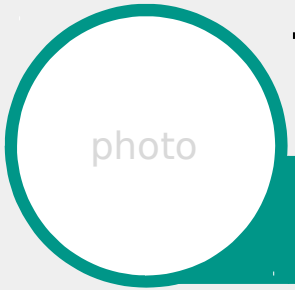
Introducción a Deep Learning con Tensorflow



José Laruta - jose.laruta@ieee.org
Elemental

AGENDA

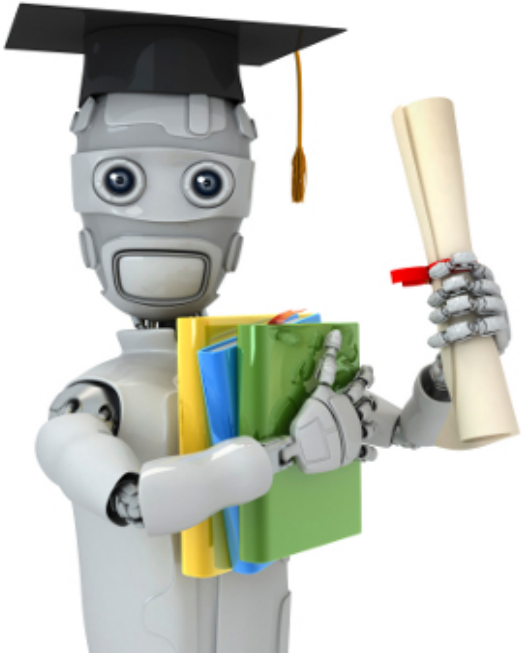
- MACHINE LEARNING
  - Qué es? Para qué sirve?
  - Conceptos Básicos
    - Aprendizaje Supervisado
    - Recordando nuestros orígenes: Regresión lineal
    - Descenso de gradiente
- REDES NEURONALES
  - El concepto de neurona
  - Funciones de activacion
  - Redes neuronales profundas
    - Redes densamente conectadas
    - Redes convolucionales
    - Redes recurrentes

GD G

MACHINE LEARNING

- Subcampo de la inteligencia artificial.
- Modelos de aprendizaje.
- Se basa en la búsqueda de patrones en grandes cantidades de datos.

Tipos de Aprendizaje Automático



- Aprendizaje Supervisado.
- Aprendizaje No Supervisado.
- Aprendizaje por Refuerzo.

Tipos de Aprendizaje Automático

Poniendo el ejemplo, Aprendizaje Supervisado

- Conjunto de datos etiquetados previamente.
- Mientras más datos, mejor.
- Buscamos generalizar para nuevos casos.

Aprendizaje Supervisado

- Regresión (salida contínua)
- Clasificación (salida discreta)

Aprendizaje Supervisado

- Regresión (salida contínua)
  - Estimación del precio de una casa
- Clasificación (salida discreta)
  - Detección de rostros en imágenes

GD
G

Patrones ocultos, Aprendizaje No Supervisado

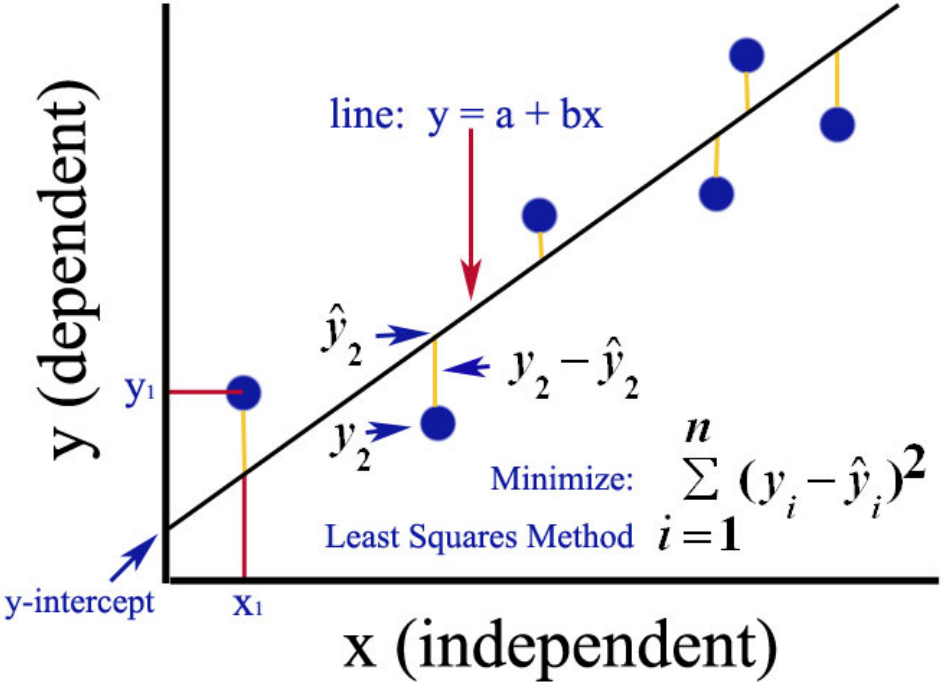
- Conjunto de datos no etiquetados.
- Mayor abundancia.
- Se buscan patrones inmersos en los datos

