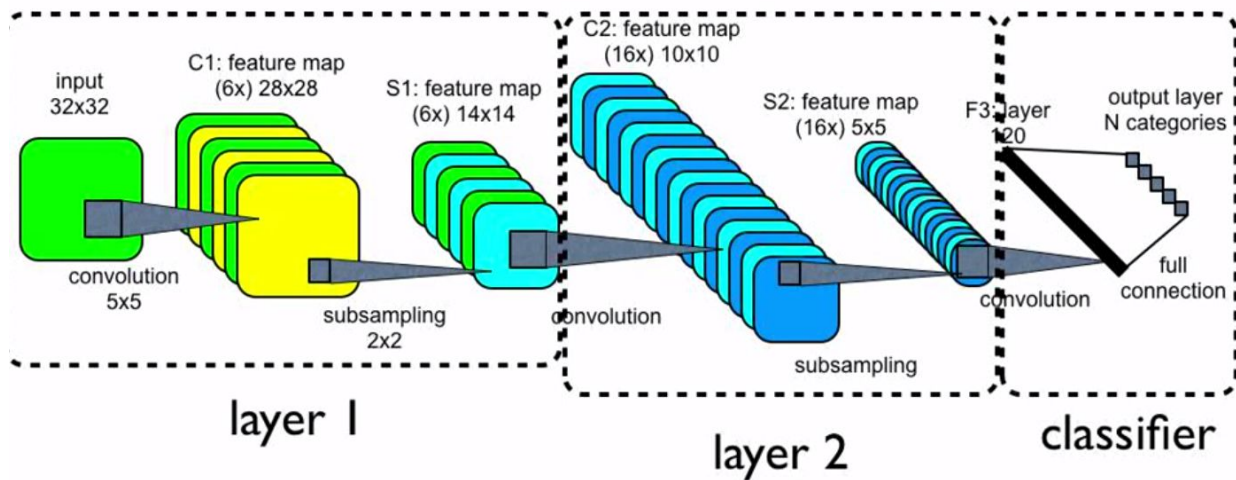
```
sess = tf.Session()  
sess.run(tf.initialize_all_variables())  
sess.run(h_i, {x: np.random.random(64, 784)})
```



