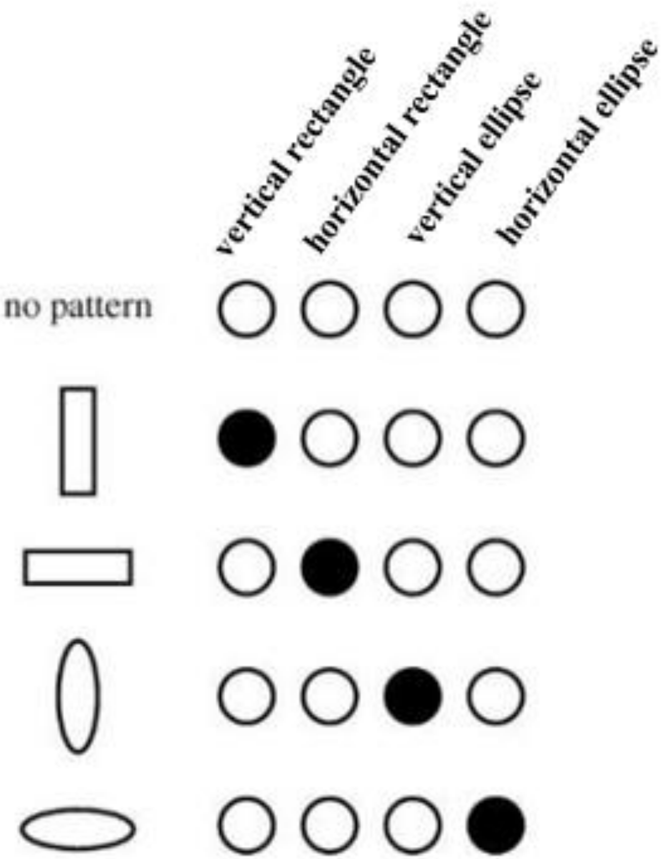
Redes neuronales

Introducción

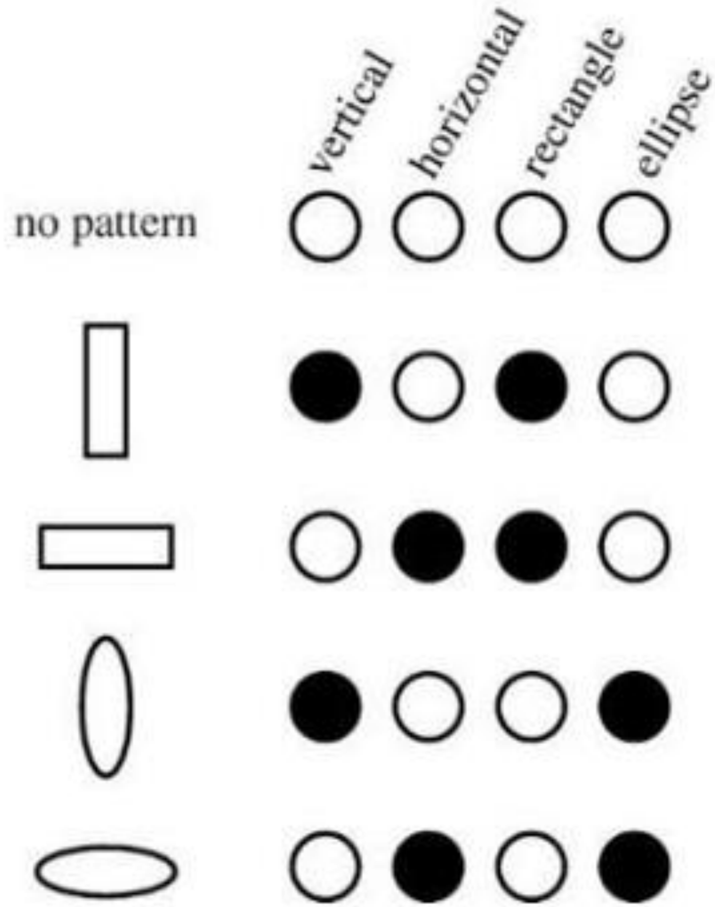
Diplomado de Minería de Datos

ITAM

Representación distribuida

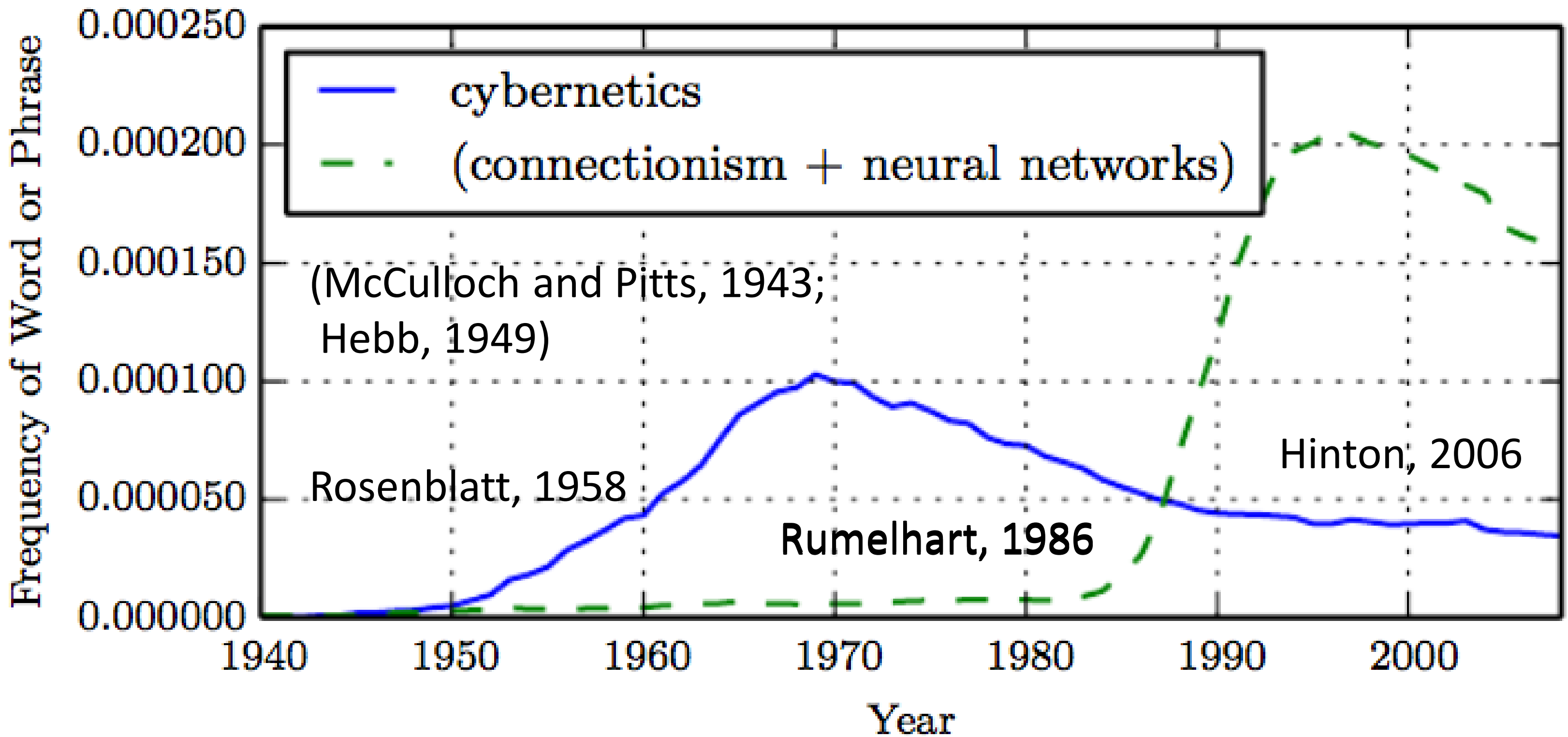


localista

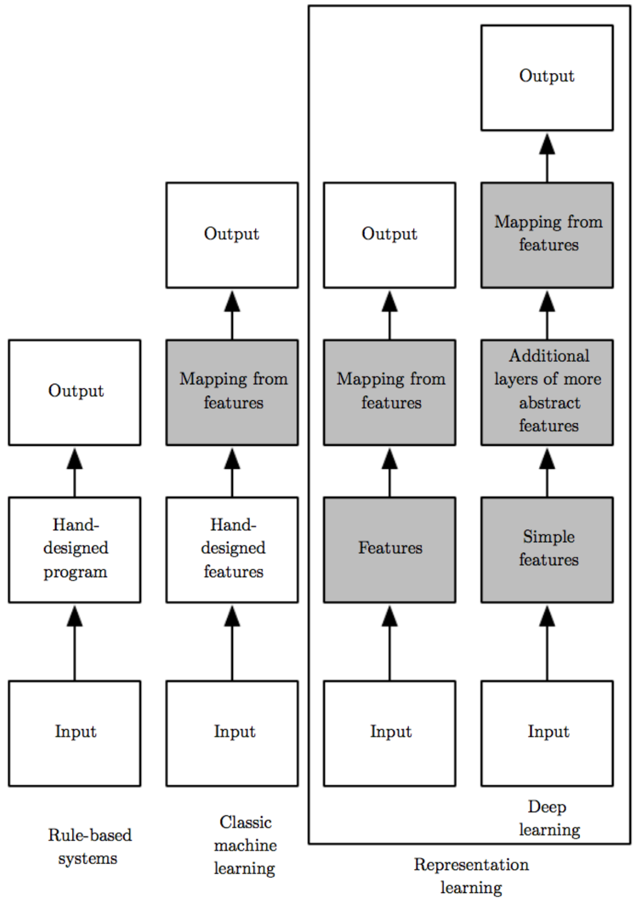


distribuida

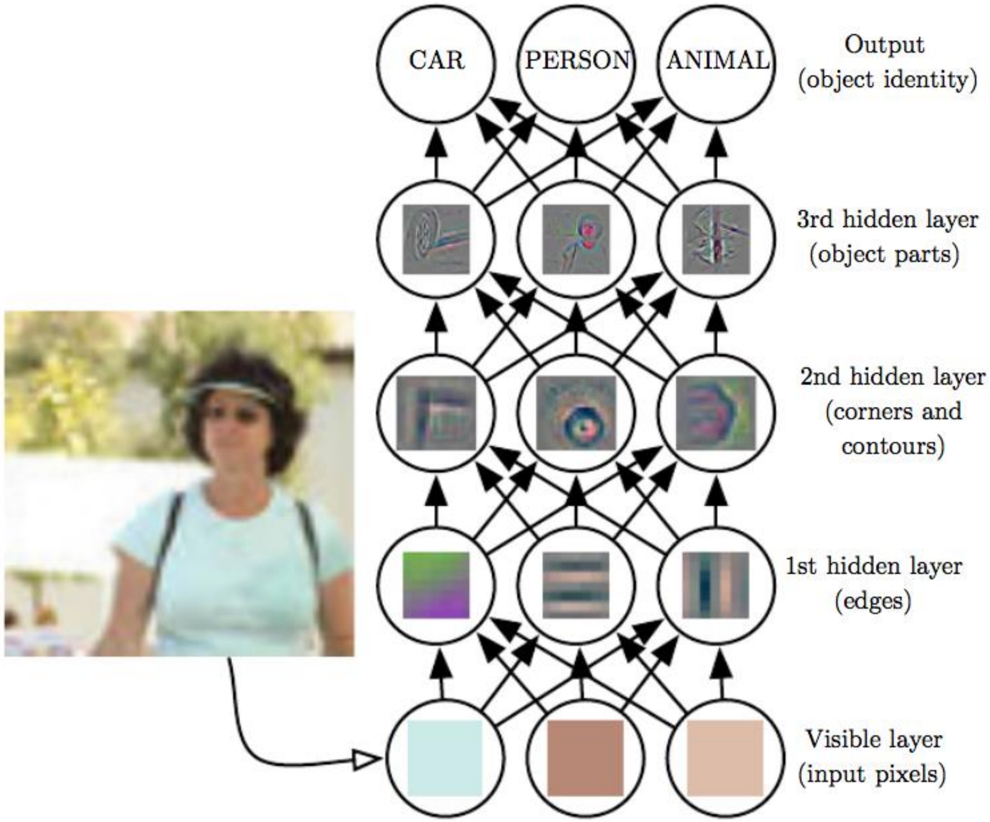
3 etapas en la investigación de RN



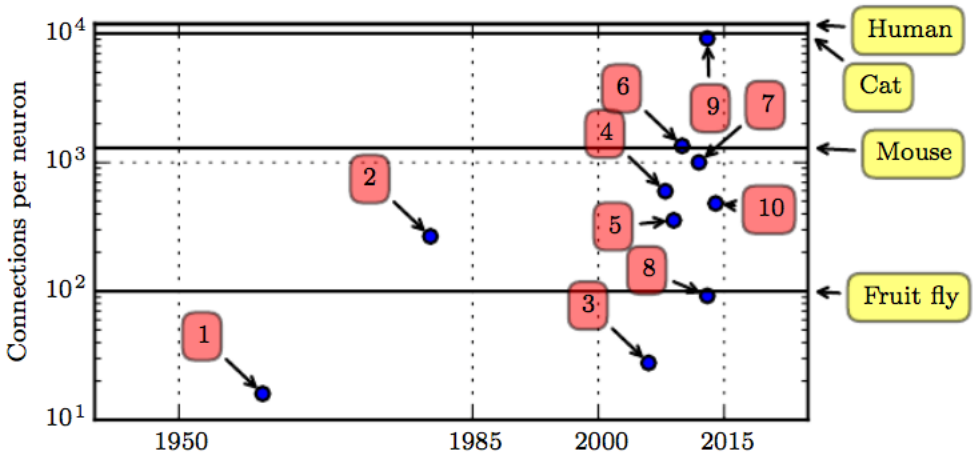
Disciplinas en inteligencia artificial



Deep learning



Número de conexiones en redes neuronales



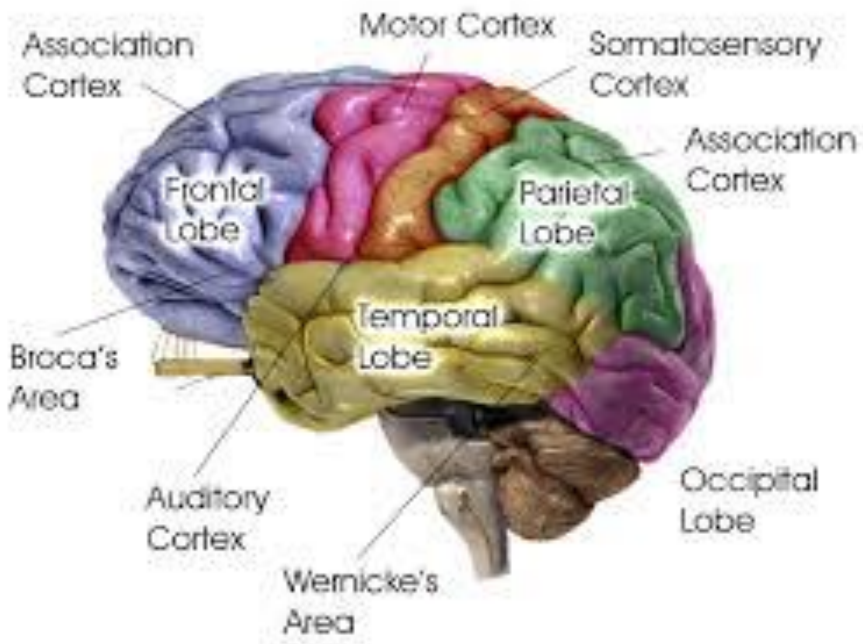
1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla *et al.*, 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett *et al.*, 2009)
6. GPU-accelerated multilayer perceptron (Ciresan *et al.*, 2010)
7. Distributed autoencoder (Le *et al.*, 2012)
8. Multi-GPU convolutional network (Krizhevsky *et al.*, 2012)
9. COTS HPC unsupervised convolutional network (Coates *et al.*, 2013)
10. GoogLeNet (Szegedy *et al.*, 2014a)

Deep learning vs. RNA

