# TensorFlow

ML Meetup

+ Meetups

# Hello!

## I am Camilo Vásquez

You can find me at:

Twitter: @ **@jcvasquezc1**

LinkedIn: @**jcvasquezc**

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

[http://jcvasquezc.wixsite.com/home](http://jcvasquezc.wixsite.com/home)

# Hello!

## I am Camilo Vásquez

- Lecturer and machine learning researcher at UdeA

UNIVERSIDAD DE ANTIOQUIA
1803

GITA

You can find me at:

Twitter: @ **@jcvasquezc1**

LinkedIn: @**jcvasquezc**

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

[http://jcvasquezc.wixsite.com/home](http://jcvasquezc.wixsite.com/home)

# Hello!

## I am Camilo Vásquez

- PhD(c) Pattern recognition Lab. University of Erlangen-Nuremberg.

FAU
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

LEHRSTUHL FÜR
MUSTER-
ERKENNUNG

You can find me at:

Twitter: @ **@jcvasquezc1**

LinkedIn: @**jcvasquezc**

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

[http://jcvasquezc.wixsite.com/home](http://jcvasquezc.wixsite.com/home)

# Hello!

## I am Camilo Vásquez

- Machine learning engineer and software developer at Brandon Logan

Brandon Logan™

Inspired By data.

You can find me at:

Twitter: @ **@jcvasquezc1**

LinkedIn: @**jcvasquezc**

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

[http://jcvasquezc.wixsite.com/home](http://jcvasquezc.wixsite.com/home)

# Content!

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

OP 1
OP 1
OP 1
OP 2
OP 2
OP 2
OP 1
OP 1
OP 1
A
A
A
TensorFlow

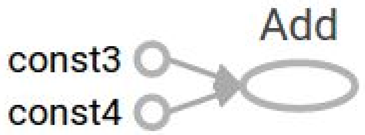
# Slides and code available at

https://github.com/jcvasquezc/rockstarTF

# TensorFlow basis

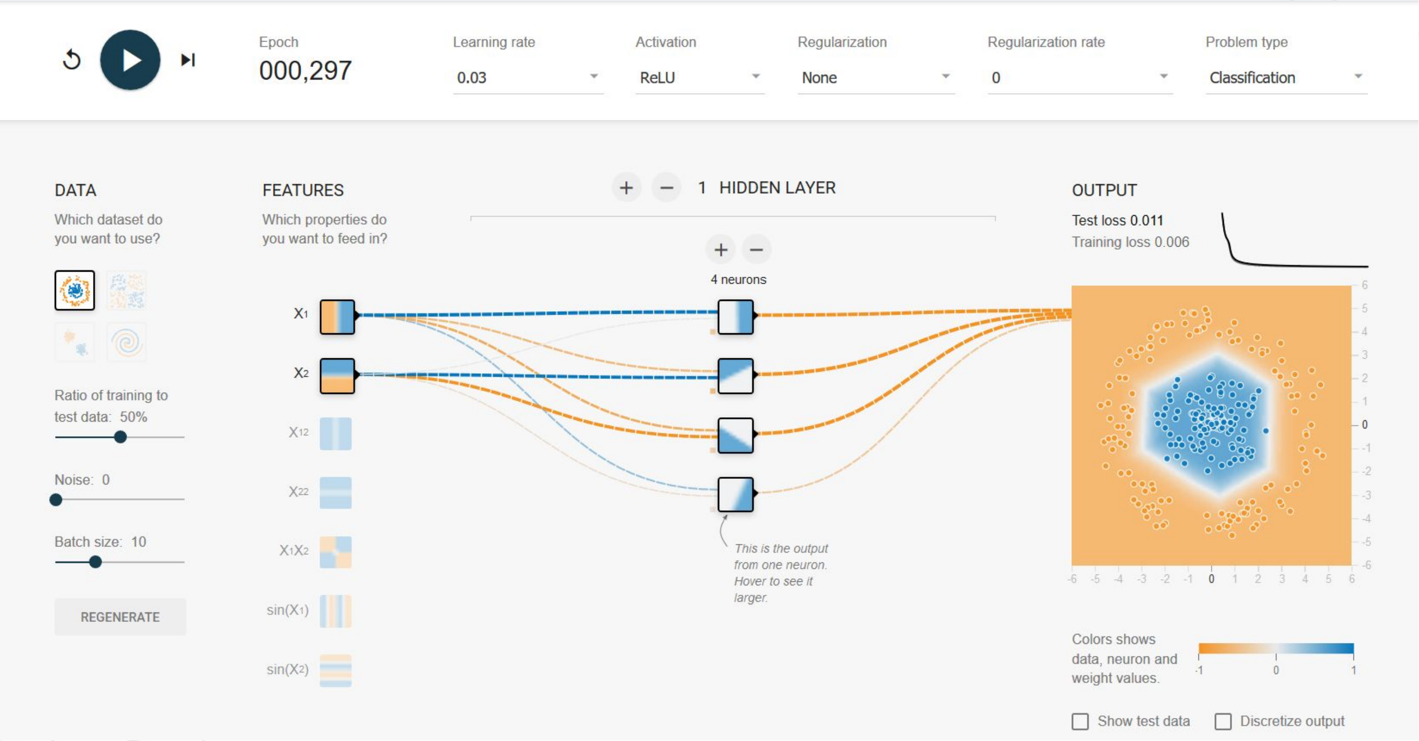
- Open source software library for numerical computation using data flow graphs.
  - Nodes = represent mathematical operations,
  - tensors = The central unit of data.

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

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



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

# TensorFlow (http://playground.tensorflow.org)



Epoch 000,297

Learning rate 0.03

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

1 HIDDEN LAYER

4 neurons

This is the output from one neuron. Hover to see it larger.

OUTPUT

Test loss 0.011

Training loss 0.006

Colors shows data, neuron and weight values.