Aprendizaje No Supervisado

- Clustering (Agrupación de datos)
  - Segmentación de clientes
- Visualización (Reducción de dimensionalidad)
  - Visualización de datos de gran dimensionalidad

GD
G

Regresión Lineal



GD
G

Modelo de la regresión lineal

Hipótesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Función de costo:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

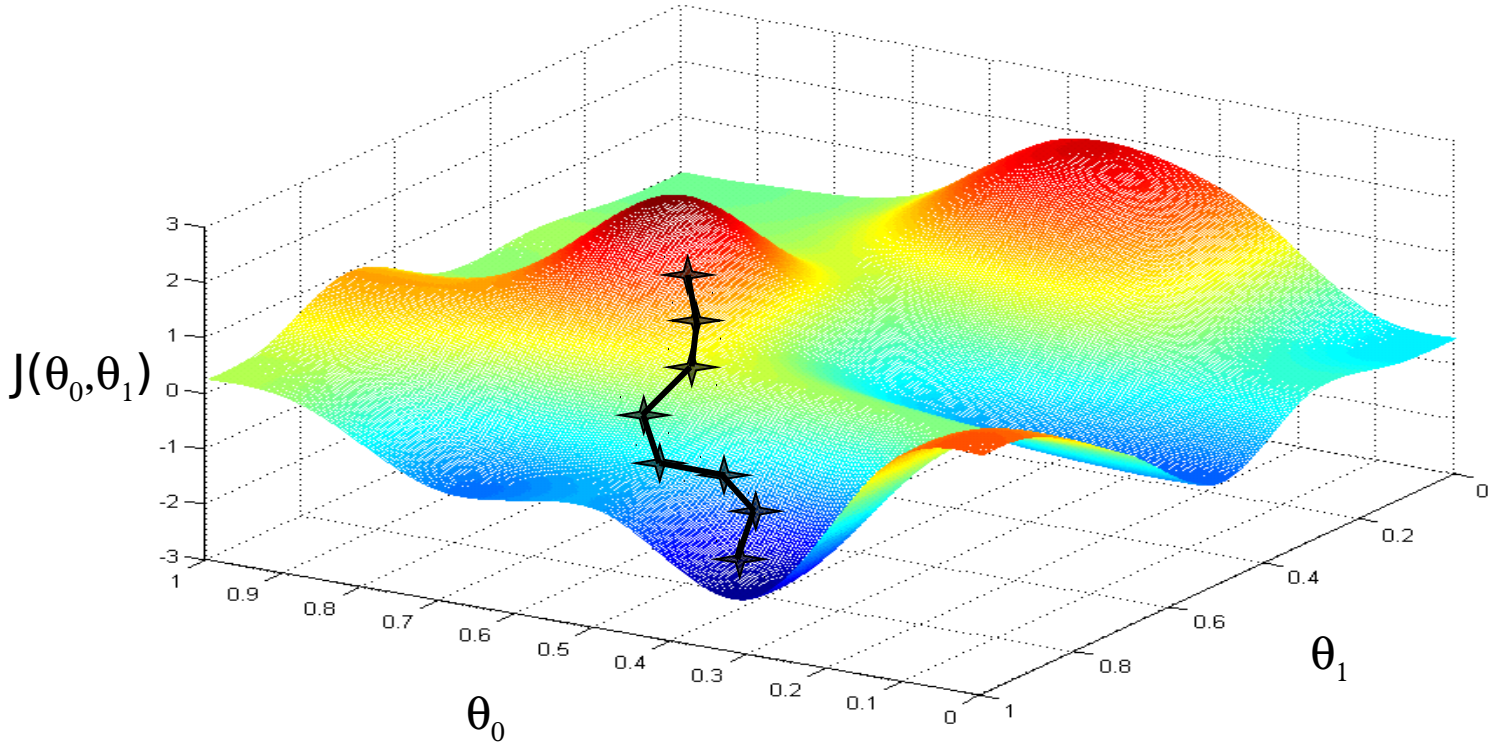
Descenso de gradiente

$$\text{repeat until convergence } \{$$

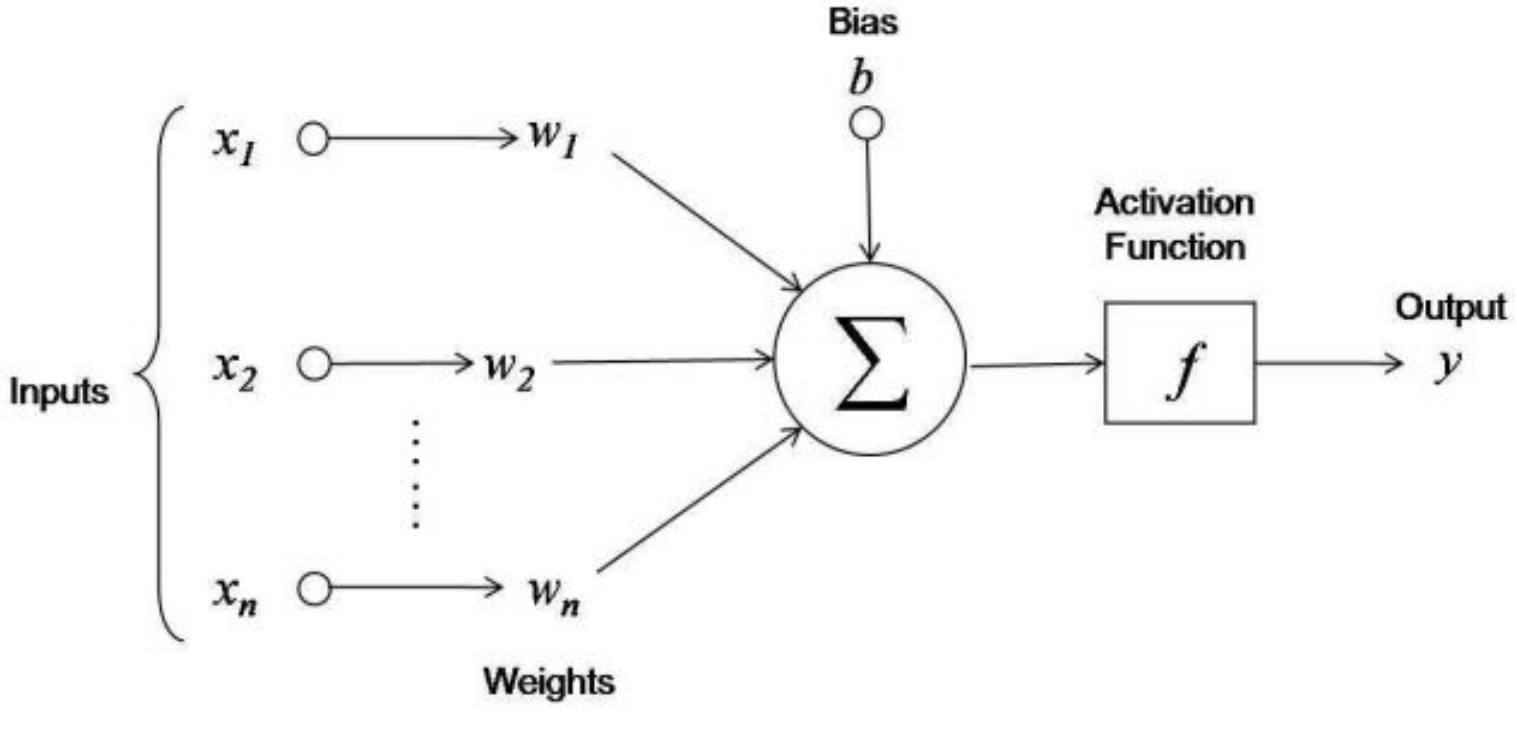
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

$$\}$$


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Una neurona


Bias
b
Activation
Function
Output
Inputs
x_1
x_2
x_n
w_1
w_2
w_n
$\sum$
f
y
Weights

Modelo de la regresión lineal

Hipótesis:

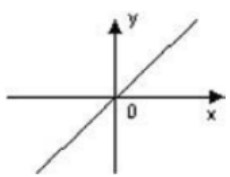
$$h_\theta(x) = \theta_0 + \theta_1 x$$

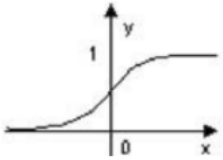
Función de costo:

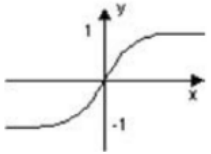
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Funciones de Activación