- Los conjuntos de datos etiquetados eran miles de veces más pequeños
- Las computadoras eran millones de veces más lentas
- Inicialización de los pesos aleatoriamente (se sigue haciendo pero tiende a un proceso de inicialización no supervisado).
- Se utilizaba el tipo incorrecto de función de activación no lineal

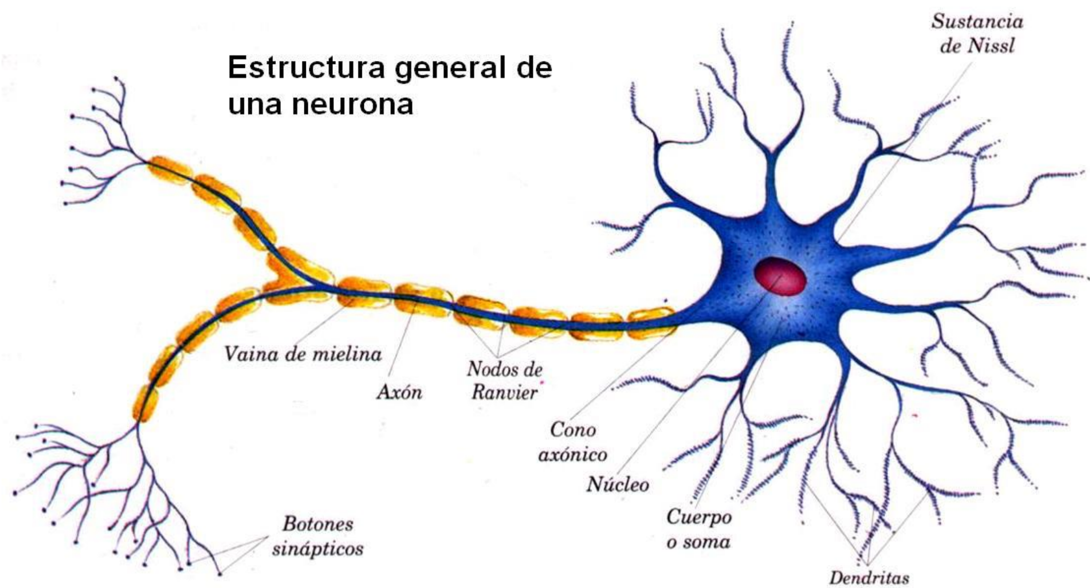
Ingeniería inversa

- Unos de los proyectos de ingeniería inversa más retadores es entender como funciona el cerebro



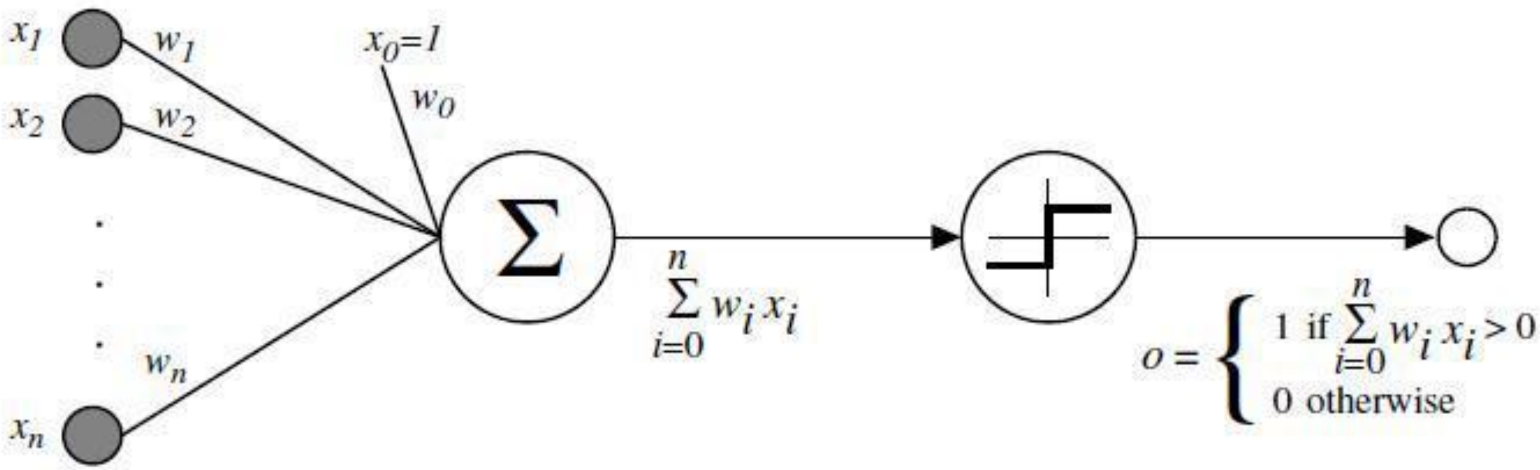
Neurona

- Las células básicas con las que procesa la información el cerebro son las neuronas
  - El cerebro humano tiene 81 mil millones de neuronas y $1.5x10^{14}$ sinapsis



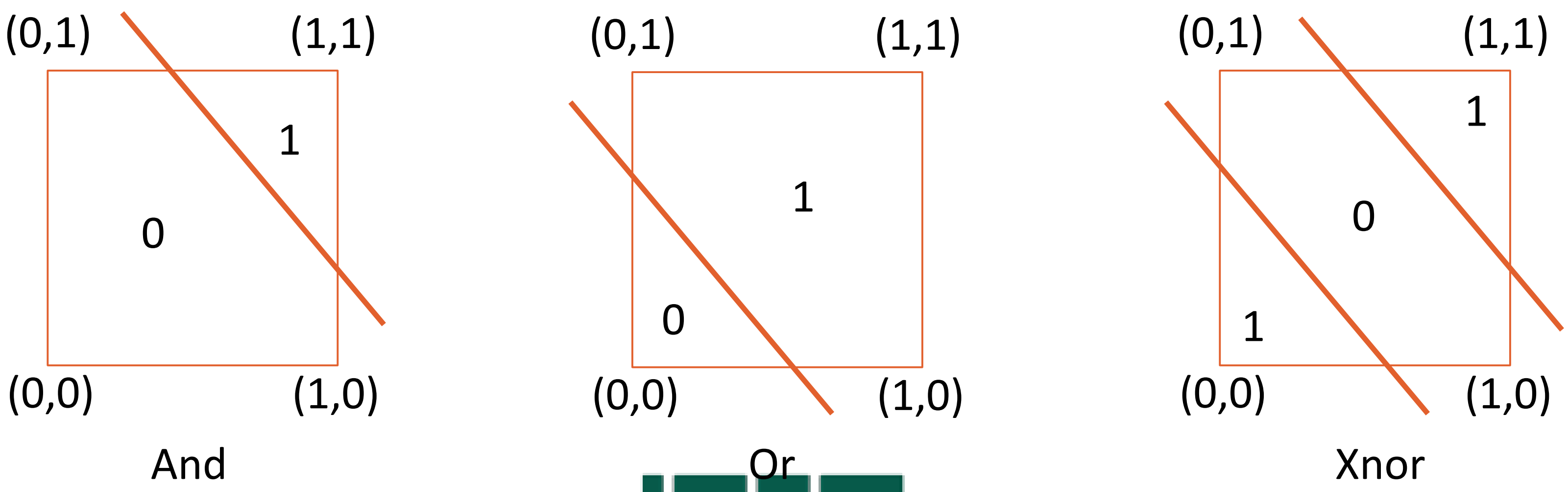
Redes neuronales

- Perceptrón



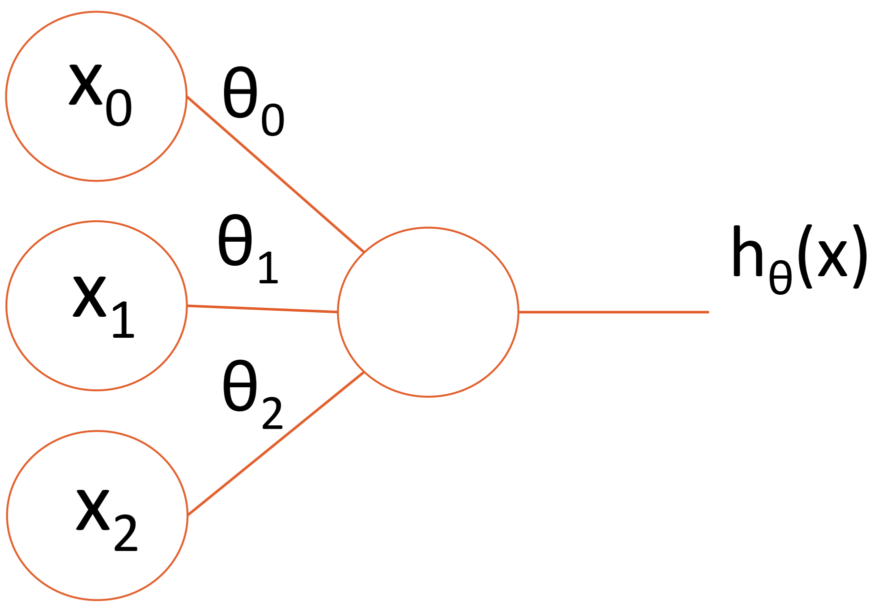
Clasificación

- Un problema tan simple como una operación XNOR no puede ser resuelto con un solo perceptrón.
  - Pero sí con varios de ellos



Modelo de neurona

$X_0 = 1$ (unidad de sesgo)