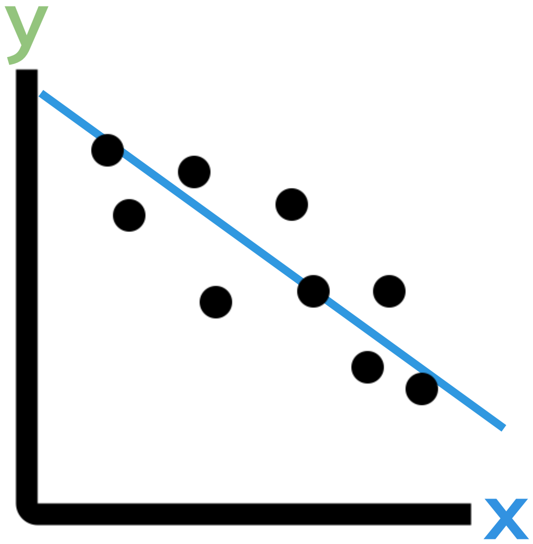
□ Show test data □ Discretize output

# Demo 1

# Linear model

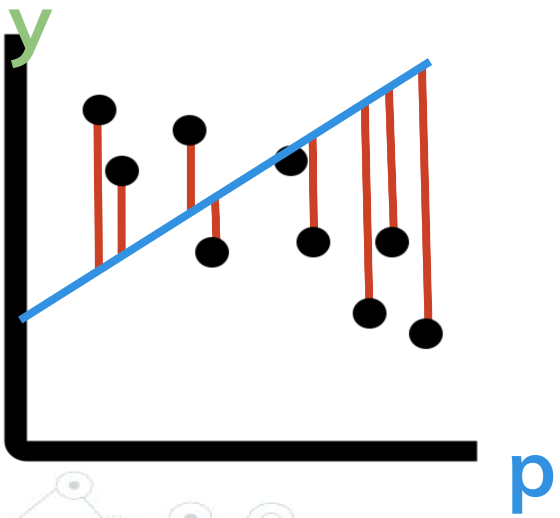
$y=W^*x+b$

## Regression



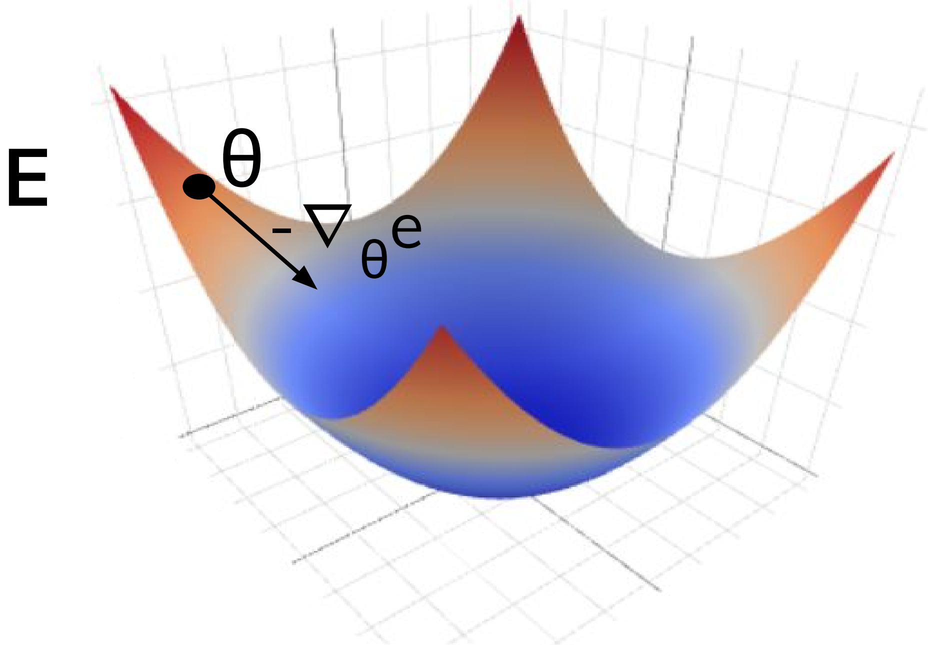
# How to find ***W*** and ***b***?

Error function: Mean square error

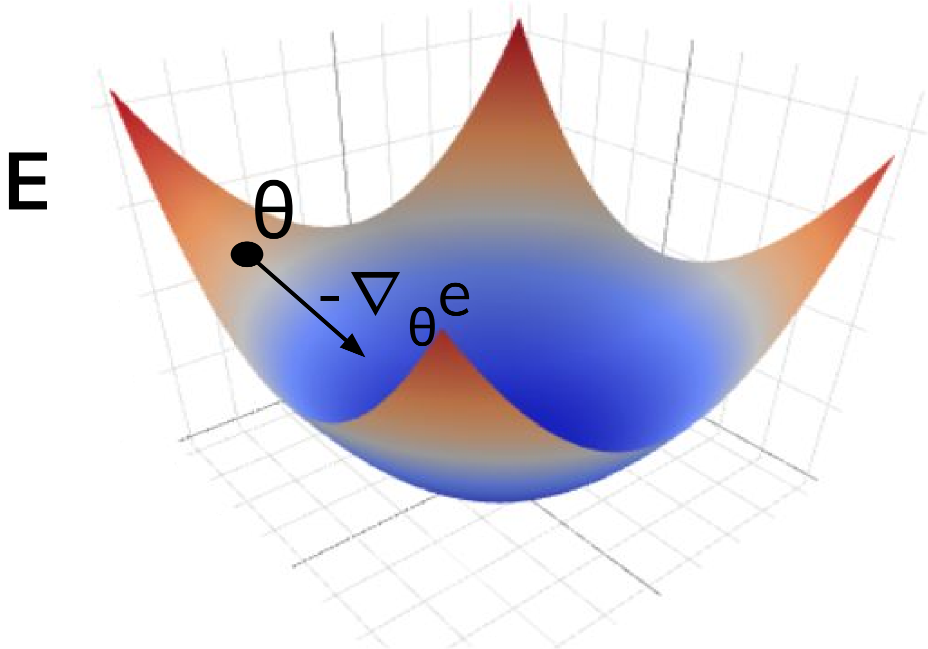


$$E = \frac{\sum (p - y)^2}{N}$$

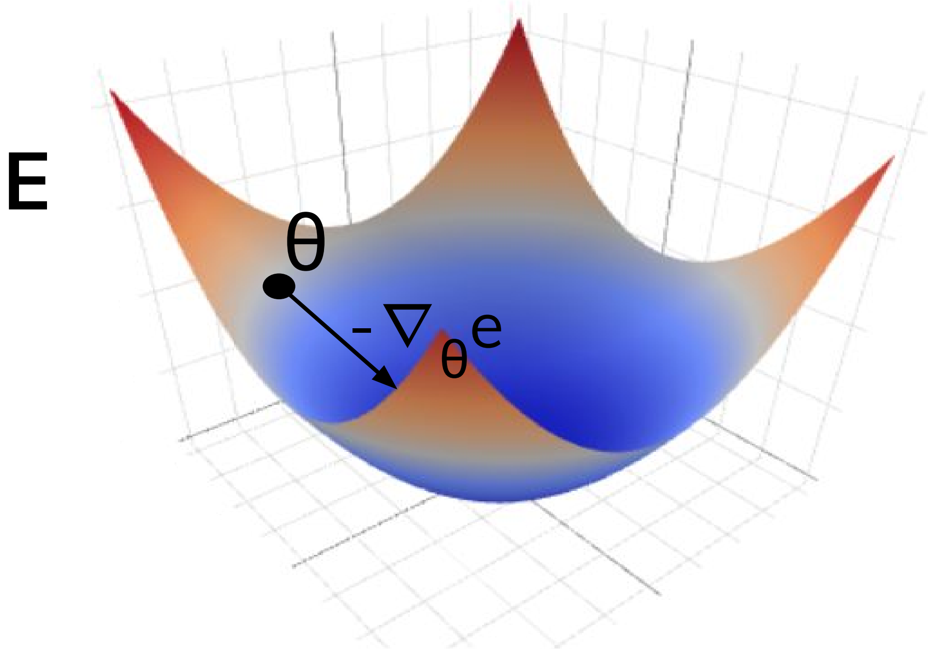
# How to find ***W*** and ***b***?