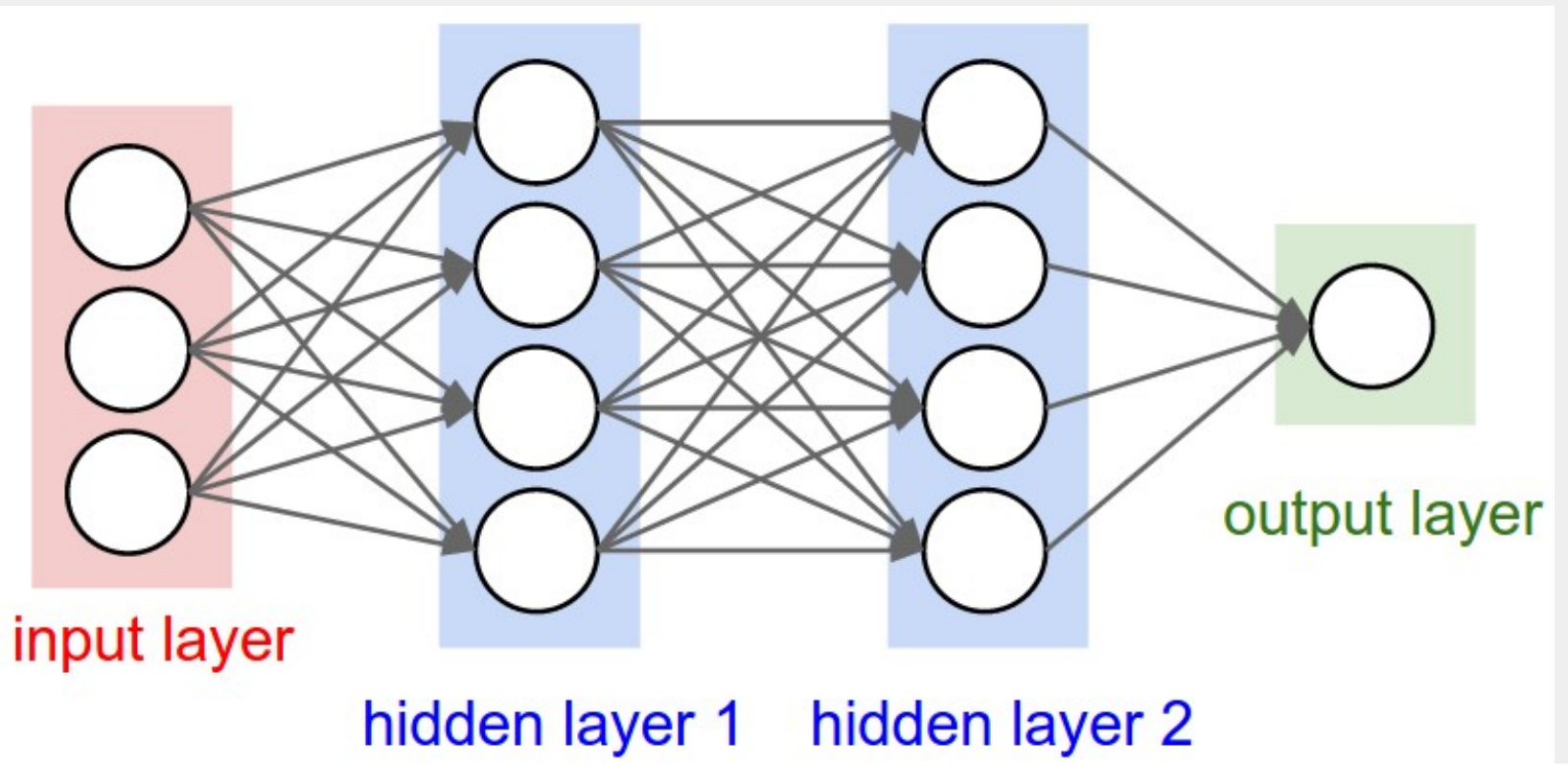
- Dependiendo el tipo de aplicación se incluye una función de activación.
- Regresión: Activación lineal o RELU.
- Clasificación: Sigmoide o tanh