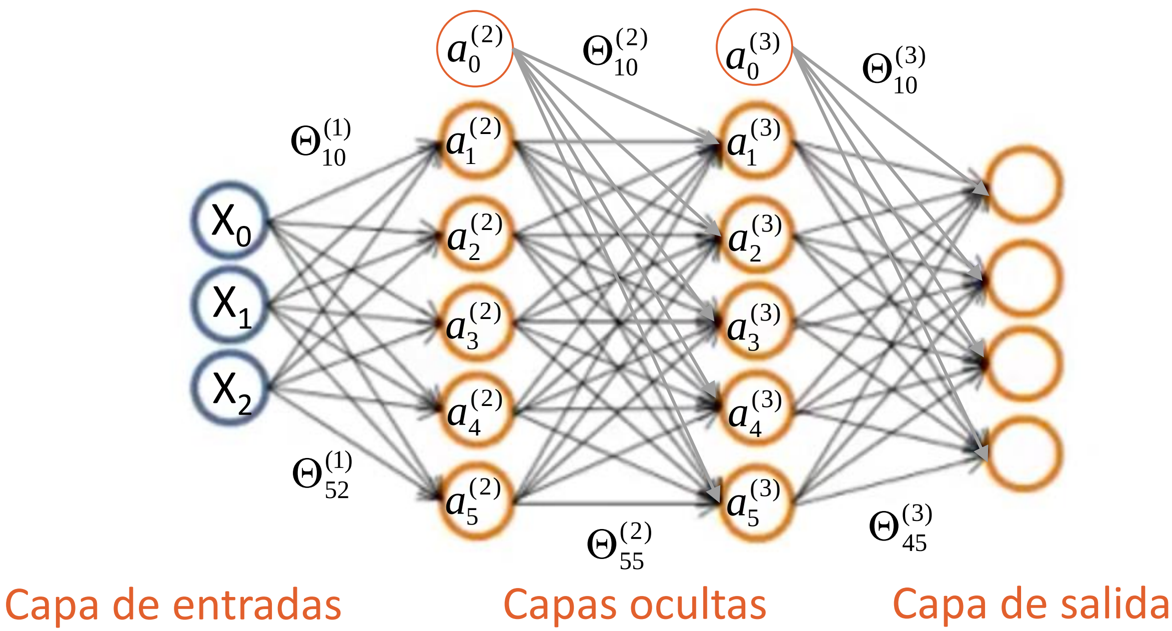


$$x = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix}$$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Función de activación sigmoidea o logística

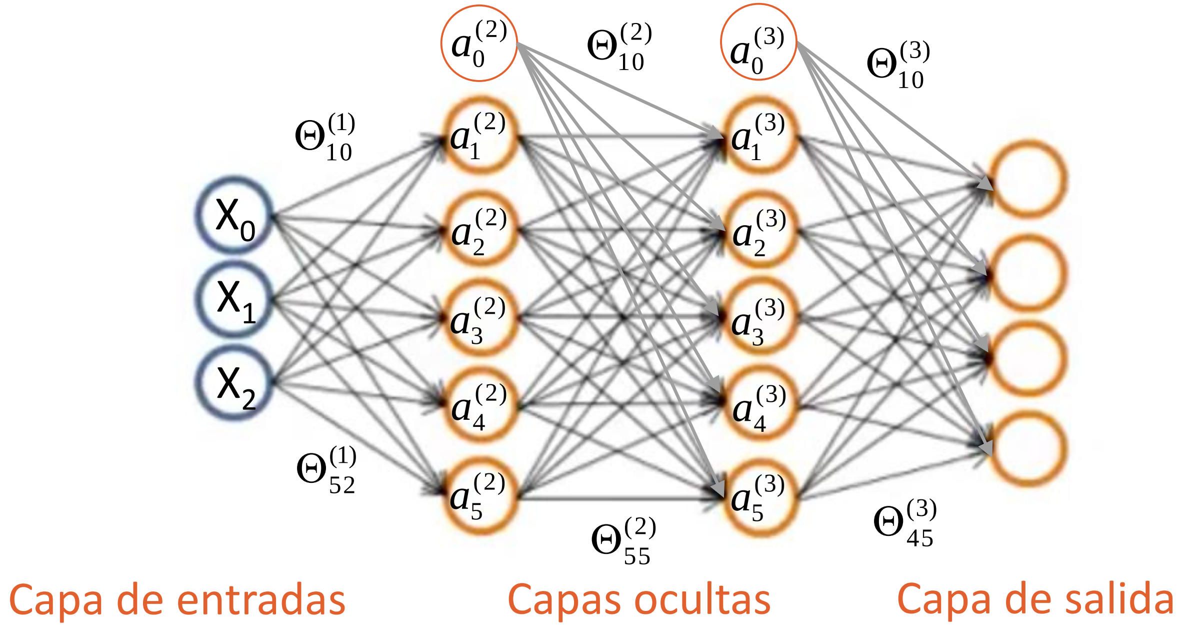
Redes neuronales



$a_i^{(j)}$ Unidad *i* de “activación” en la capa *j*

$\Theta^{(j)}$ Matriz de pesos controlando el mapeo de la función de la capa j a la j+1

Redes neuronales



$$a^{(1)} = x$$

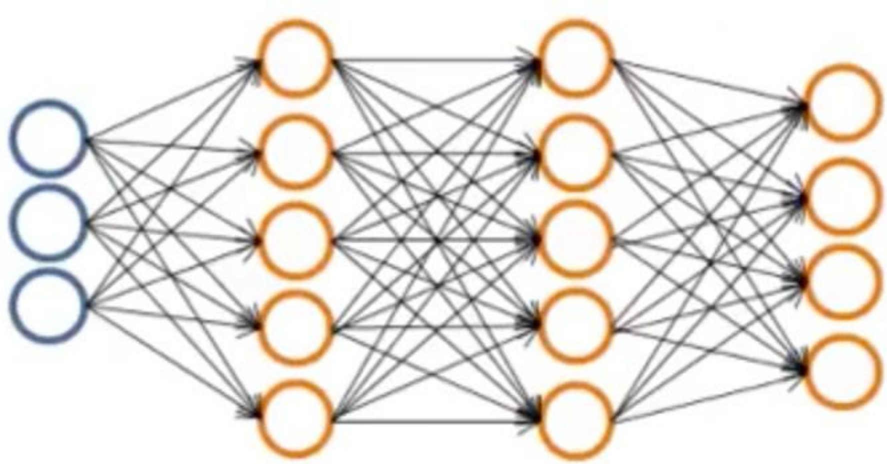
$$a^{(2)} = g(\Theta^{(1)} a^{(1)})$$

$$a^{(3)} = g(\Theta^{(2)} a^{(2)})$$

$$h_\Theta(x) = a^{(4)} = g(\Theta^{(3)} a^{(3)})$$

Clasificación multiclase

- Softmax



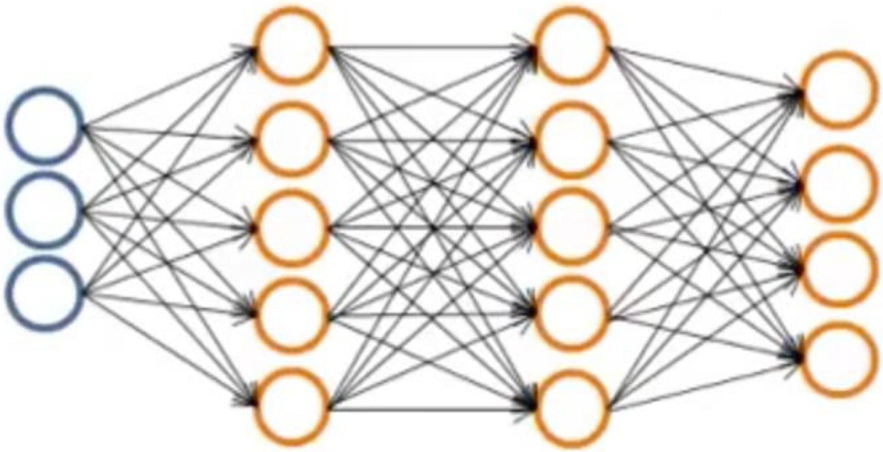
$$h_\Theta(x) \in \mathbb{R}^4$$

$$h\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad h\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \qquad h\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad h\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Clase 1 Clase 2 Clase 3 Clase 4

Clasificación multiclase

- Softmax

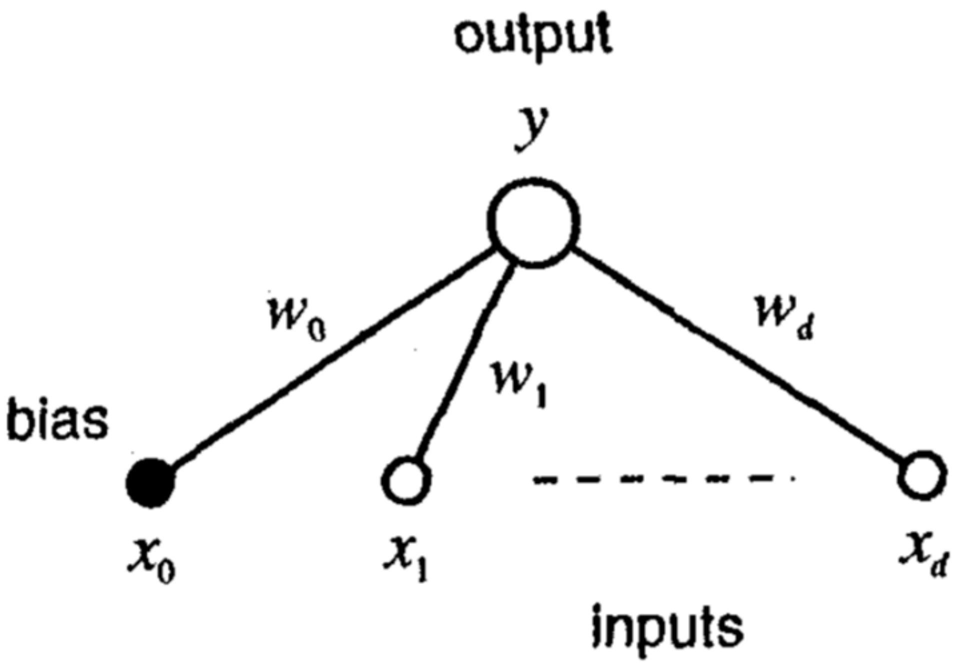


$$h_\Theta(x) \in \mathbb{R}^4$$

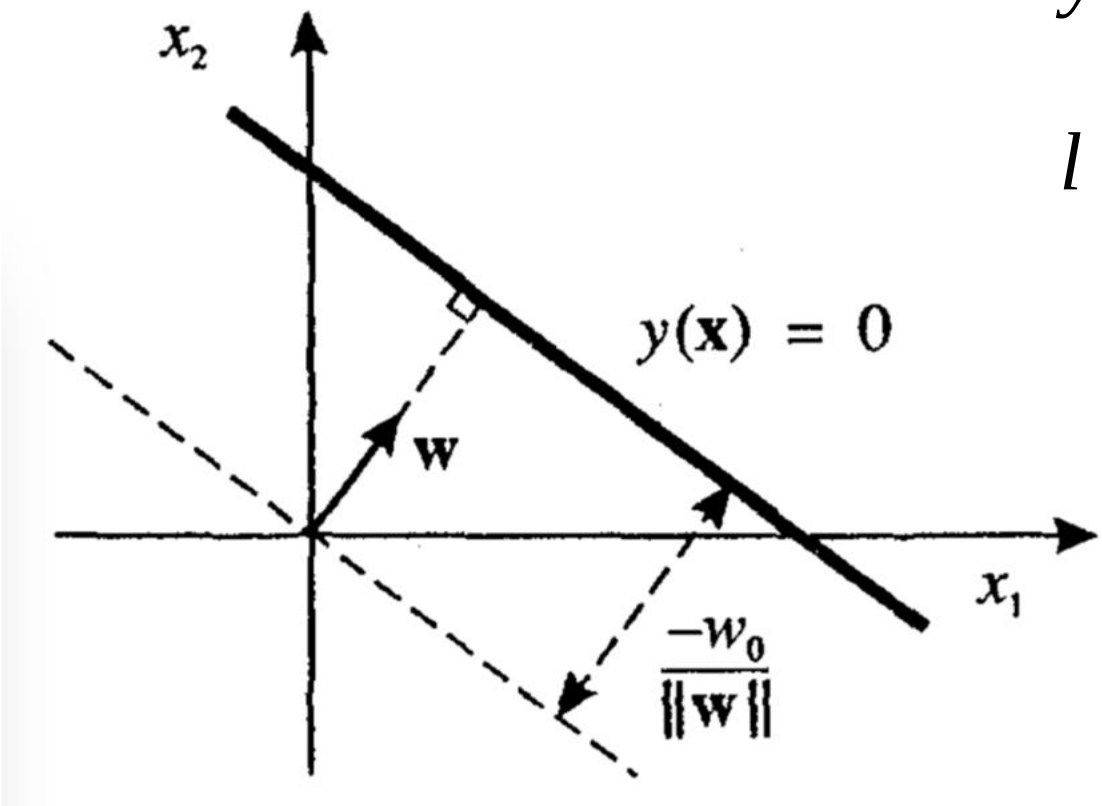
$$\sigma(x^T w)_j = P(y = j \mid x) = \frac{e^{x^T w_j}}{\sum_{i=1}^{K} e^{x^T w_i}} \quad \text{for j} = 1,\ldots,\text{K}$$