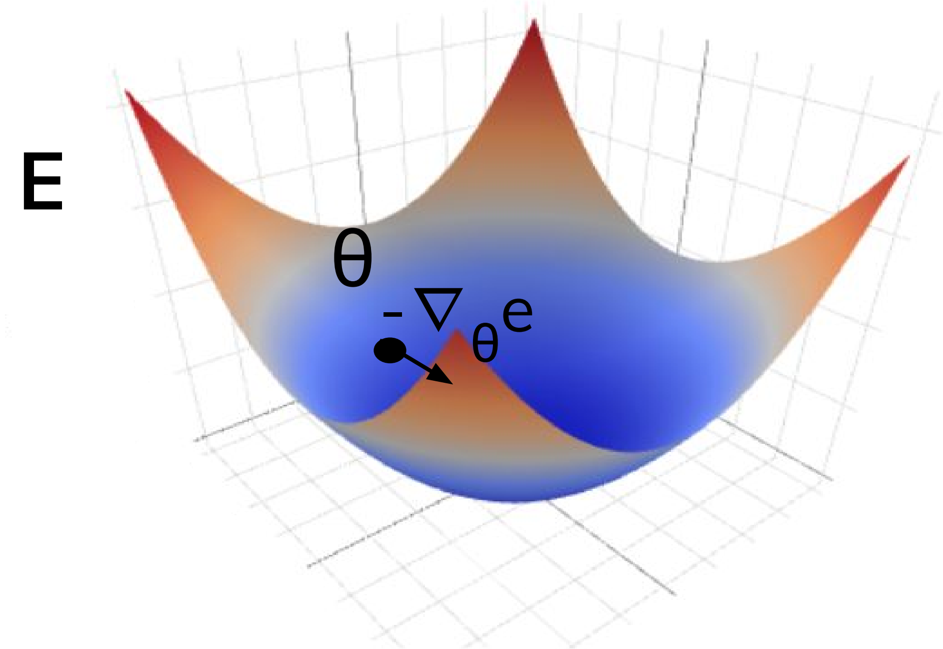
# How to find ***W*** and ***b***?



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

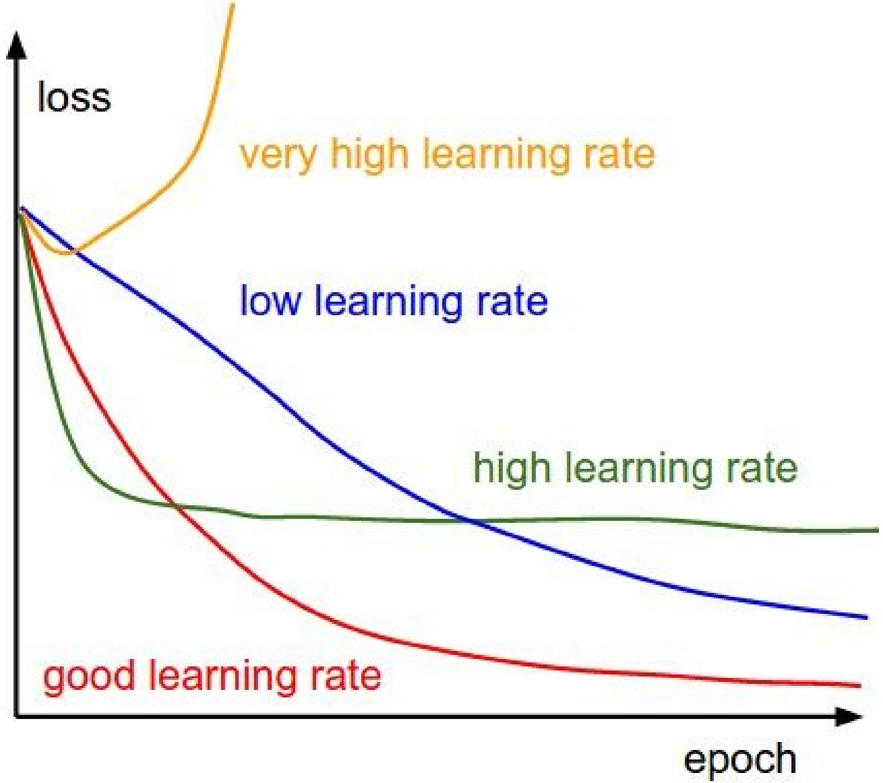


# How to find ***W*** and ***b***?



And so on...