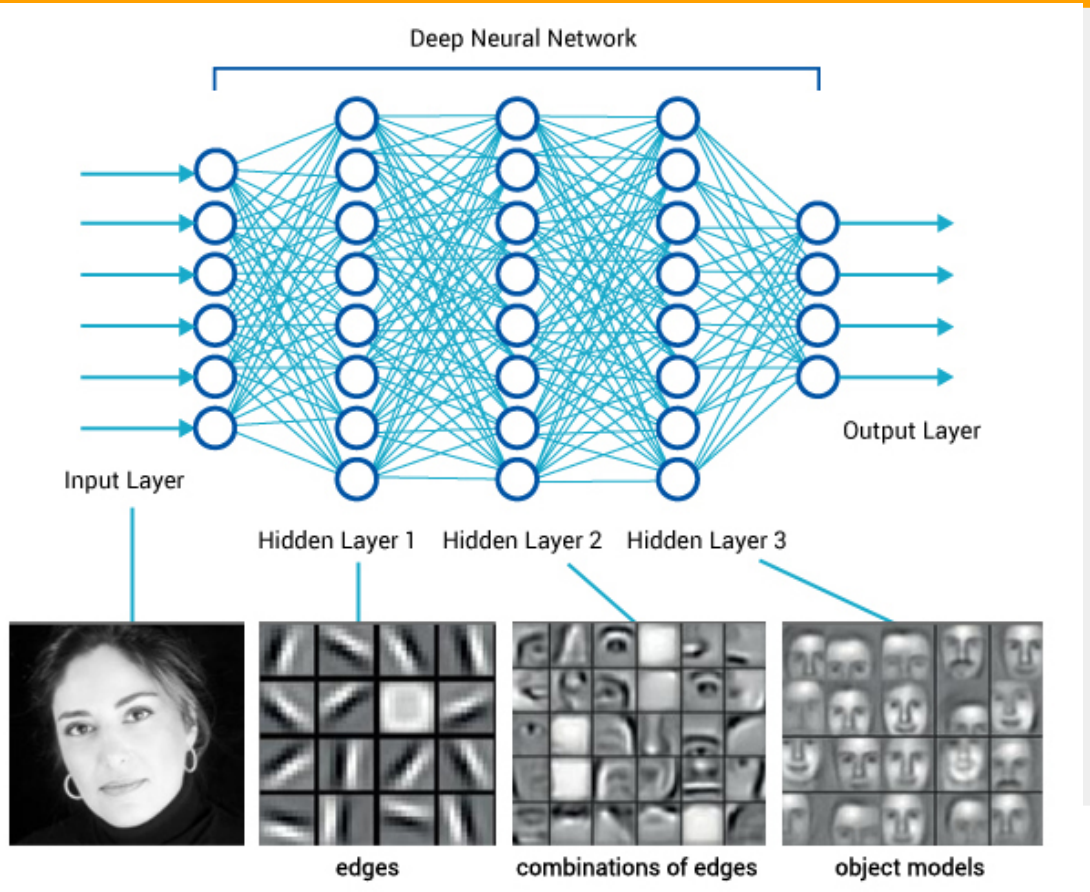




Red Neuronal



Red Neuronal Profunda



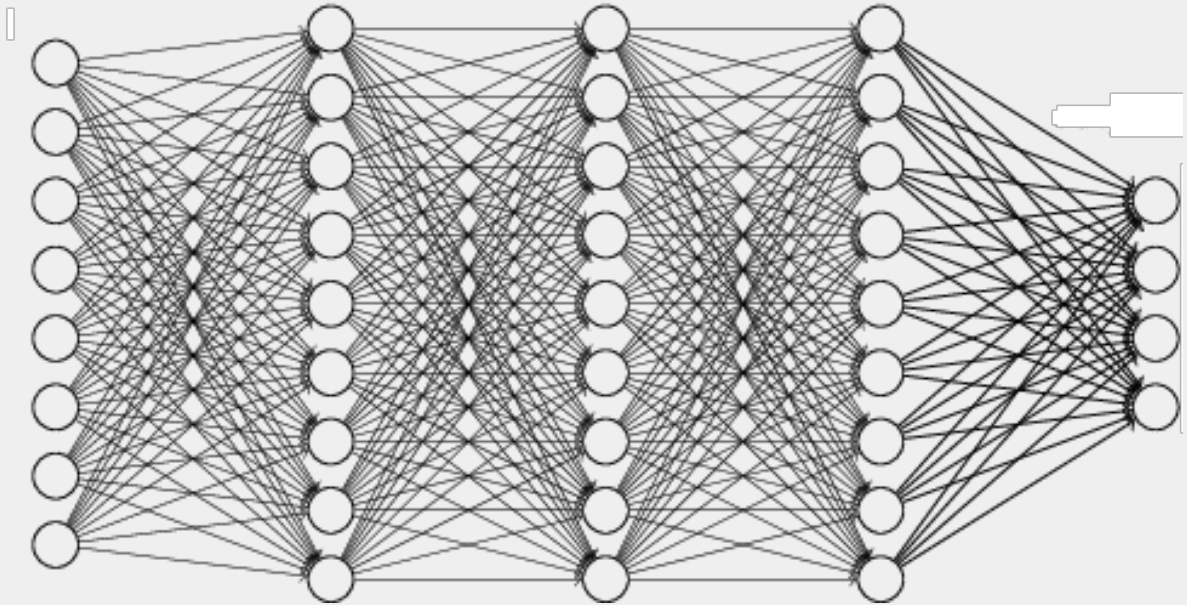
Aprendizaje Profundo

- Clasificación y regresión.
- Procesamiento de imágenes y video.
- Procesamiento de texto y series de tiempo.