$$\frac{\partial \mathrm{E}^{\mathrm{n}}}{\partial a_k} = y_k - t_k$$

Redes de una capa



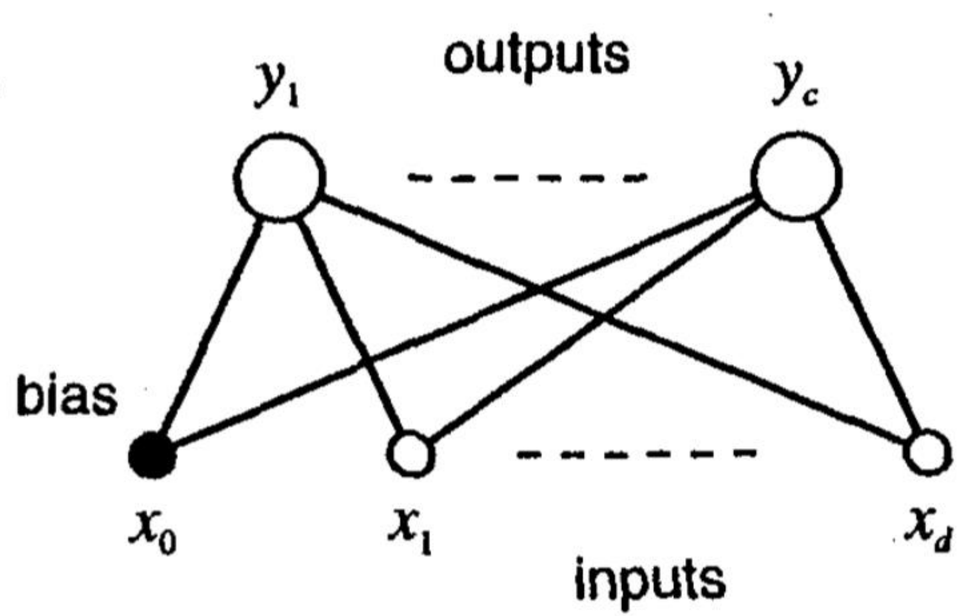
Redes de una capa



$$y(x) = w^T x + w_0$$

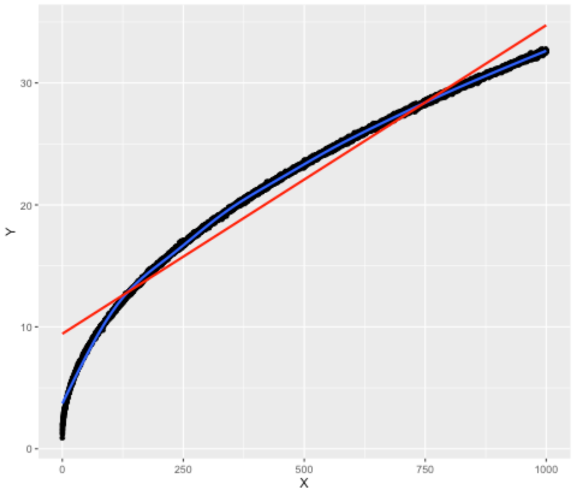
$$l = \frac{w^T x}{\|w\|} = -\frac{w_0}{\|w\|}$$

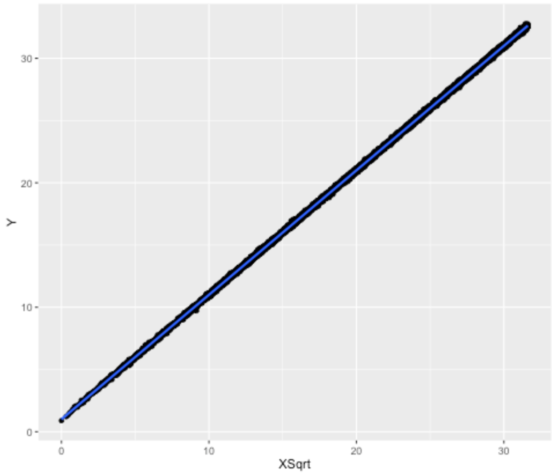
Redes de una capa



Discriminante lineal generalizado

$$h_{k\theta}(x) = \sum_{j=0}^{n} w_{kj}\phi_j(x)$$





Perceptrón

- Algoritmo de aprendizaje

Repetir hasta que converja {

$$\frac{1}{m}\sum_{i=1}^{m}\left|y^{(i)} - g(\theta^T x^{(i)})\right|$$

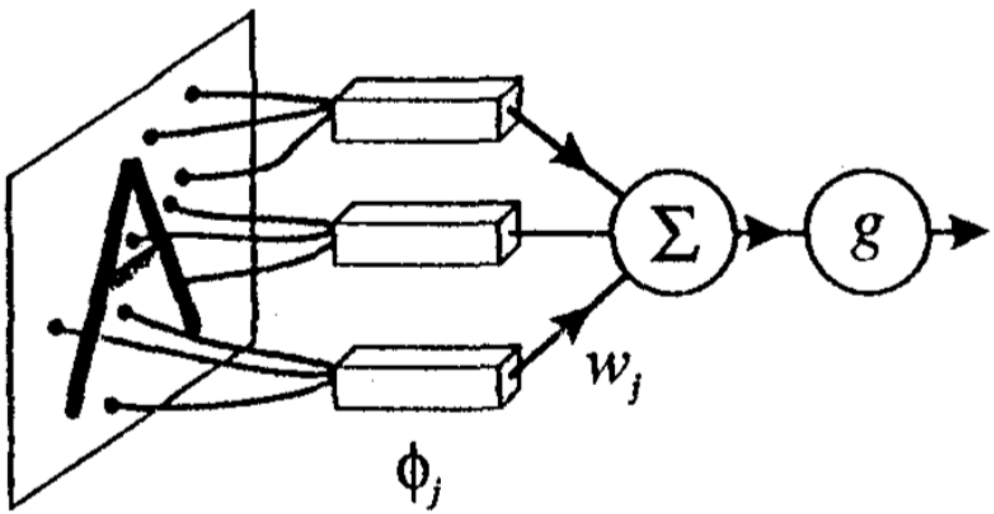
for i=0 to m {

$$\theta_j := \theta_j + (y^{(i)} - g(\theta^T x^{(i)}))\cdot\ x_j^{(i)}$$

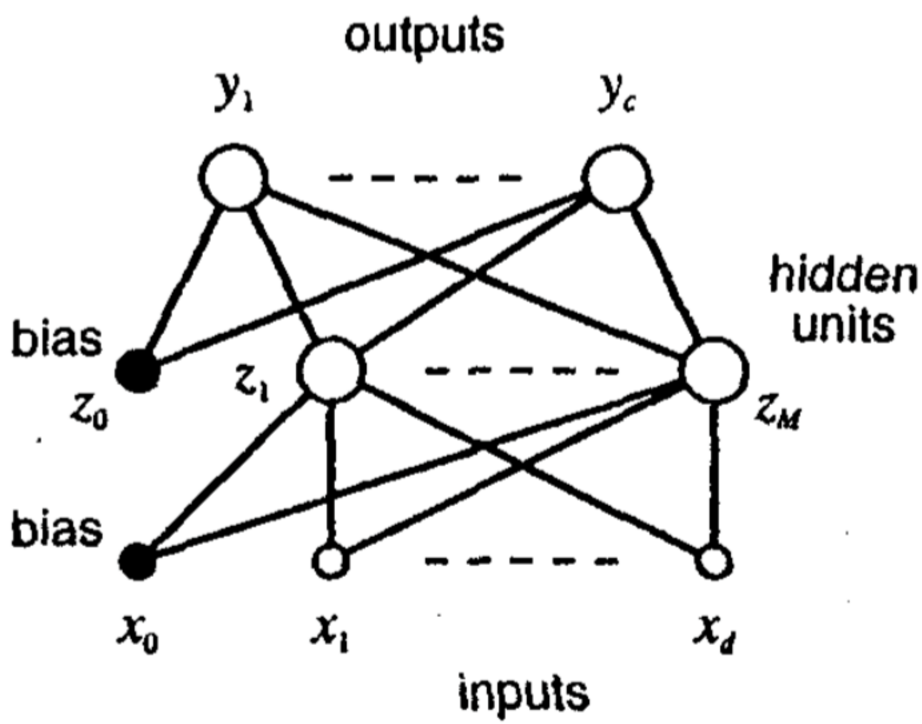
Para j=0 hasta n

}

}



Redes de múltiples capas



ITAM

Redes de múltiples capas

$$a_j = \sum_{i=0}^{d} w_{ji}^{(1)} x_i.$$

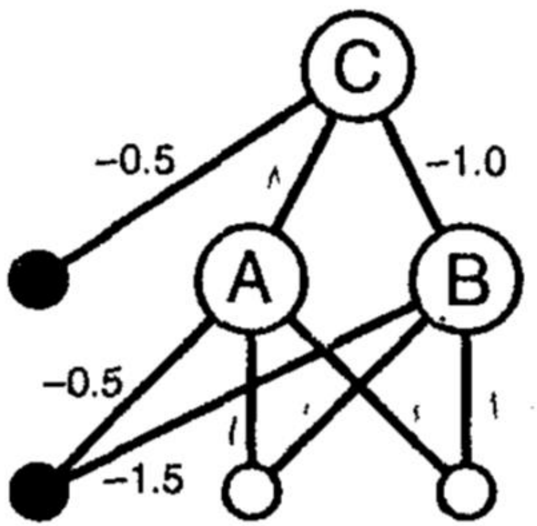
$$z_j = g(a_j).$$

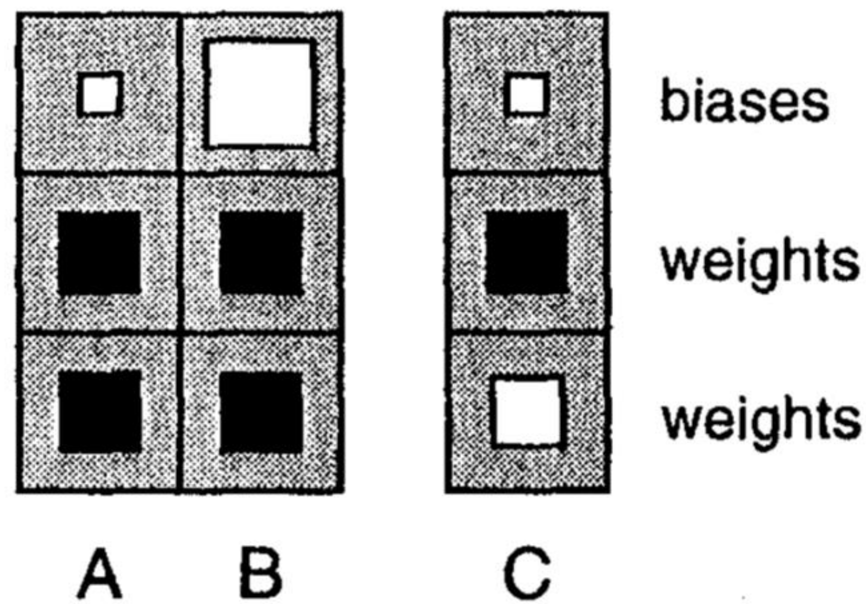
$$a_k = \sum_{j=0}^{M} w_{kj}^{(2)} z_j$$

$$y_k = \widetilde{g}(a_k).$$

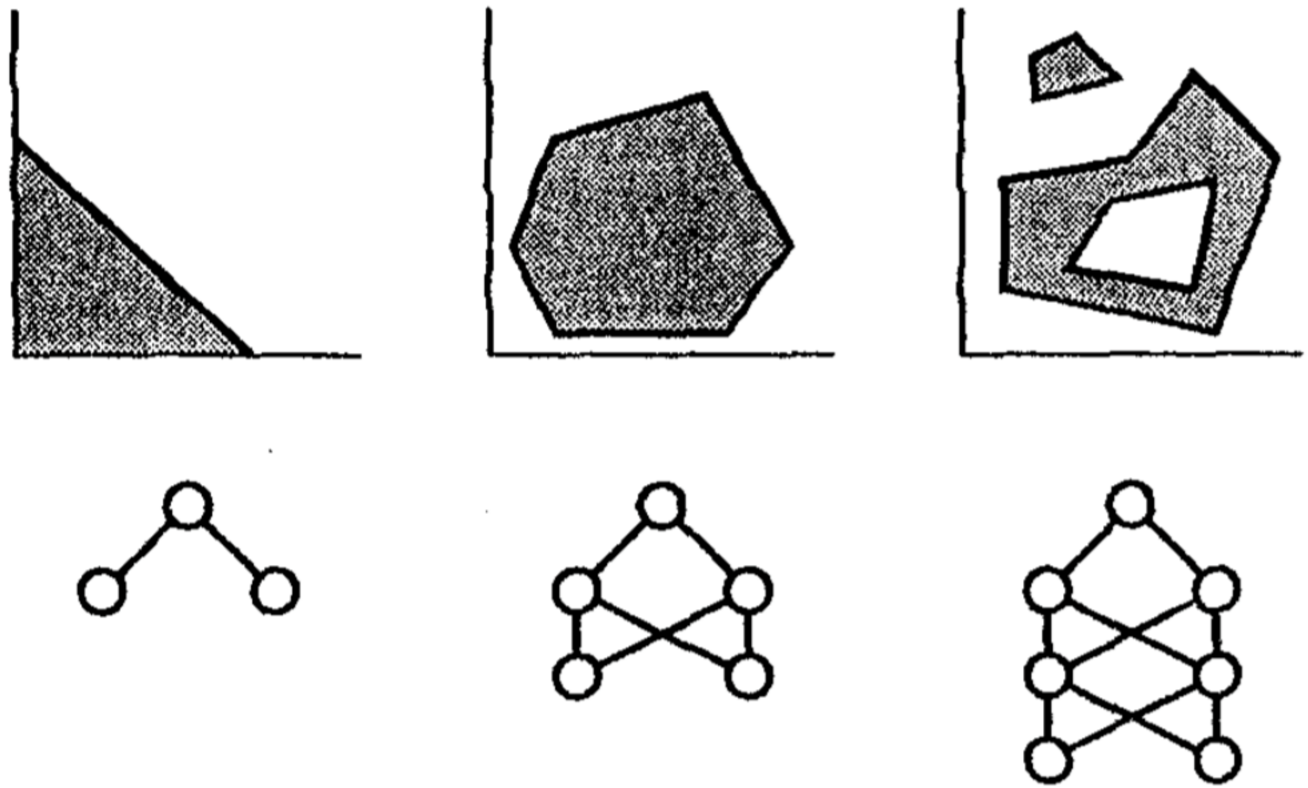
$$y_k = \widetilde{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)\right).$$