# How to find $\boldsymbol{W}$ and $\boldsymbol{b}$?

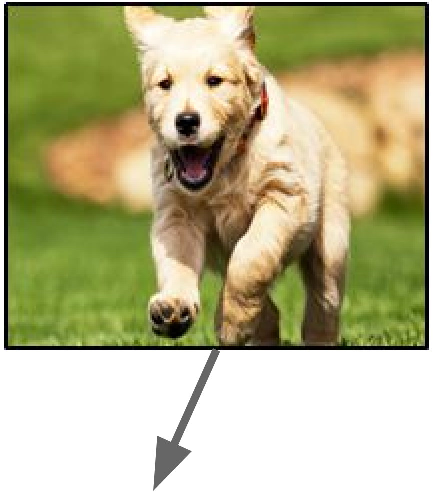


# Demo 2

# Logistic regression

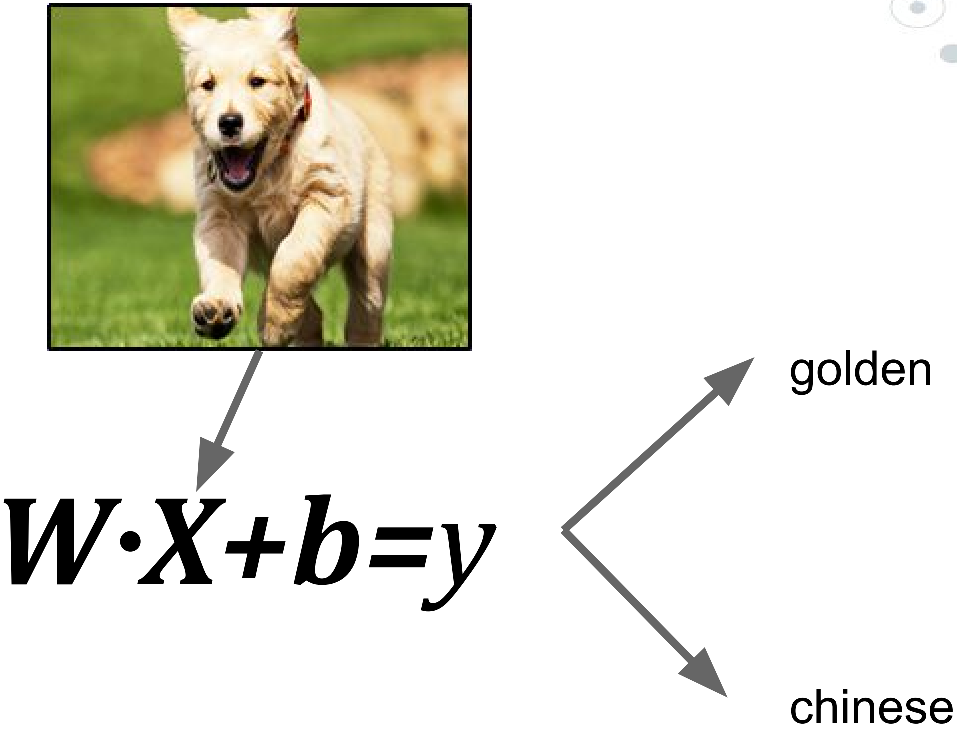
$$\boldsymbol{W \cdot X + b = y}$$

# Logistic regression



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

# Logistic regression

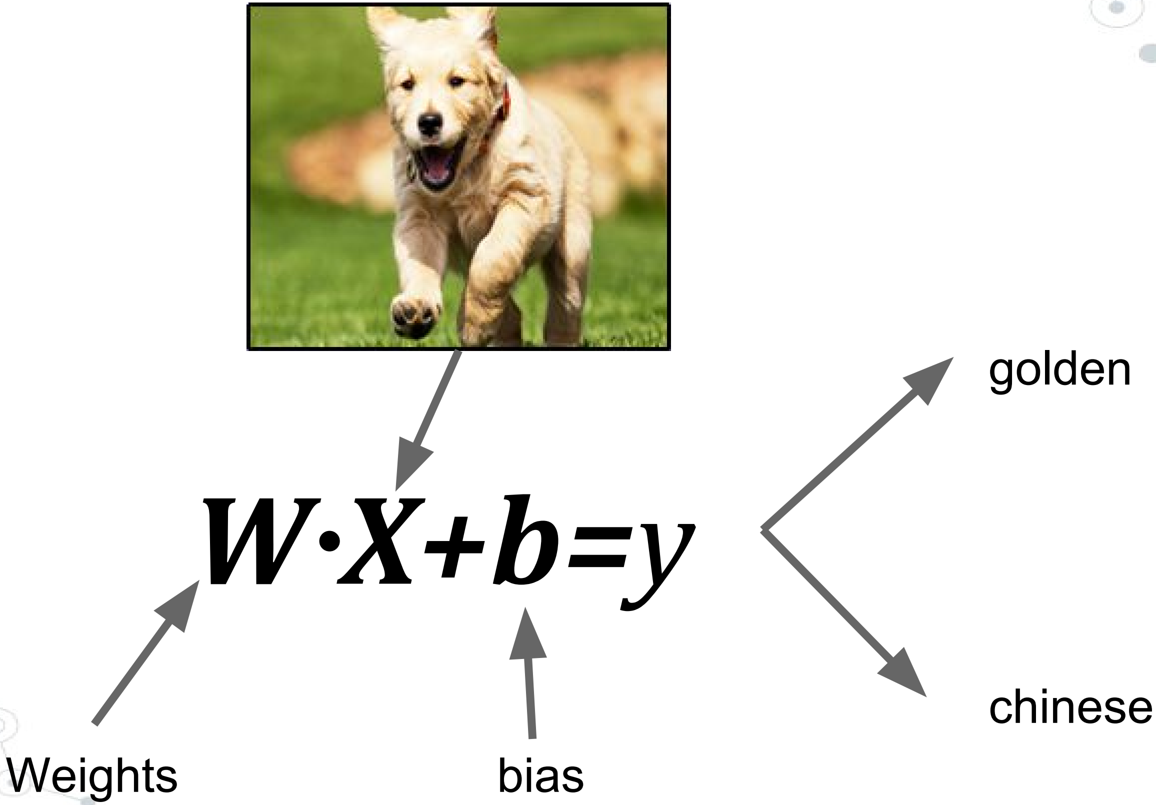


$$W \cdot X + b = y$$

golden

chinese

# Logistic regression



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

golden

chinese

Weights

bias

# Logistic regression

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

golden

chinese

# Logistic regression

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

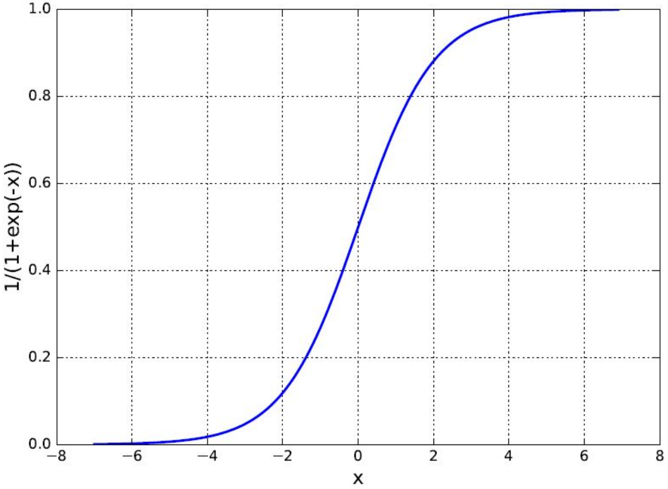
golden ✓
P(X)=0.99

chinese ✗
P(X)=0.05

# Logistic regression

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

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



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

# Logistic regression

+ Easy to train

+ Free adjustable hyper-parameters

- Could be very simple

- Only for linearly-separable classes

## How to find $\boldsymbol{W}$ and $\boldsymbol{b}$?

Answer:

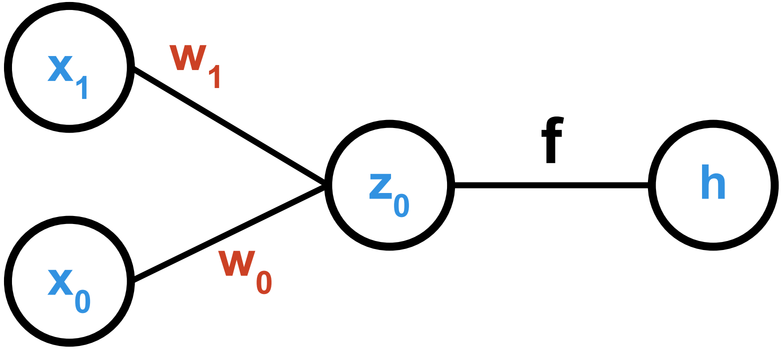
Gradient descendent algorithm

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

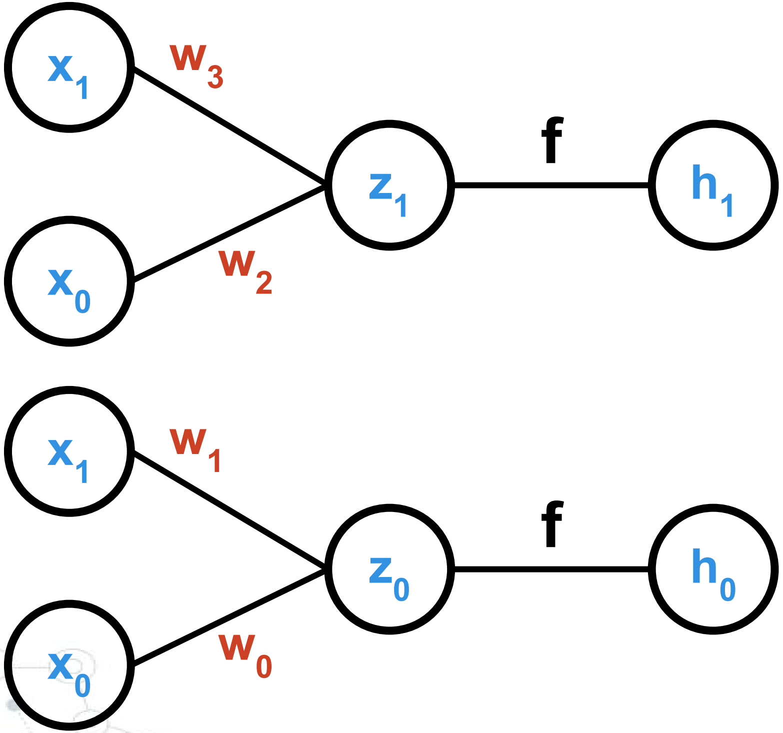
# Demo 3

# Neural Networks

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

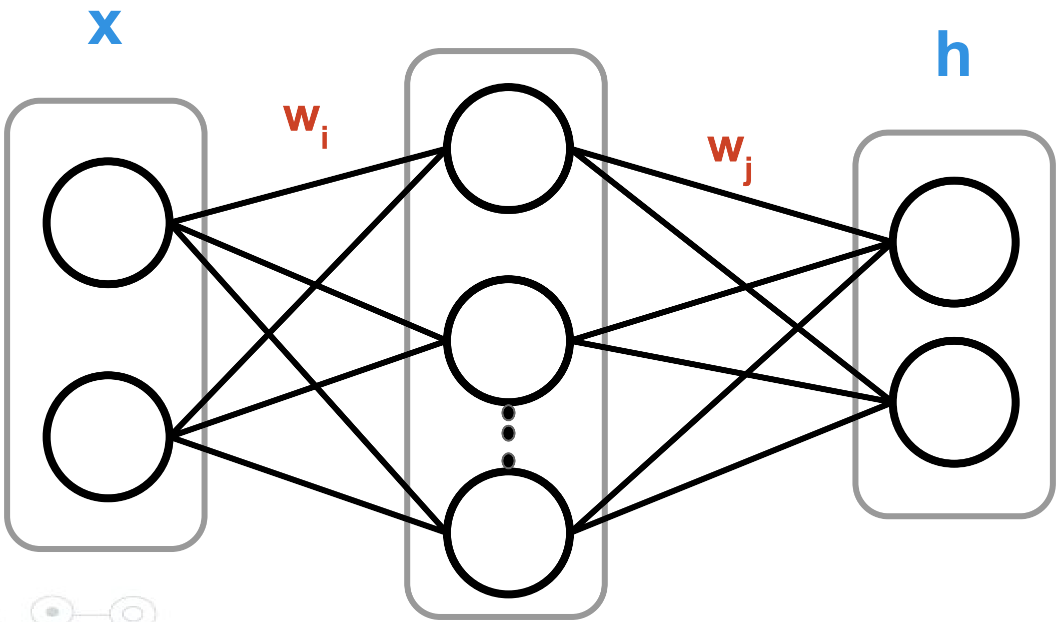


# Neural Networks

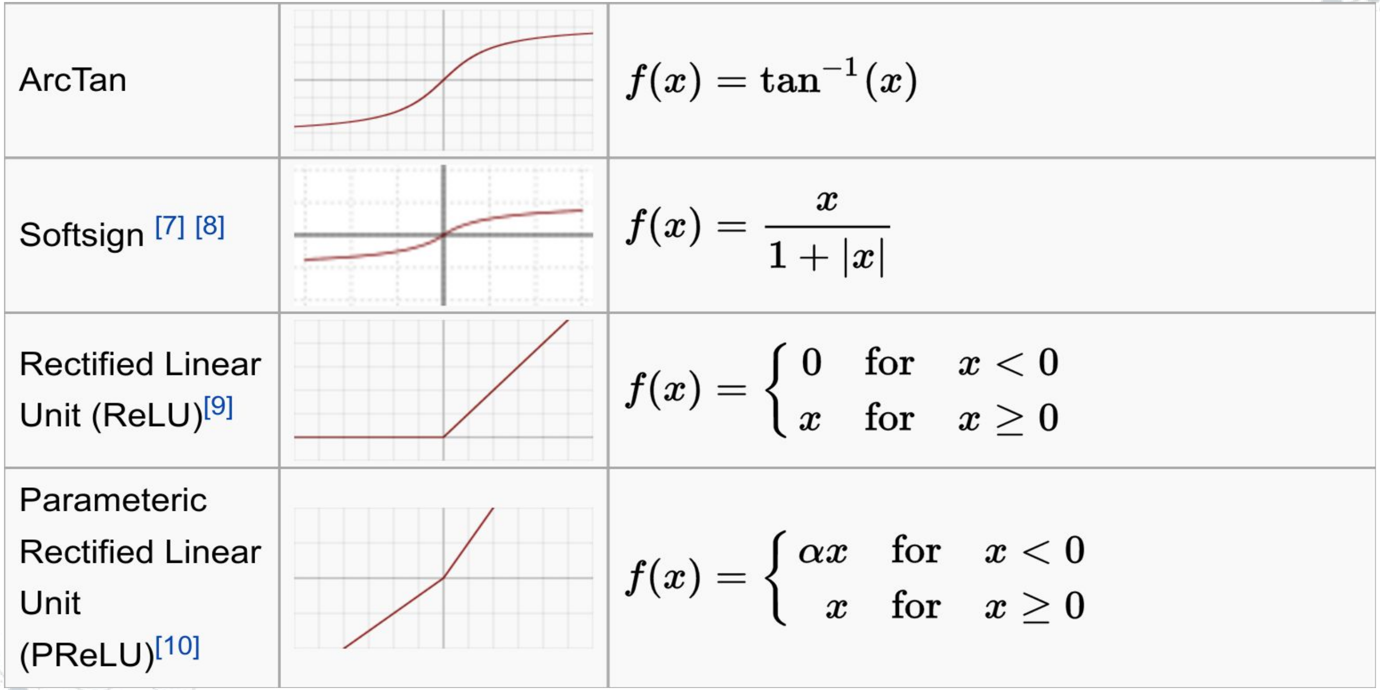


## Neural Networks

$$h = f(x; \theta)$$

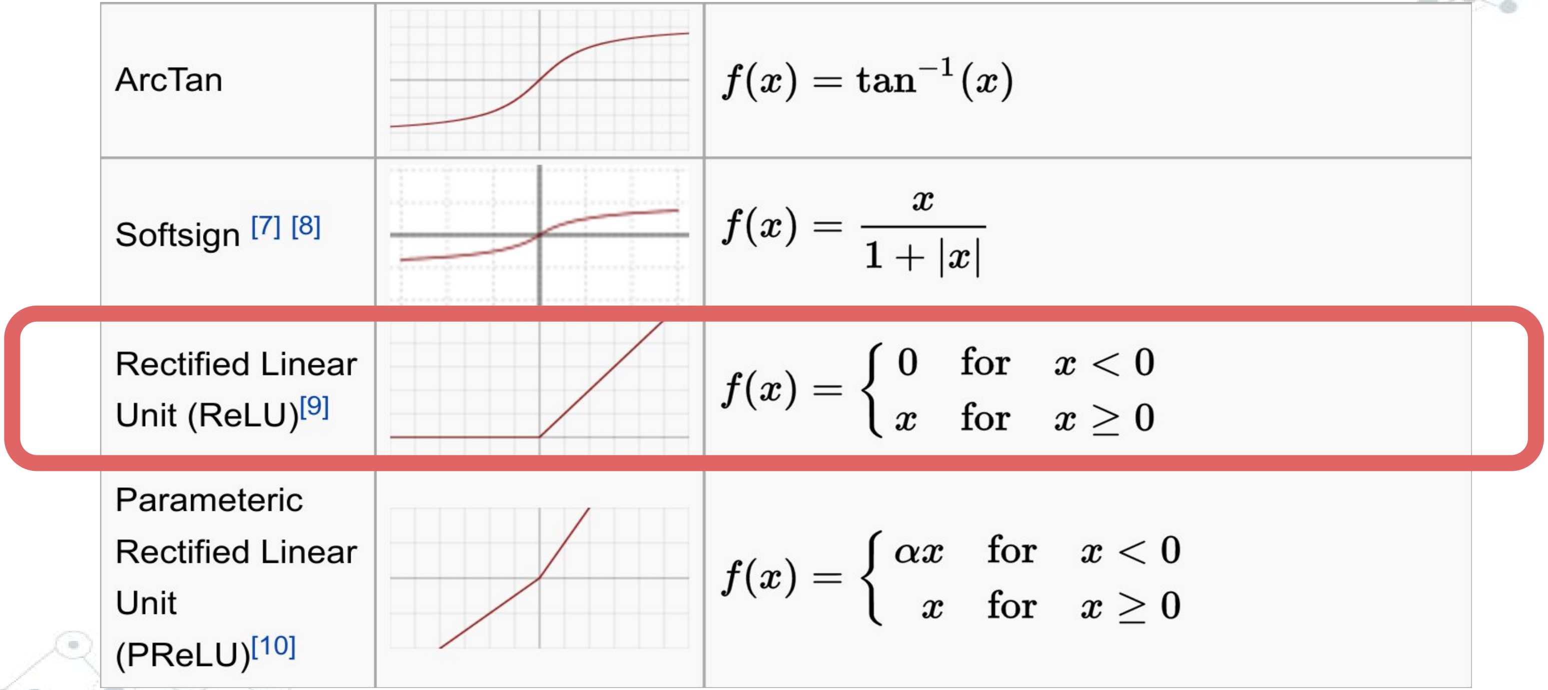


# Neural Networks



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