Aprendizaje Profundo

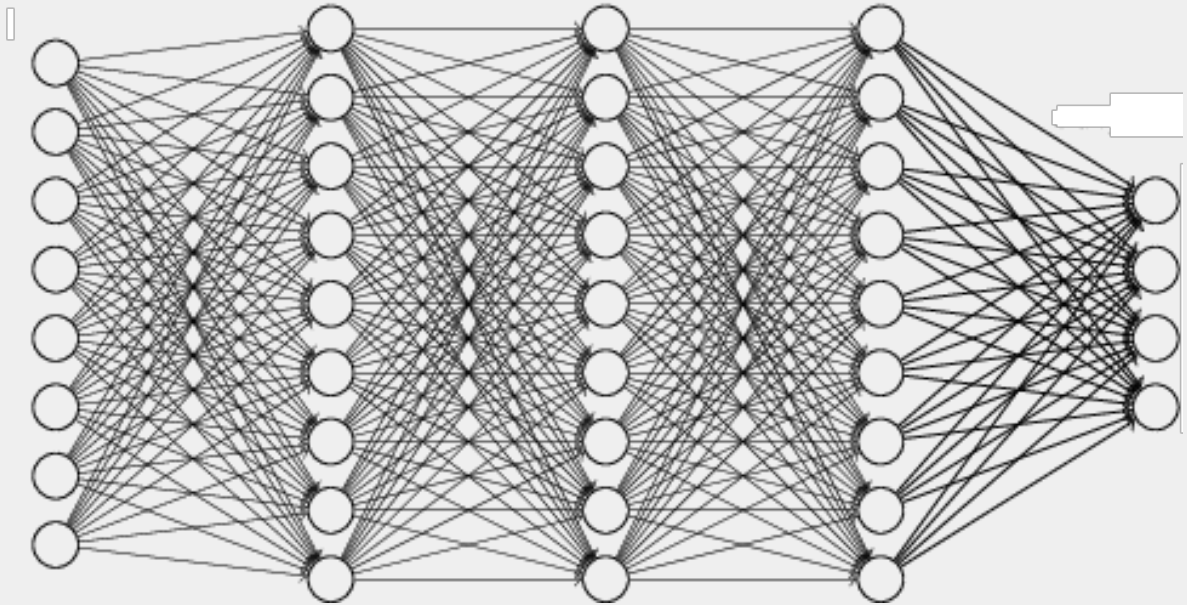
- Clasificación y regresión.
- Procesamiento de imágenes y video.
- Procesamiento de texto y series de tiempo.
- Redes densamente conectadas
- Redes neuronales Convolucionales
- Redes neuronales recurrentes.

Redes Densamente Conectadas



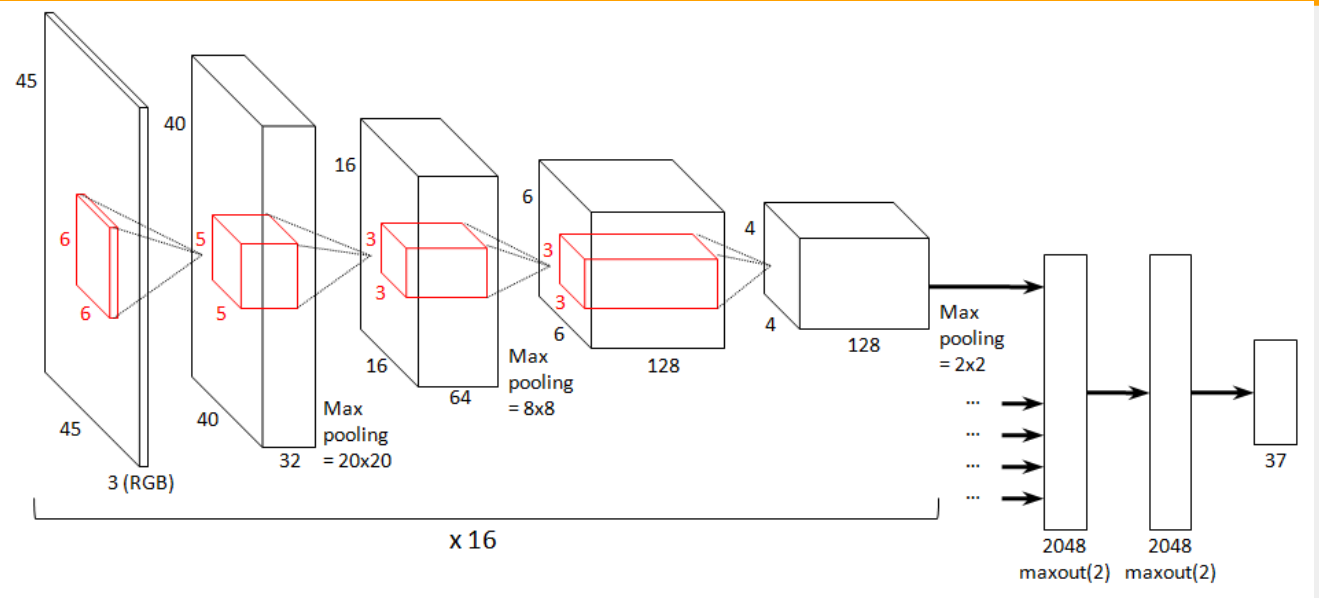
- Clasificación y regresión.
- Procesamiento de imágenes y video.
- Procesamiento de texto y series de tiempo.