Redes de múltiples capas

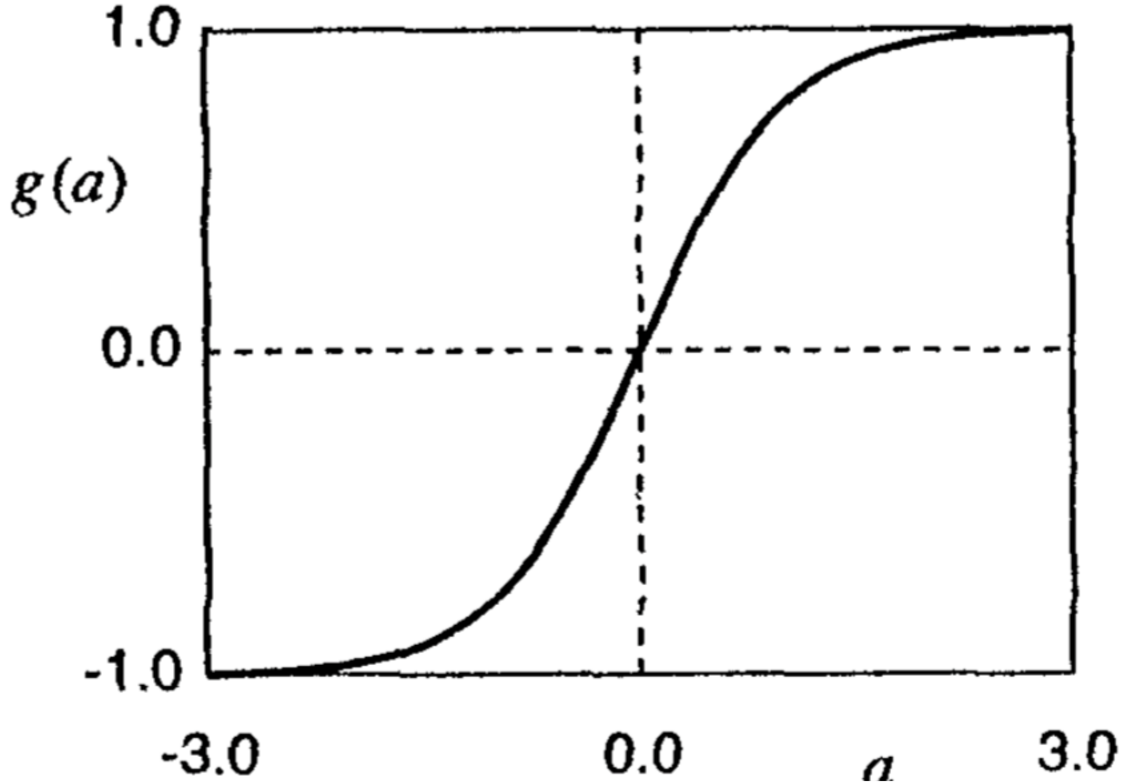




Redes de múltiples capas



Redes de múltiples capas



$$g(a) \equiv \tanh(a) \equiv \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

ITAM

Descenso por gradiente (incremental)

- Algoritmo de descenso por gradiente para regresión lineal

Repetir hasta que converja {

for i=1 to m {

$$\theta_j := \theta_j - \alpha \cdot (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$ Para j=0 hasta n

}

}

Back propagation

$$a_j = \sum_i w_{ji} z_i$$

$$E = \sum_n E^n$$ n datos de entrenamiento

$$E^n = E^n(y_1, \ldots, y_c).$$

Usando la regla de la cadena

$$\frac{\partial E^n}{\partial w_{ji}} = \frac{\partial E^n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}.$$

$$\delta_j \equiv \frac{\partial E^n}{\partial a_j}$$

$$\frac{\partial a_j}{\partial w_{ji}} = z_i.$$

$$\frac{\partial E^n}{\partial w_{ji}} = \delta_j z_i.$$

$$\delta_k \equiv \frac{\partial E^n}{\partial a_k} = g'(a_k) \frac{\partial E^n}{\partial y_k}$$

Usando la regla de la cadena

$$\delta_j \equiv \frac{\partial E^n}{\partial a_j} = \sum_k \frac{\partial E^n}{\partial a_k} \frac{\partial a_k}{\partial a_j}$$

Inicialización de pesos

- Hallazgos empíricos indican que para funciones de activación logística o tangente hiperbólica, se obtienen tasas mas bajas de error y menor tiempo de convergencia si los pesos se inicializan de la siguiente forma:

$$W \sim U\left[-\frac{\sqrt{6}}{n^{(l)}+n^{(l+1)}}, \frac{\sqrt{6}}{n^{(l)}+n^{(l+1)}}\right]$$

$n^{(l)}$ número de entradas hacia W
$n^{(l+1)}$ número de salidas desde W

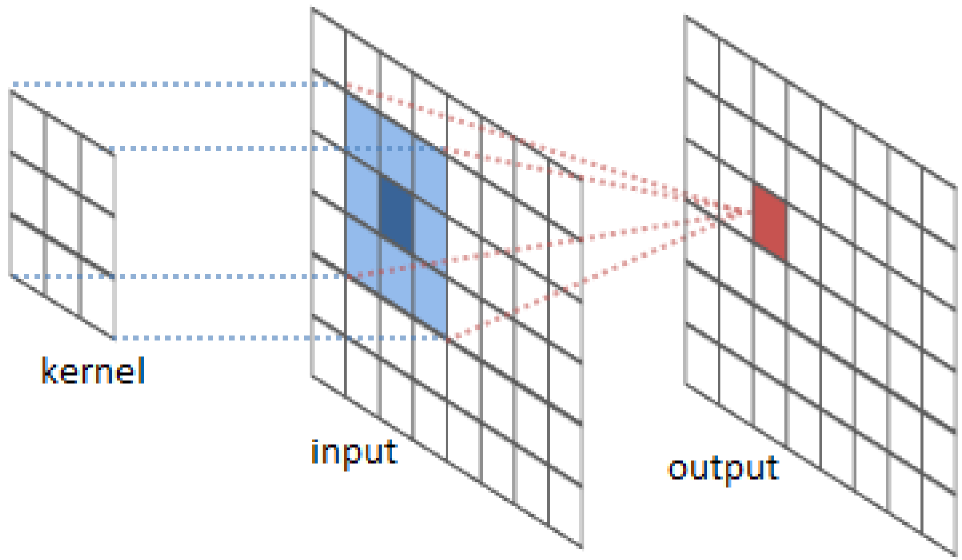
Redes neuronales convolucionales

- La convolución es una integral que expresa la cantidad de superposición de una función g mientras se desplaza sobre otra función f.
  - Por lo tanto, "mezcla” una función con otra.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau$$

Redes neuronales convolucionales

- Podemos pensar que las imágenes son funciones bidimensionales
  - Varias transformaciones de imágenes son convoluciones de una imagen con un kernel

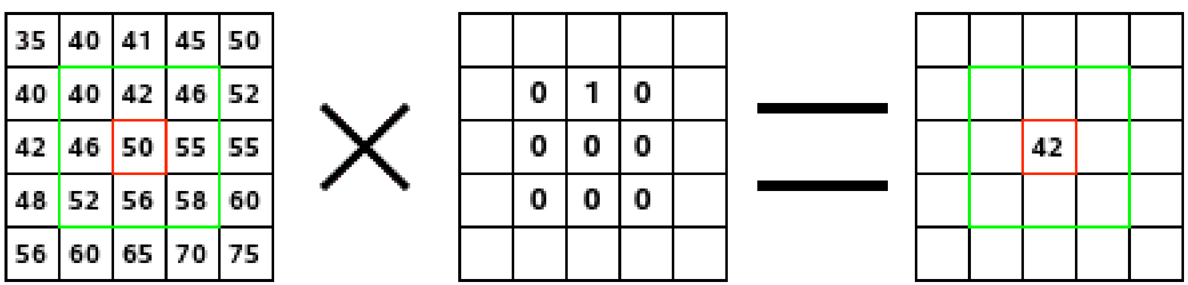


Redes neuronales convolucionales

- Un ejemplo simple:

  (40*0)+(42*1)+(46*0) + (46*0)+(50*0)+(55*0) + (52*0)+(56*0)+(58*0) = 42

  – Sólo movió el pixel superior un reglón abajo



Redes neuronales convolucionales

- Difuminar (promediando pixeles)

0	0	0	0	0
0	1/9	1/9	1/9	0
0	1/9	1/9	1/9	0
0	1/9	1/9	1/9	0
0	0	0	0	0

Redes neuronales convolucionales

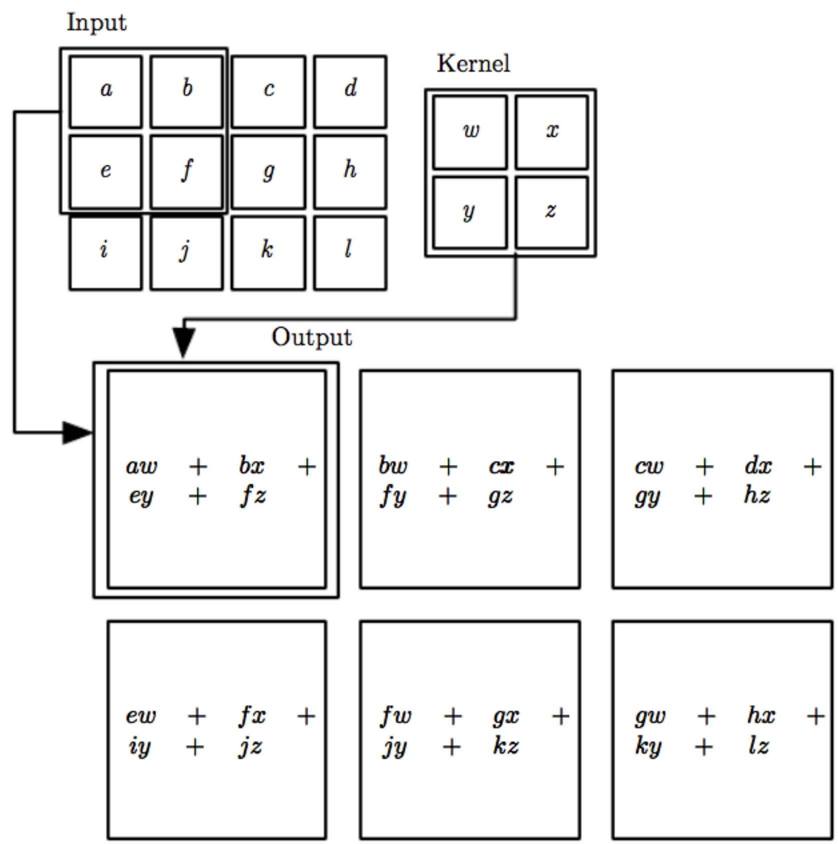
- Detección de bordes (llevando a valores cercanos a cero pixeles vecinos similares)

0	0	0	0	0
0	0	0	0	0
0	-1	1	0	0
0	0	0	0	0
0	0	0	0	0

Redes neuronales convolucionales

- Correlación cruzada

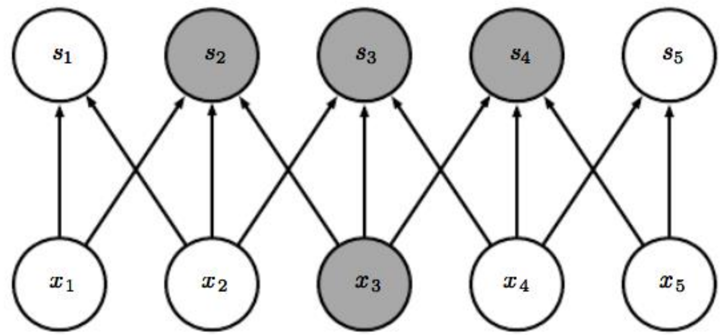
$$(I * K)(i, j) = \sum_m \sum_n I(i + m, j + n) K(m, n)$$



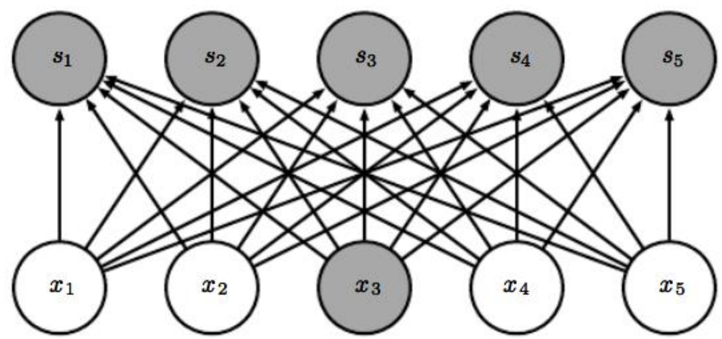
Redes neuronales convolucionales

- Conectividad poco densa (vista desde la capa inferior)