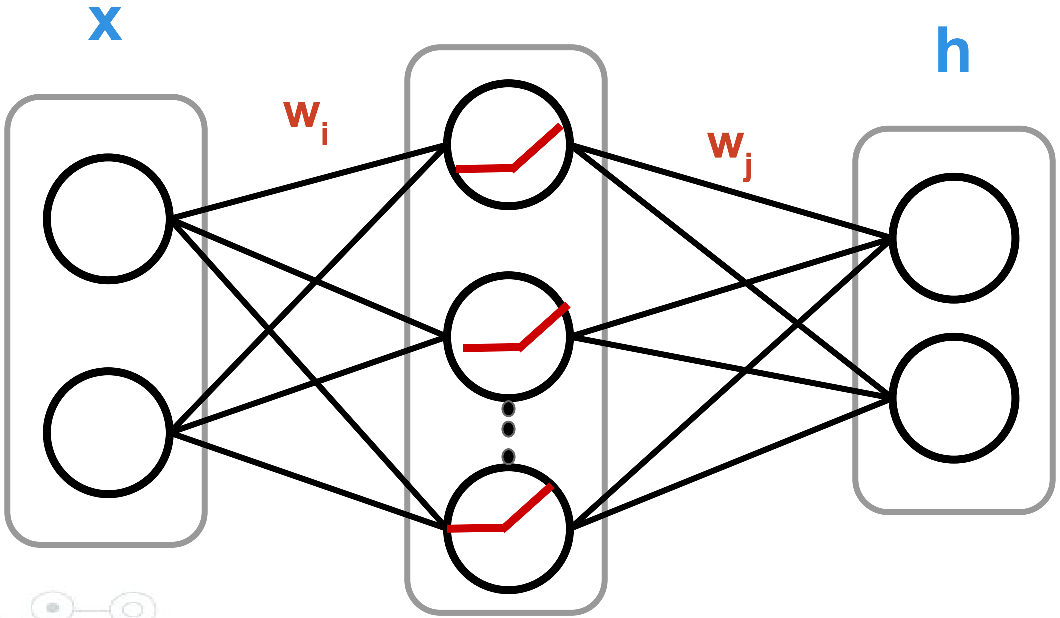
# Neural Networks



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

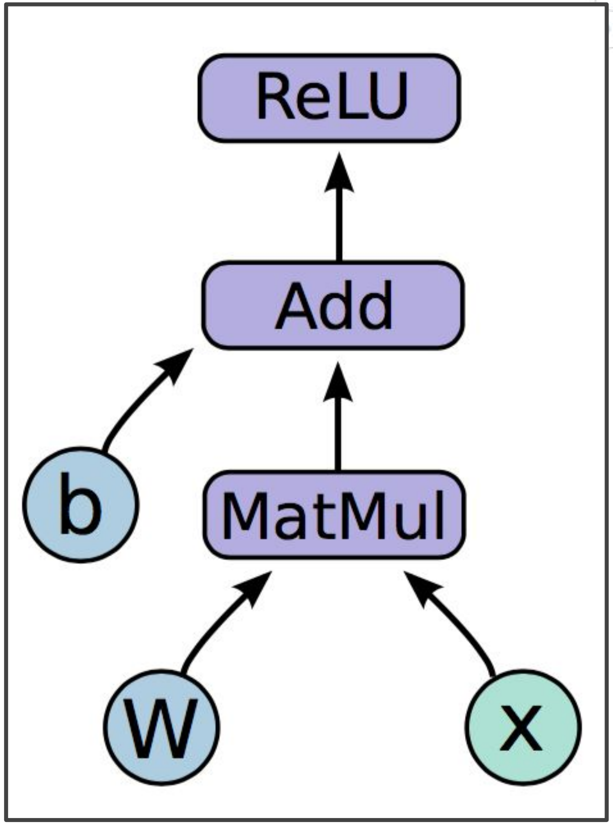
# Neural Networks

$$h = f(x; \theta)$$



# Neural networks

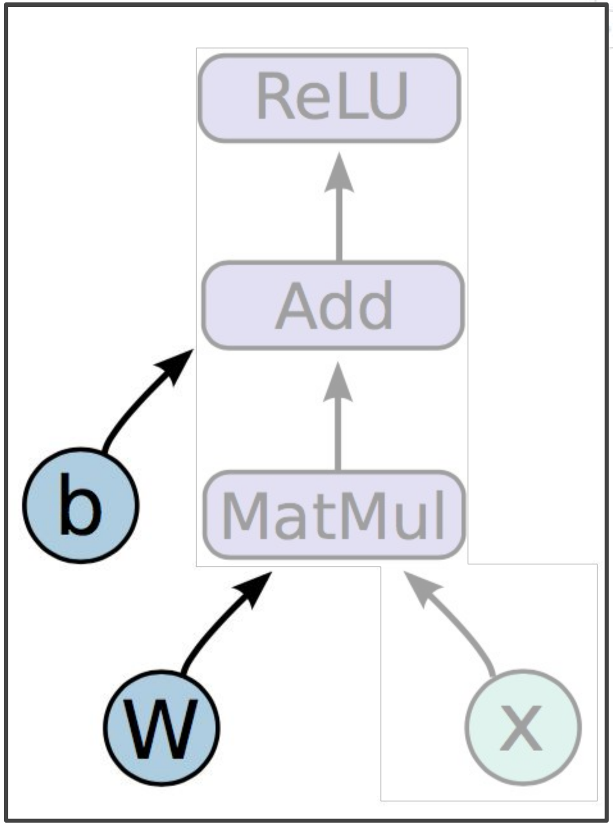
$$h_i = \mathrm{ReLU}(Wx + b)$$



## Neural networks

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

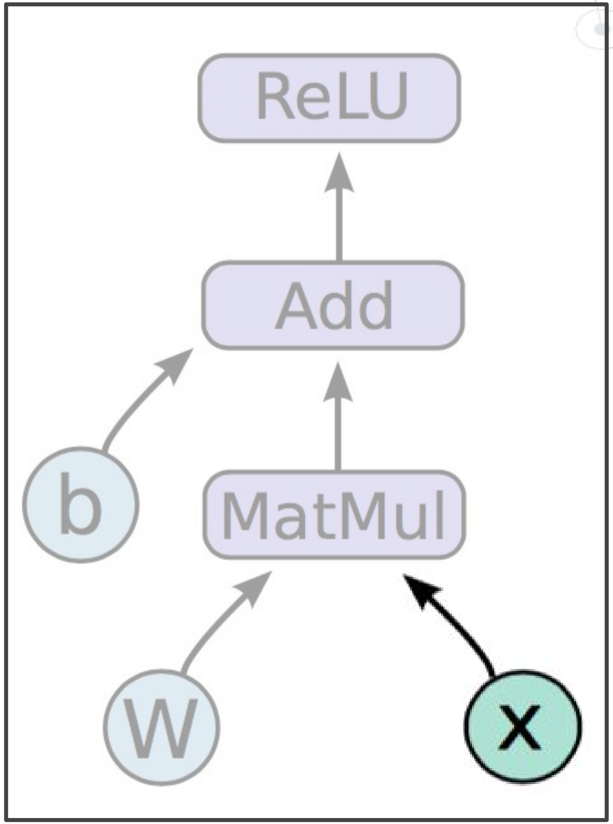
Parameters to fit (Variables)