# Demo 4

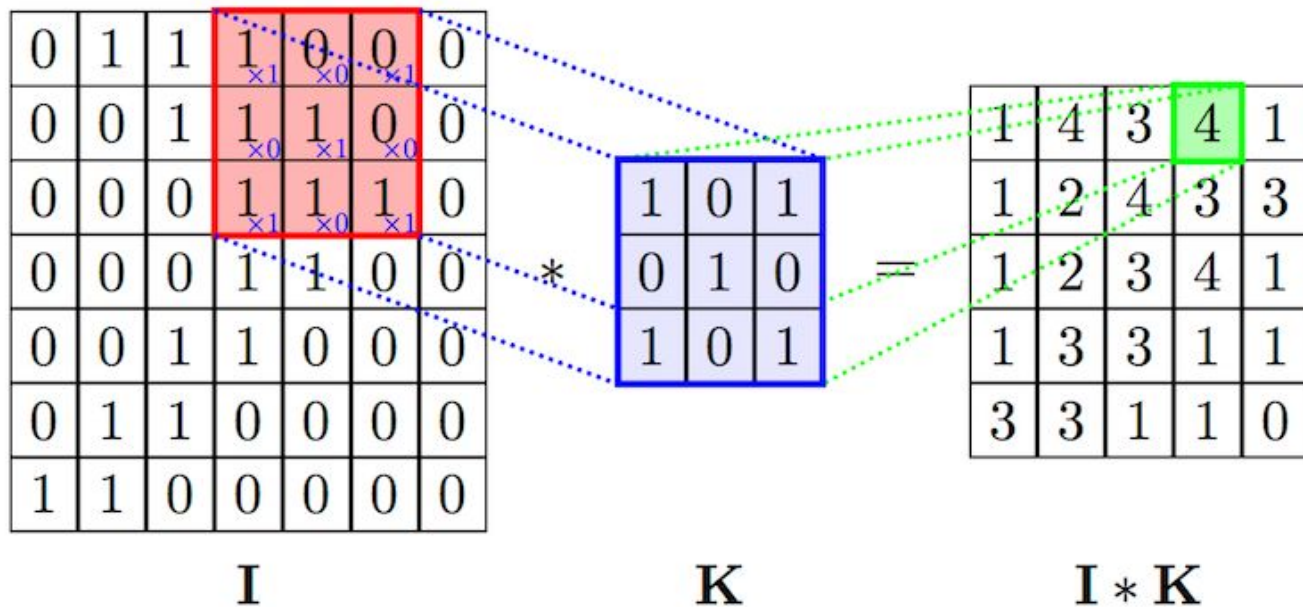
# Convolutional layers

## Convolutional Neural Networks



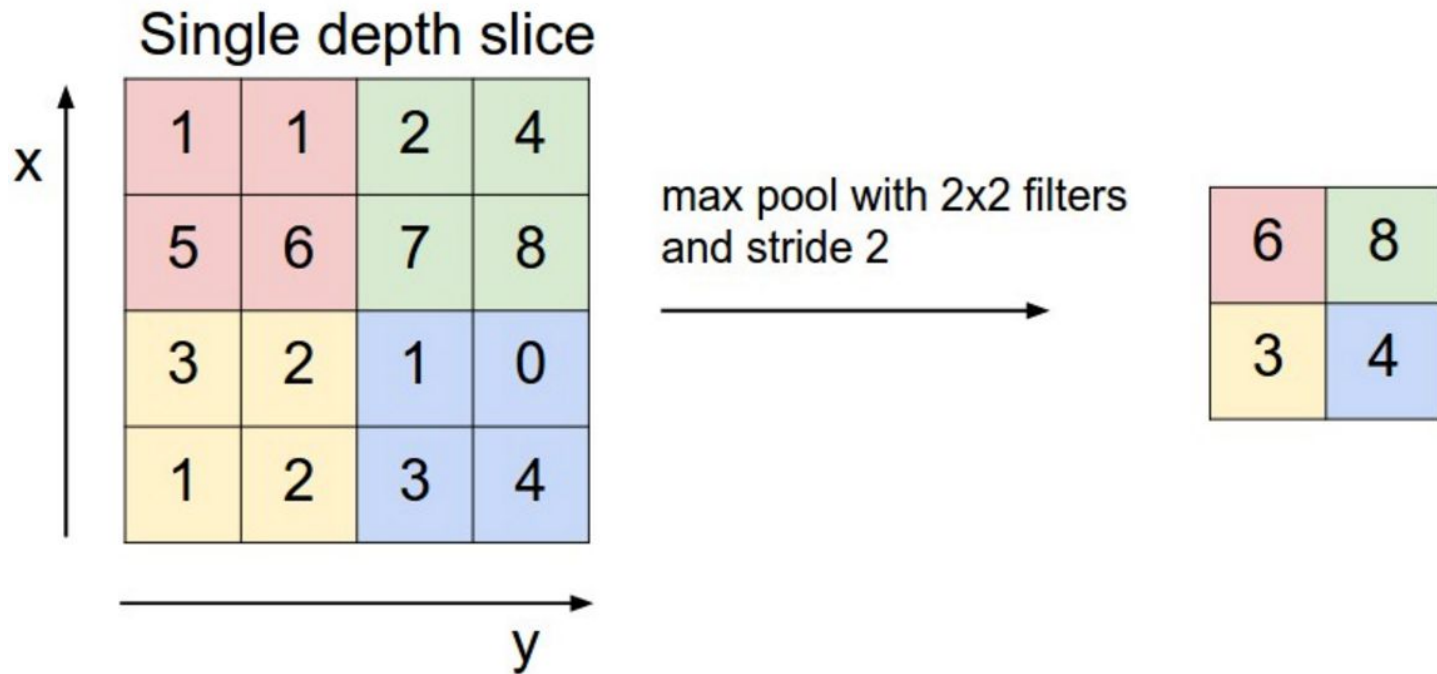


# Convolutional layers



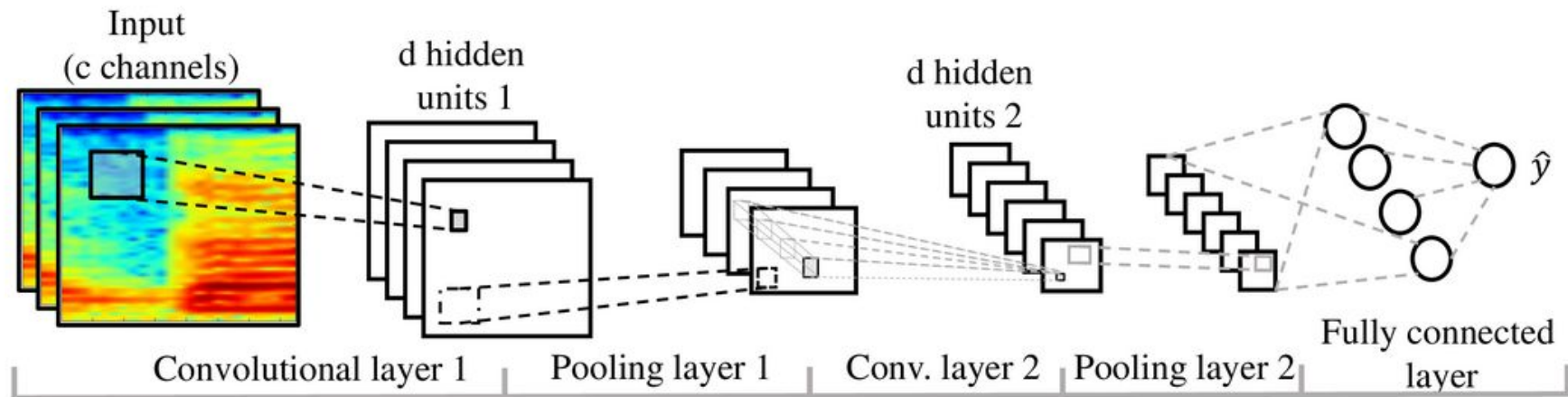
```
d=tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')
```

# Pooling layers



```
tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
```

# Convolutional neural network





# Demo 5

CNN in Keras  
<http://keras.io>



Keras

# Thanks

Twitter: @ **@jcvasquezc1**

LinkedIn: @**jcvasquezc**

Email: [jcamillo.vasquez@udea.edu.co](mailto:jcamilo.vasquez@udea.edu.co)

<http://jcvasquezc.wixsite.com/home>



# Tensorflow and Neural Networks

**ML** *Meetup*

+ Meetups