Kernel de tamaño 3



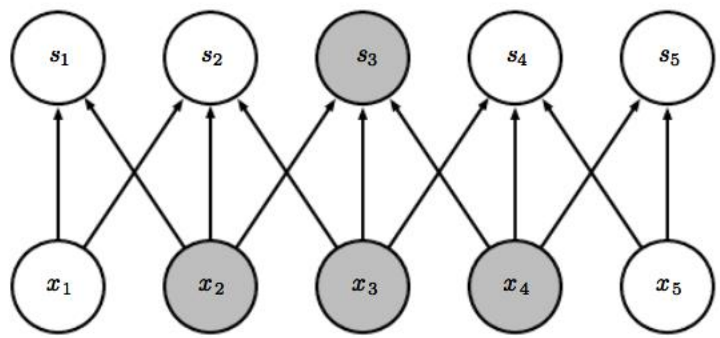
Conectividad completa



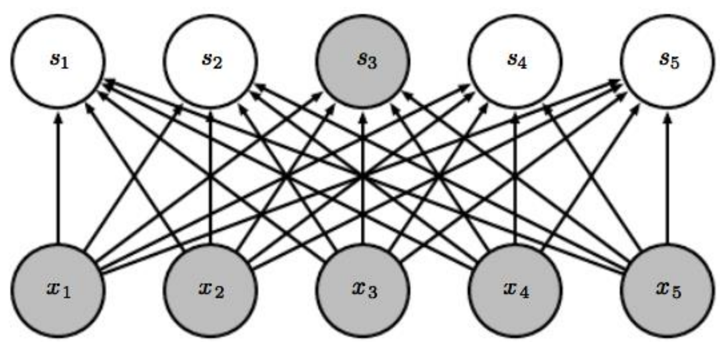
Redes neuronales convolucionales

- Conectividad poco densa (vista desde la capa superior)

Kernel de tamaño 3

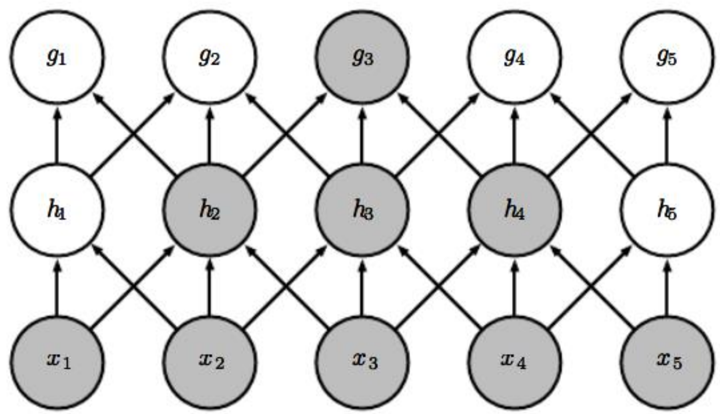


Conectividad completa



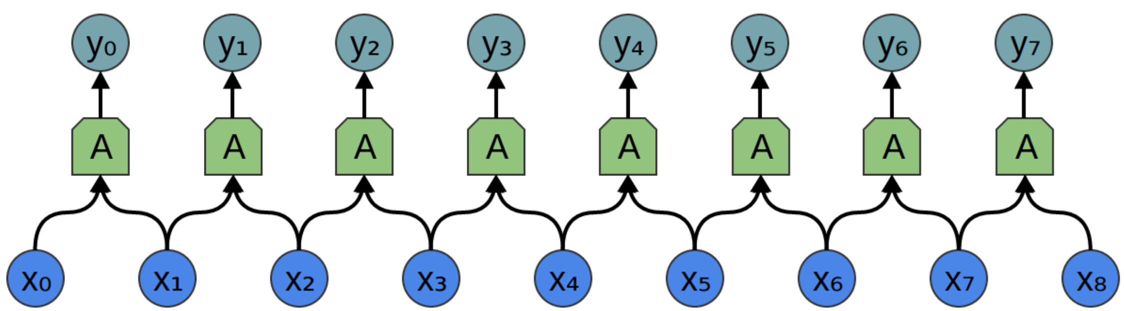
Redes neuronales convolucionales

- Neuronas de las capas más profundas están conectadas a un mayor número de entradas indirectamente.



Redes neuronales convolucionales

- En su forma más básica son una especie de red neuronal que utiliza muchas copias idénticas de la misma neurona.
  - Esto permite que la red tenga muchas neuronas y mantener un número bastante pequeño de parámetros que deben ser aprendidos.



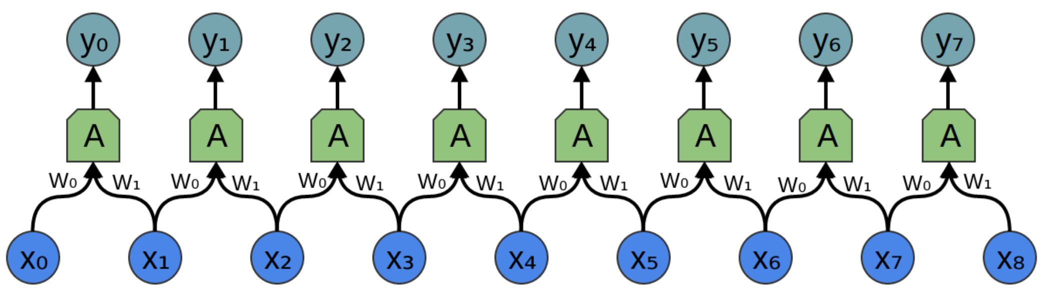
ITAM

Redes neuronales convolucionales

- Como hay múltiples copias de la misma neurona, los mismos pesos aparecen en múltiples posiciones:

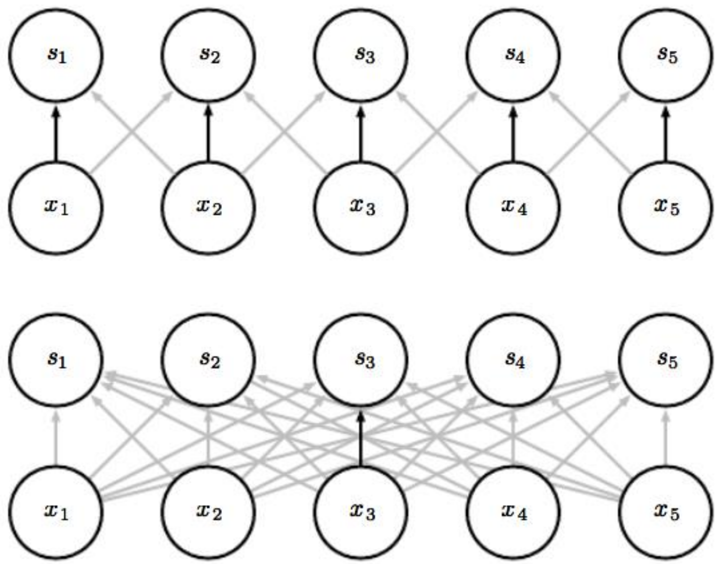
$$y_0 = \sigma(W_0 x_0 + W_1 x_1 - b)$$

$$y_1 = \sigma(W_0 x_1 + W_1 x_2 - b)$$



Redes neuronales convolucionales

- Parámetros compartidos



Da a lugar a equivarianza a la translación
(si la entrada cambia, la salida cambia de la misma forma

Redes neuronales convolucionales

- En vez de tener matrices de pesos que conecten un vector de entrada con todos las neuronas de la siguiente capa:

$$W = \begin{bmatrix} W_{0,0} & W_{0,1} & W_{0,2} & W_{0,3} & \dots \\ W_{1,0} & W_{1,1} & W_{1,2} & W_{1,3} & \dots \\ W_{2,0} & W_{2,1} & W_{2,2} & W_{2,3} & \dots \\ W_{3,0} & W_{3,1} & W_{3,2} & W_{3,3} & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

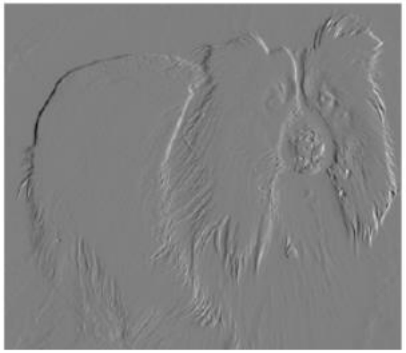
- Tenemos:

$$W = \begin{bmatrix} w_0 & w_1 & 0 & 0 & \dots \\ 0 & w_0 & w_1 & 0 & \dots \\ 0 & 0 & w_0 & w_1 & \dots \\ 0 & 0 & 0 & w_0 & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

ITAM

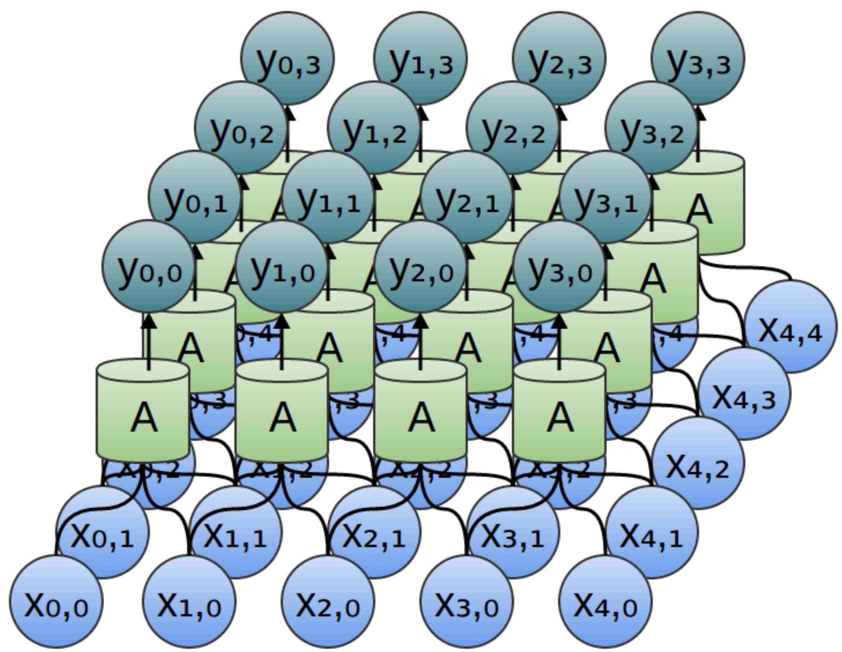
Redes neuronales convolucionales

- Esto permite obtener filtros de forma muy eficiente:
  - Convolución: ancho x largo x 3 opf
  - Multiplicación de matrices: ancho x largo x largo x ancho opf



Redes neuronales convolucionales

- Para procesamiento de imagen usamos una capa convolucional de 2 dimensiones



Redes neuronales convolucionales

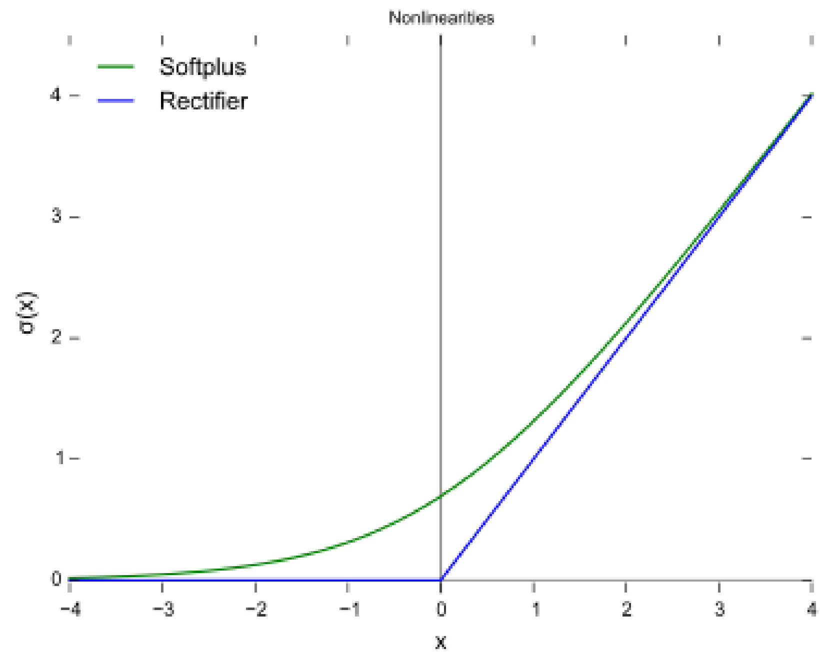
- Unidad lineal rectificada (ReLU)