# Neural networks

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

Input data
(Placeholder)

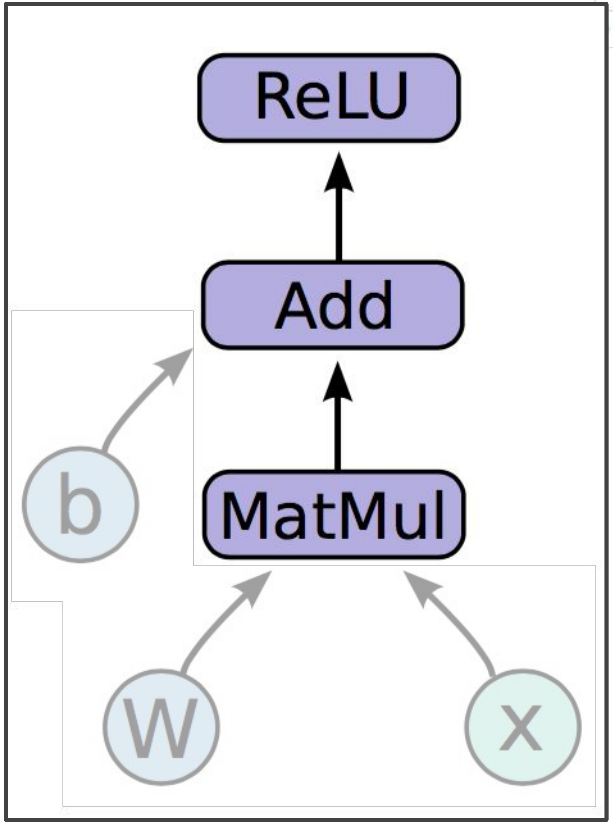


# Neural networks

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

Mathematical operations:

MatMul, Add, RELU

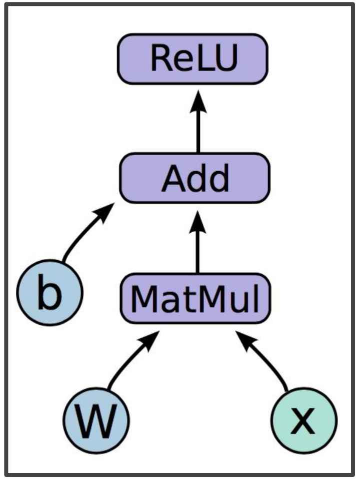


# Neural networks

## In code, please!

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

```
    import tensorflow as tf

    b = tf.Variable(tf.zeros((10,)))
1   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)
```



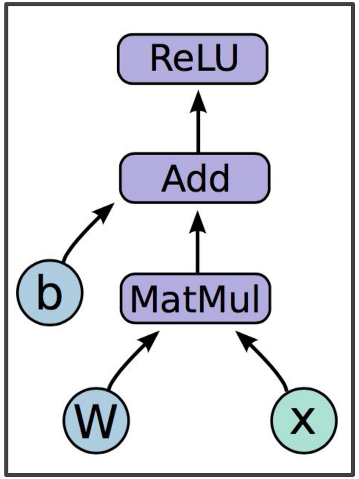
# Neural networks

## How to run it?

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

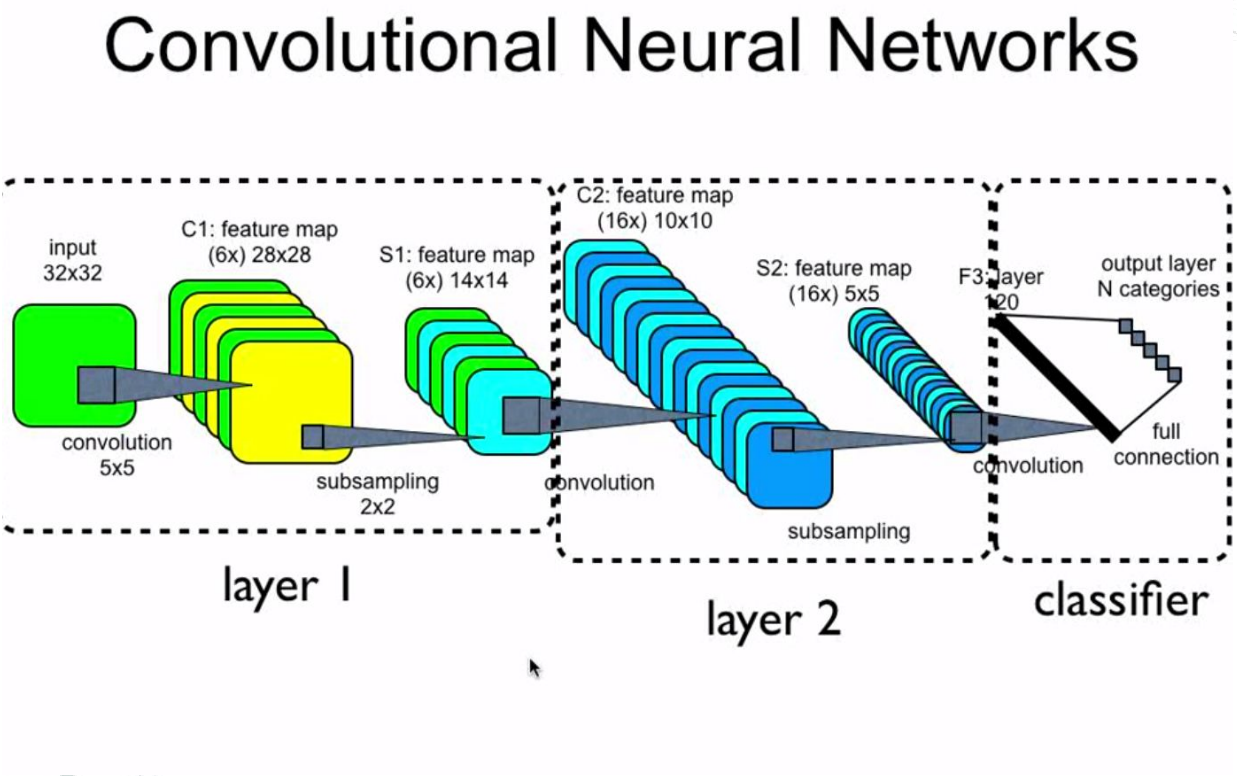
```
  import tensorflow as tf

  b = tf.Variable(tf.zeros((10,)))
1 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)})
```