Redes Densamente Conectadas



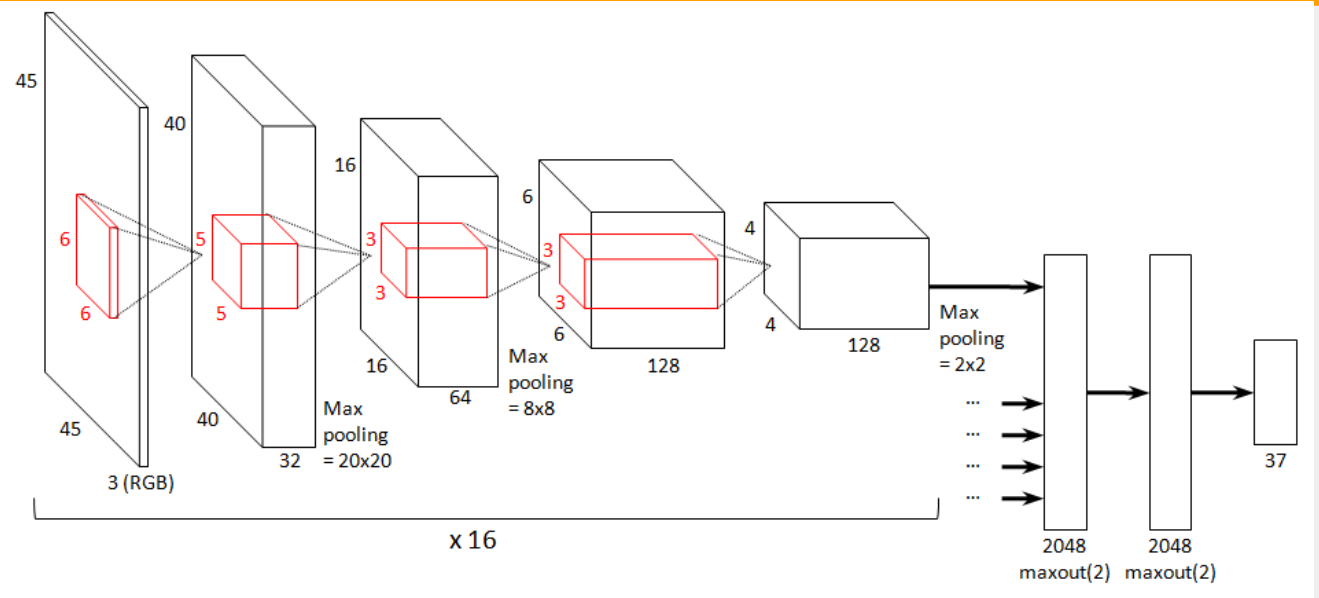
- Detección de fraude bancario
- Análisis de grandes bases de datos.

Redes Convolucionales



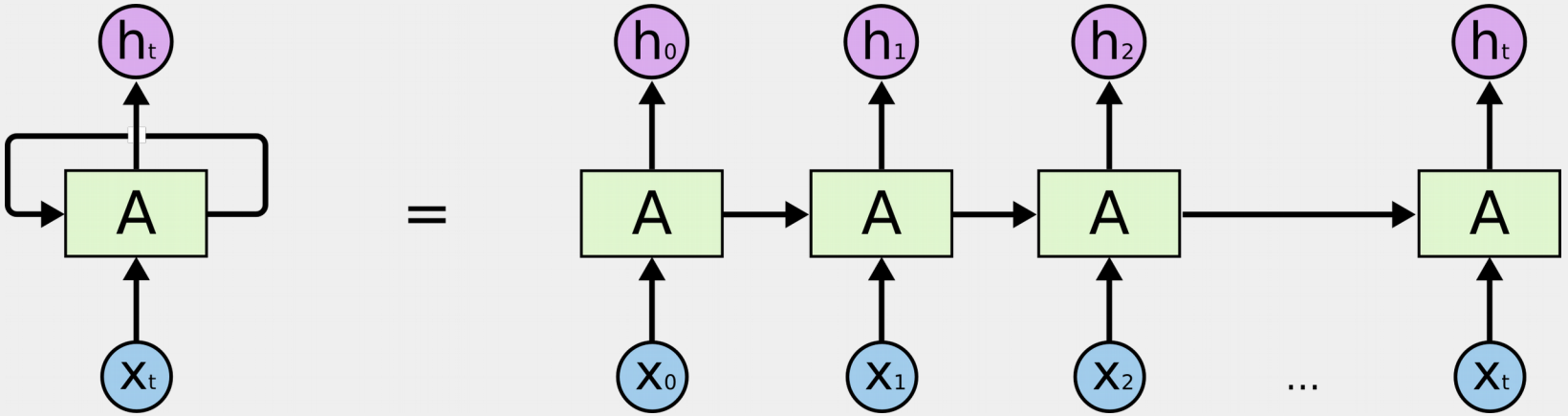
- Kernels para convolución
- Util para dependencias espaciales (imágenes)
- Alto costo computacional