Rampa

$$f(x) = \max(0, x)$$

Softplus

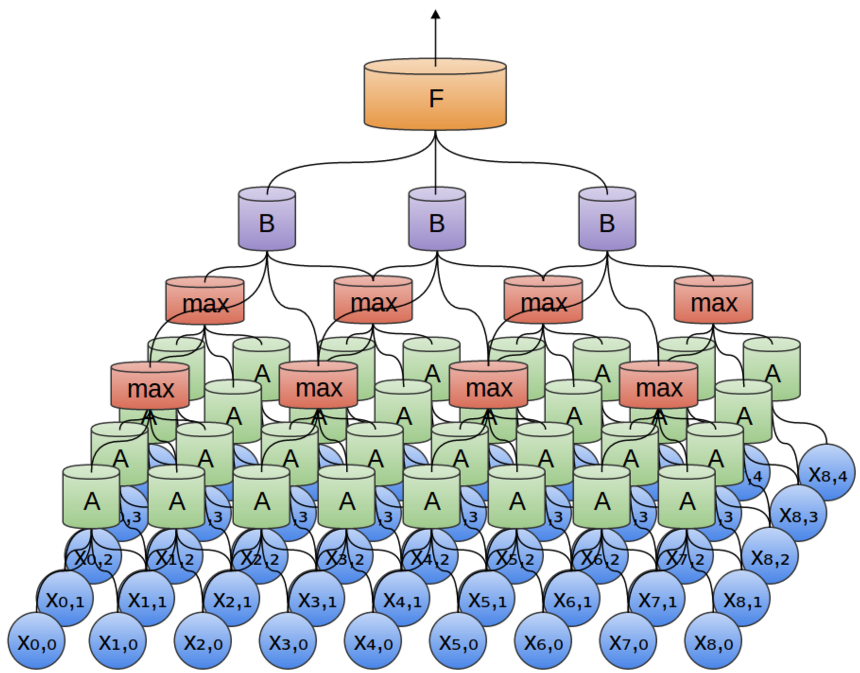
$$f(x) = \ln(1 + e^{x})$$



Redes neuronales convolucionales

- Reducción de resolución mediante max-pooling para resumir características sobre una región

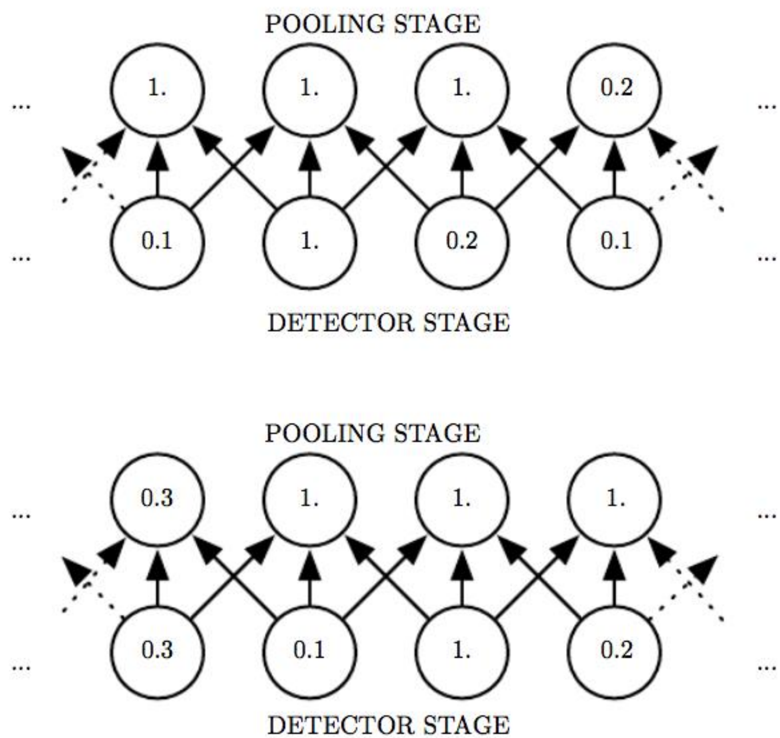
Es más importante conocer si se dio una característica que justo el lugar en donde se dio



Hace la arquitectura más tolerante a pequeñas translaciones

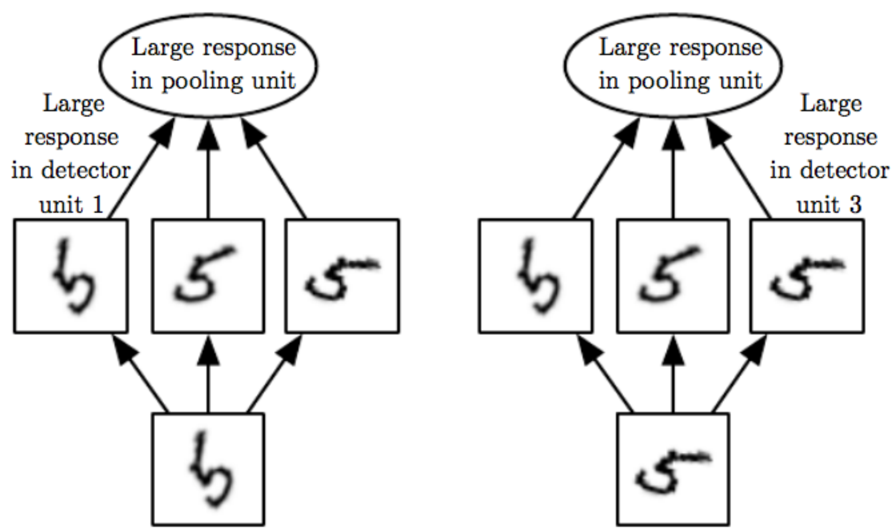
Redes neuronales convolucionales

- Tolerancia a la translación



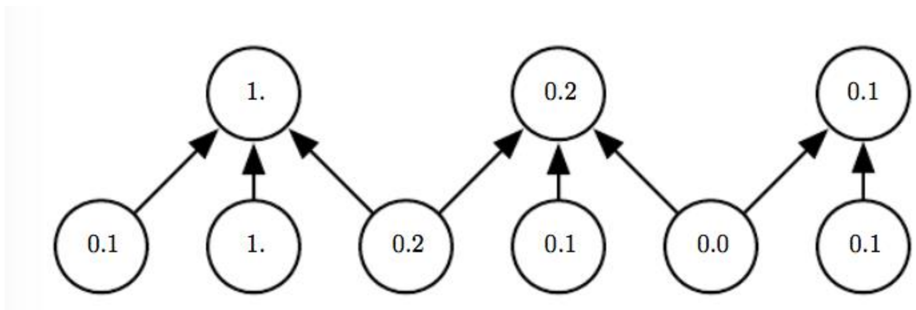
Redes neuronales convolucionales

- Invarianzas aprendidas



Redes neuronales convolucionales

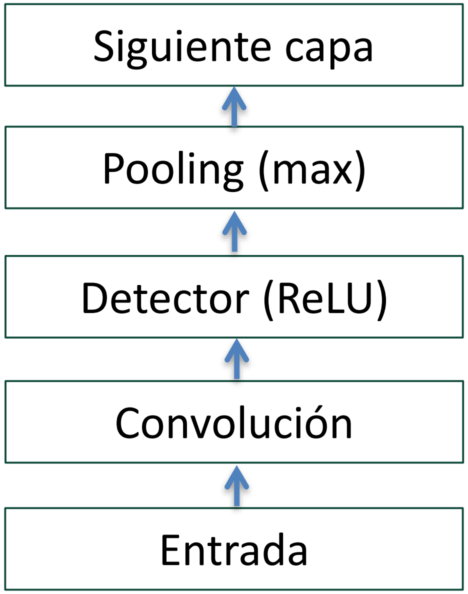
- Reducción de la resolución



Kernel = 3
Paso entre unidades pooling (Stride) = 2

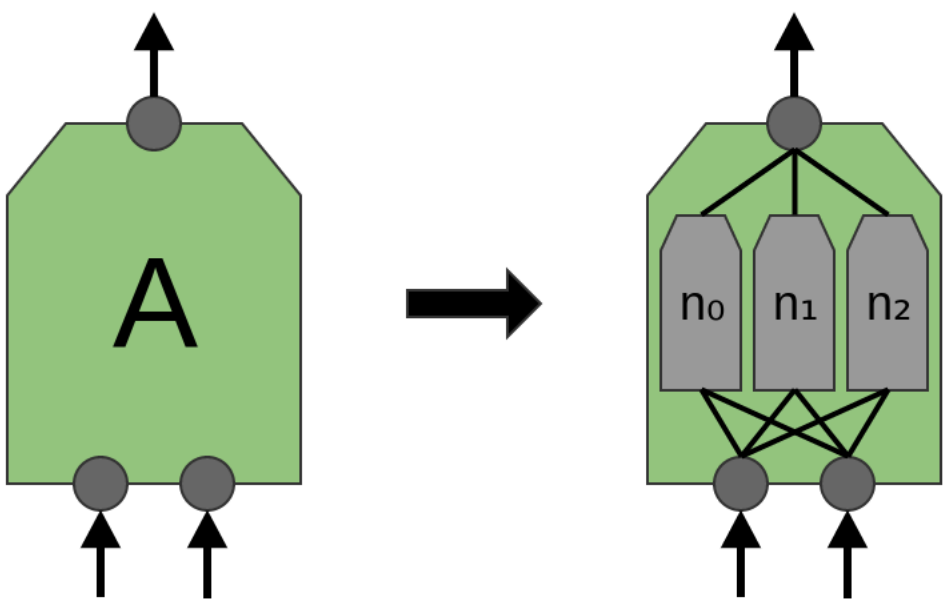
Redes neuronales convolucionales

- Arquitectura común

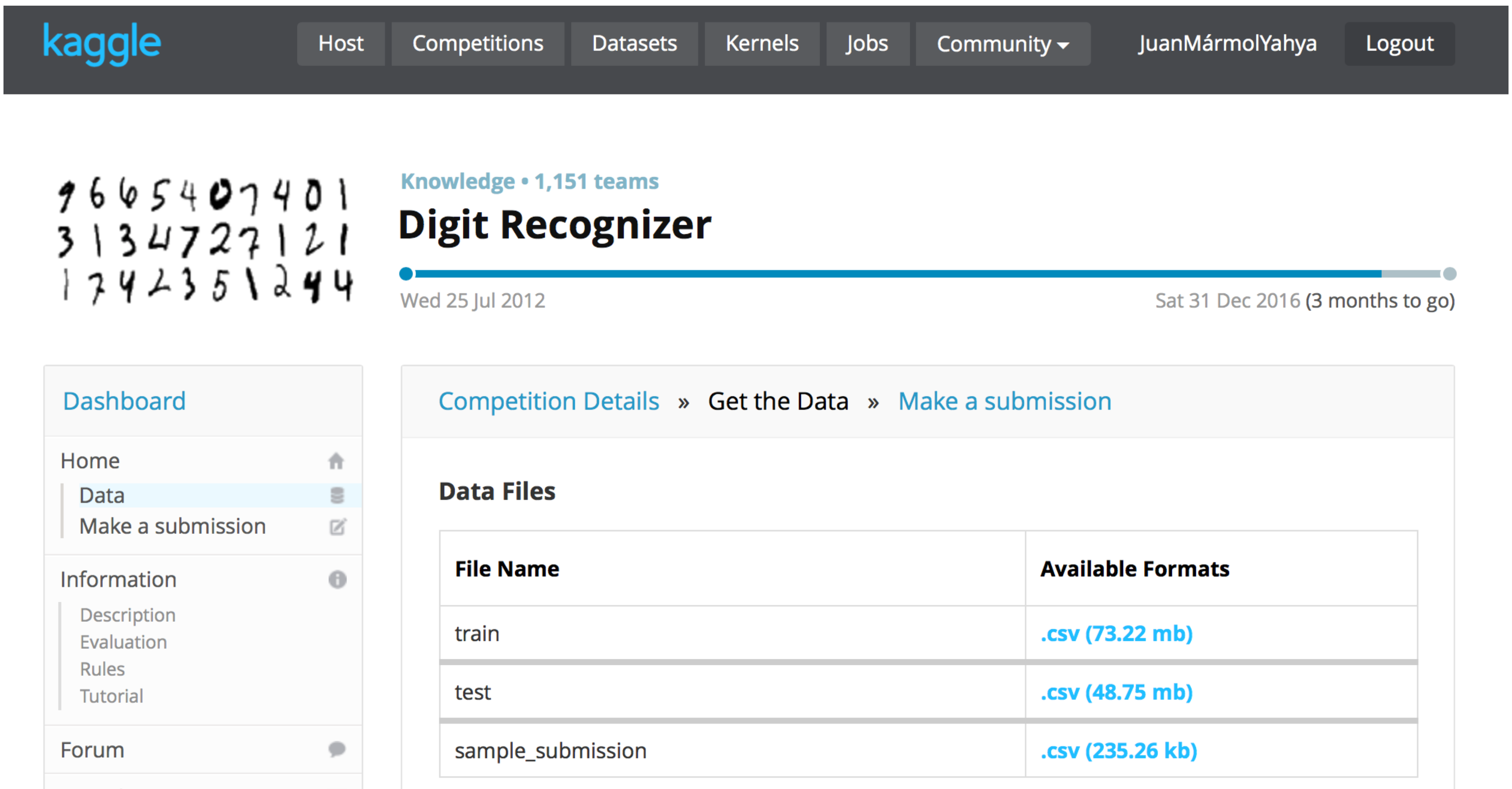


Redes neuronales convolucionales

- Normalmente A es una serie de neuronas en paralelo en donde todas tienen las mismas entradas y obtienen diferentes características



Kaggle


kaggle
Host
Competitions
Datasets
Kernels
Jobs
Community
JuanMármolYahya
Logout
Knowledge • 1,151 teams
Digit Recognizer
Wed 25 Jul 2012
Sat 31 Dec 2016 (3 months to go)
Dashboard
Home
Data
Make a submission
Information
Description
Evaluation
Rules
Tutorial
Forum
Kernels
Competition Details » Get the Data » Make a submission
Data Files
File Name
Available Formats
train
.csv (73.22 mb)
test
.csv (48.75 mb)
sample_submission
.csv (235.26 kb)