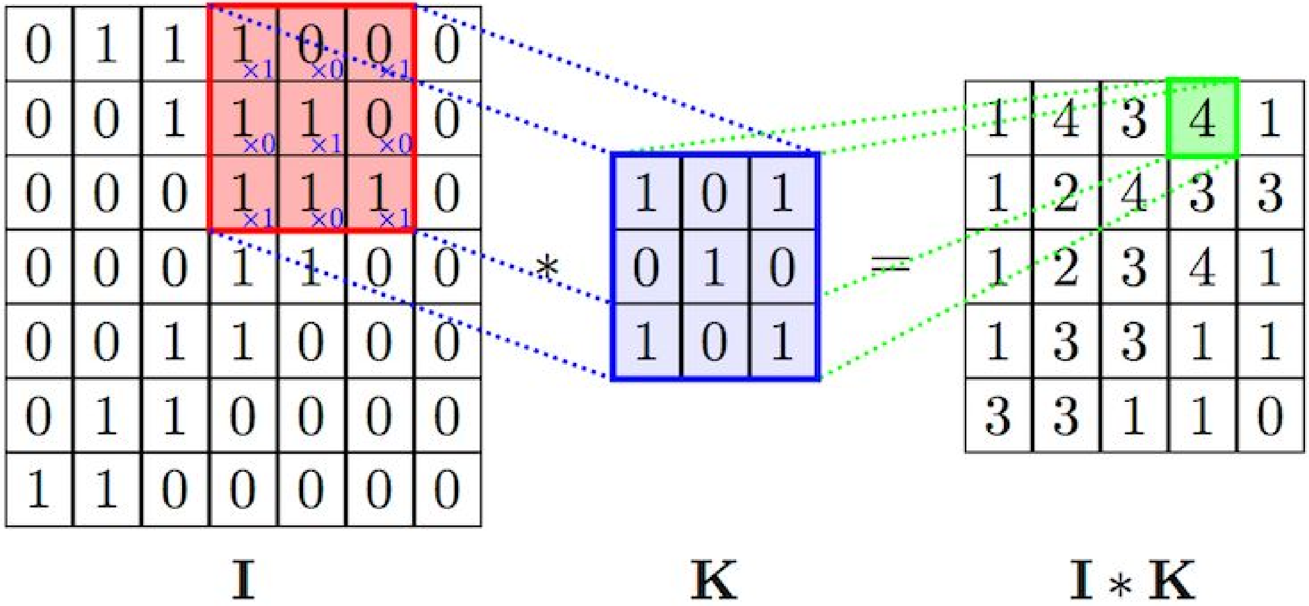
# Demo 4

# Convolutional layers



Convolutional Neural Networks

# Convolutional layers



|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |

**I**

|   |   |   |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

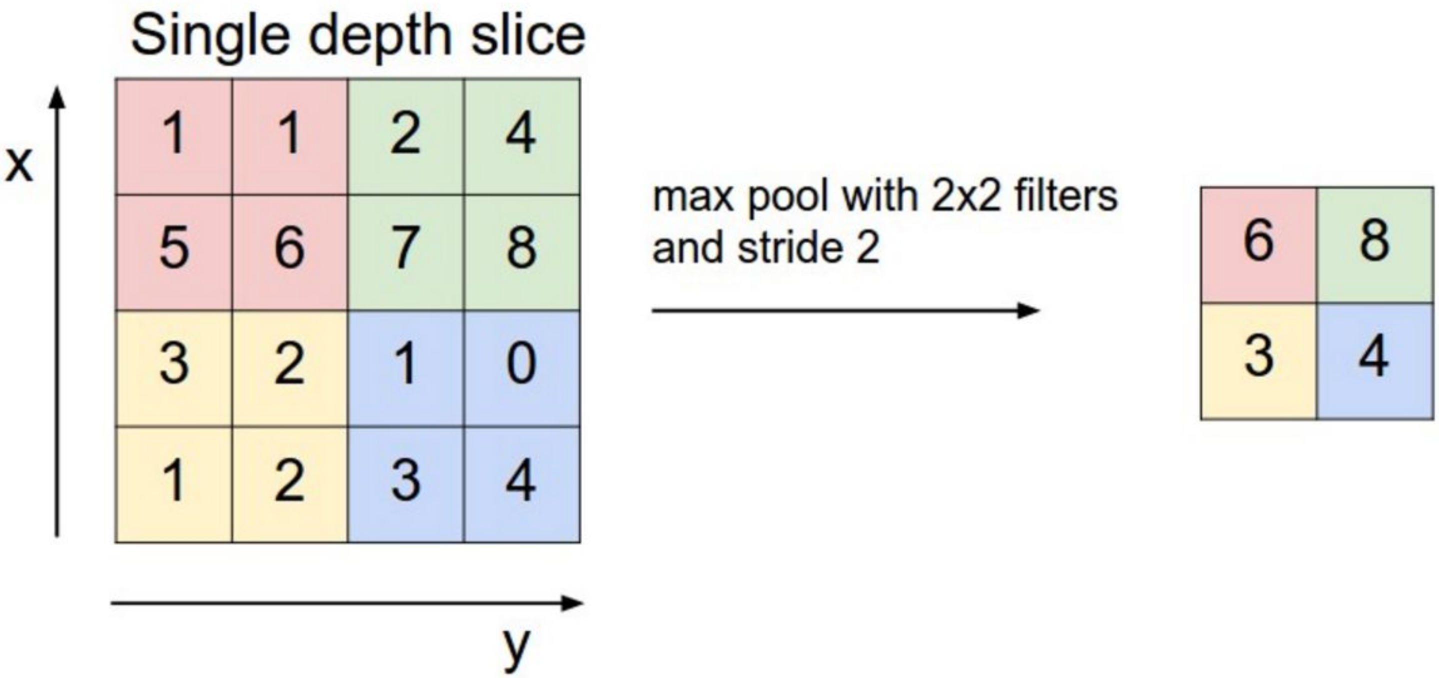
**K**

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 4 | 3 | 4 | 1 |
| 1 | 2 | 4 | 3 | 3 |
| 1 | 2 | 3 | 4 | 1 |
| 1 | 3 | 3 | 1 | 1 |
| 3 | 3 | 1 | 1 | 0 |

**I ∗ K**

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

# Pooling layers



Single depth slice

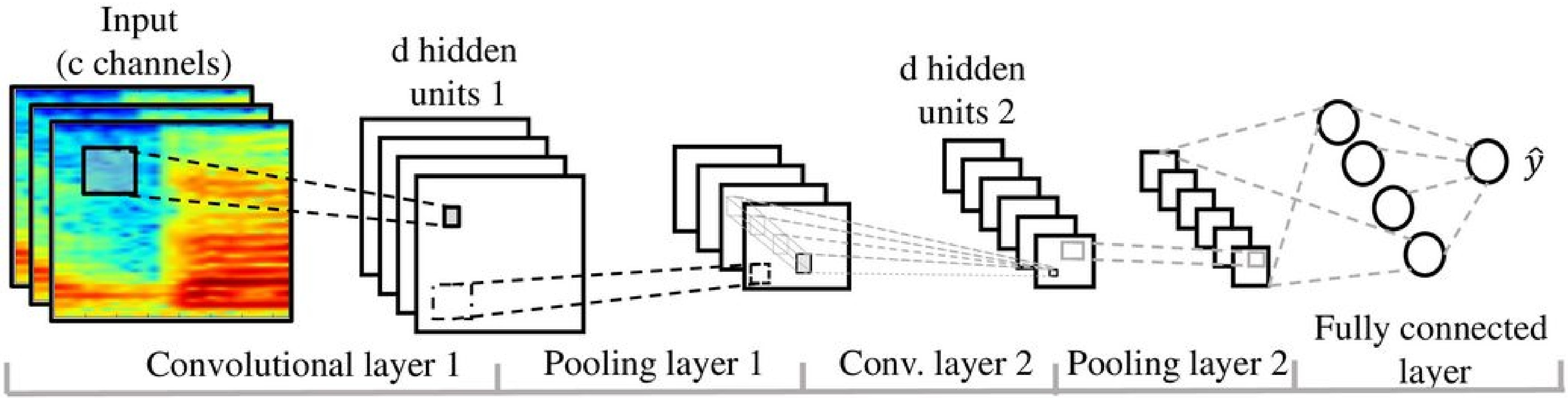
| | | | |
|---|---|---|---|
| 1 | 1 | 2 | 4 |
| 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 4 |

max pool with 2x2 filters and stride 2

| | |
|---|---|
| 6 | 8 |
| 3 | 4 |

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

[http://jcvasquezc.wixsite.com/home](http://jcvasquezc.wixsite.com/home)

# Tensorflow

ML Meetup

+ Meetups