Redes Convolucionales



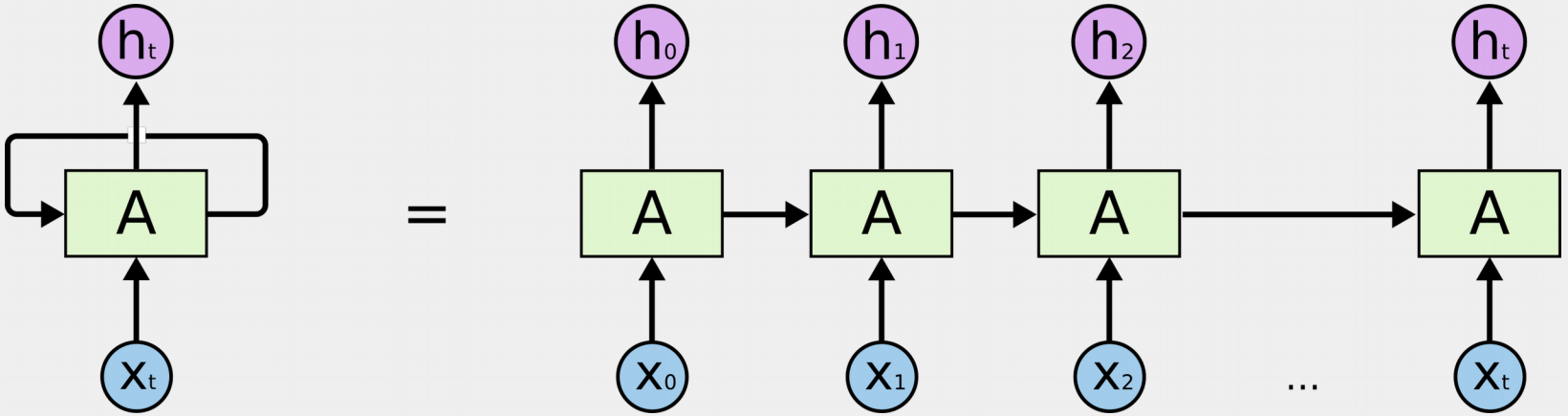
- Reconocimiento de objetos (Google Lens)
- Segmentación de imágenes
- Análisis de imágenes hiperespectrales y médicas.

Redes Recurrentes



- Dependencias temporales
- Análisis de secuencias
- Puede “recordar” estados anteriores

Redes Recurrentes



- Predicción y clasificación de series de tiempo
- Generación de secuencias
- Procesamiento de lenguaje natural (secuencias de texto)


DEEP LEARNING
memegenerator.es

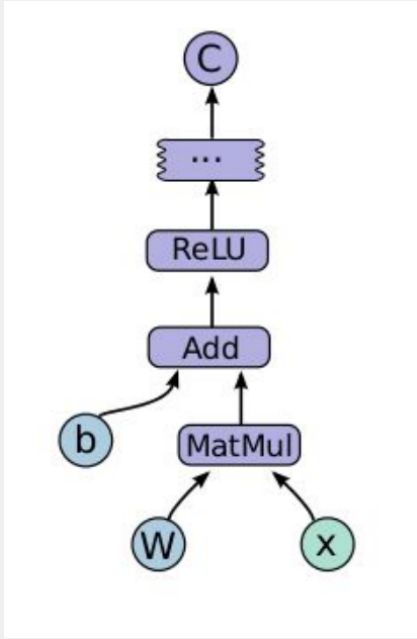

TensorFlow

Tensorflow

- Desarrollada por Google Brain
- Provee primitivas para definir funciones sobre tensores.
- Cálculo automático de derivadas (gradientes)
- Open source

Grafo de flujo de datos

- Los cálculos se representan como Grafos
  - Los nodos son operaciones.
  - Las líneas son tensores.
- Paradigma de programación declarativa
  - Se construye el grafo (modelo).
  - Se ejecutan los cálculos (flujo de tensores).



Grafo de flujo de datos

- Definición del grafo:
  - ```
    graph = tf.Graph()
    with graph.as_default():
    ```
    ```

Grafo de flujo de datos

- Definición de las variables:
  -

```
weights = tf.Variable(tf.truncated_normal([FLATTENED_SIZE, NUM_LABELS]), name='W')
biases = tf.Variable(tf.zeros([NUM_LABELS]),name='b')
```

Modelo de la regresión lineal

Hipótesis:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

```
logits = tf.matmul(tf_train_dataset, weights) + biases
```

Función de costo:

```
loss = tf.reduce_mean
            (
            tf.nn.softmax_cross_entropy_with_logits
                (
                labels=tf_train_labels,
                logits=logits
                )
            )
```

Grafo de flujo de datos

- Algoritmo de optimización

```
optimizer = tf.train.GradientDescentOptimizer(0.5).minimize(loss)
```

Ejecución de operaciones

- Creación de una sesión:

```
with tf.Session(graph=graph) as session:
    tf.global_variables_initializer().run()
```

Ejecución de operaciones

- Entrenamiento:

```
feed_dict = {
            tf_train_dataset : batch_data,
            tf_train_labels : batch_labels
            }

_, l, predictions = session.run([optimizer, loss, train_prediction], feed_dict=feed_dict)
```

Demo time!

thank you

GD
G