Redes neuronales recurrentes

- Son una familia de redes neuronales que procesan datos secuenciales
- Al igual que las redes convolucionales tienen parámetros compartidos lo que les permite generalizar:
  - distintos tamaños de secuencia
  - posiciones distintas de la secuencia

ITAM

Redes neuronales recurrentes

- Son una familia de redes neuronales que procesan datos secuenciales
- Al igual que las redes convolucionales tienen parámetros compartidos lo que les permite generalizar:
  - distintos tamaños de secuencia
  - posiciones distintas de la secuencia

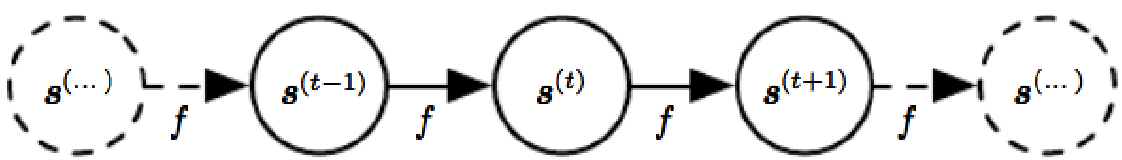
Redes neuronales recurrentes

- Grafo computacional del despliegue del sistema dinámico:

Para τ = 3

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta})$$

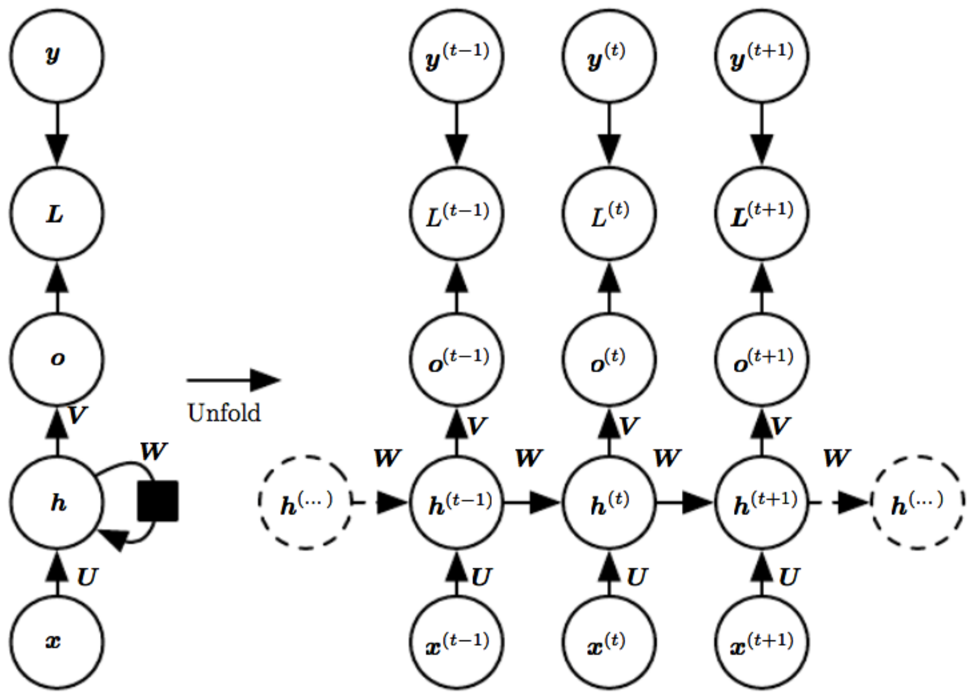
$$\begin{aligned}\boldsymbol{s}^{(3)} &= f(\boldsymbol{s}^{(2)}; \boldsymbol{\theta}) \\ &= f(f(\boldsymbol{s}^{(1)}; \boldsymbol{\theta}); \boldsymbol{\theta})\end{aligned}$$



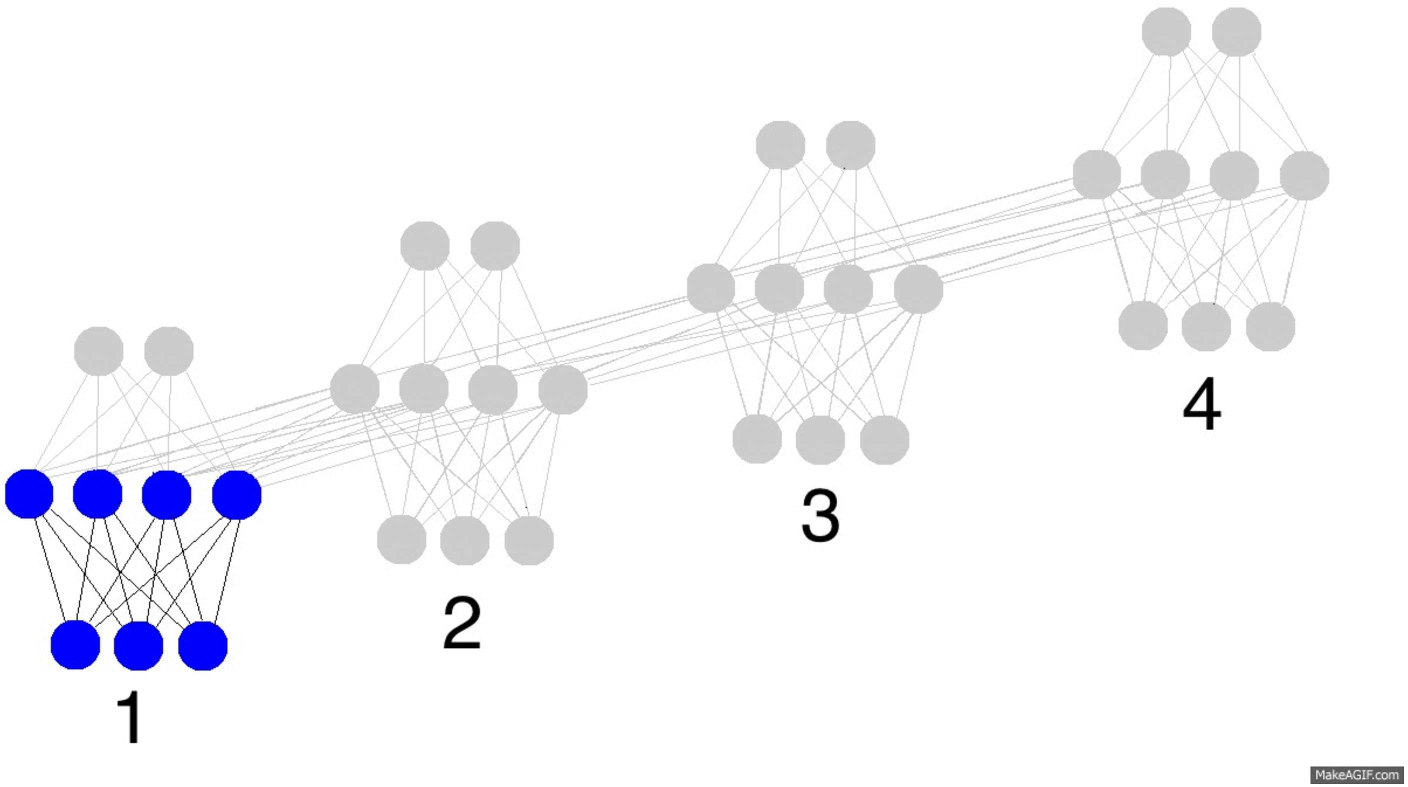
$s^{(t)}$ = estado en tiempo t

Redes neuronales recurrentes

- Una salida en cada paso y conexiones recurrentes en la capa oculta

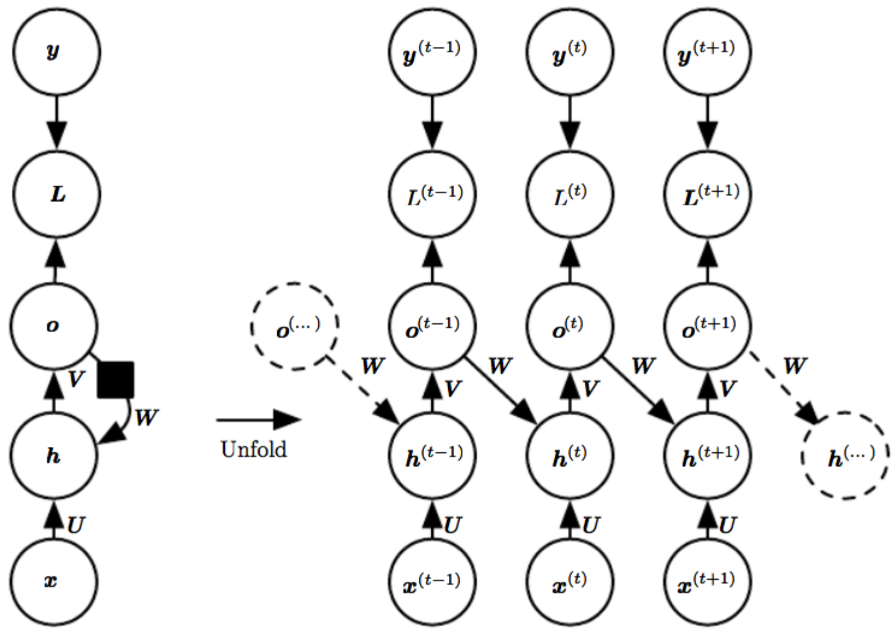


Redes neuronales recurrentes



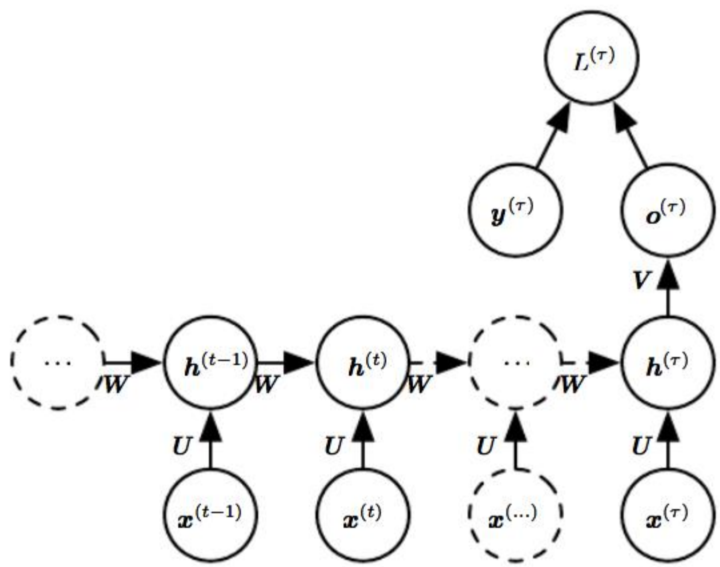
Redes neuronales recurrentes

- Una salida en cada paso y conexiones recurrentes de la salida en el tiempo t a la capa oculta en el tiempo t+1



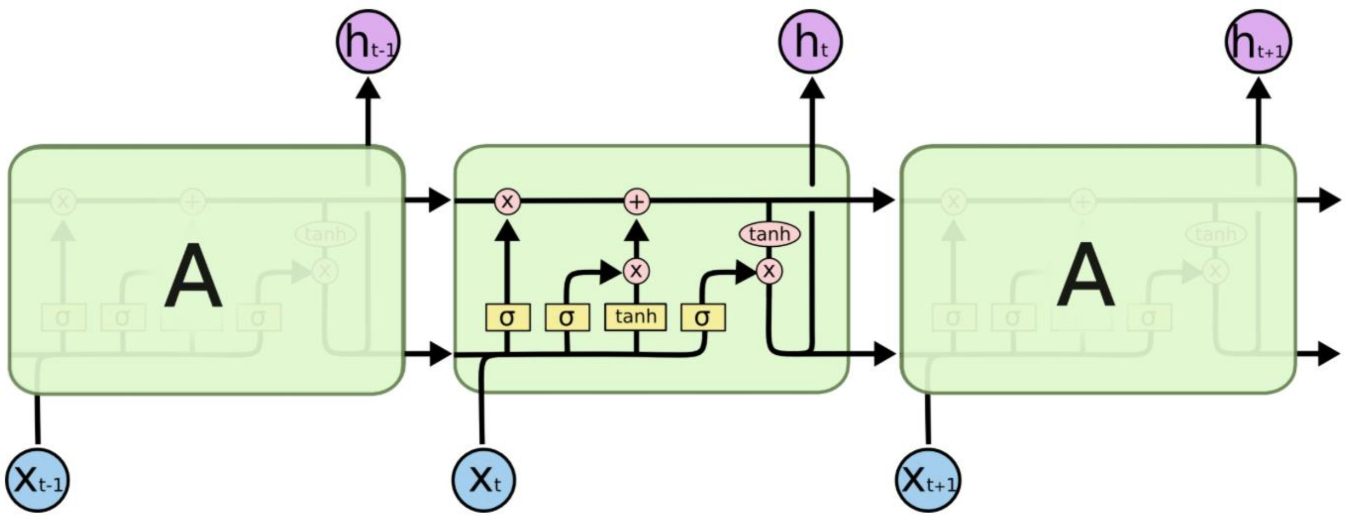
Redes neuronales recurrentes

- Una salida al final de la secuencia y conexiones recurrentes en la capa oculta



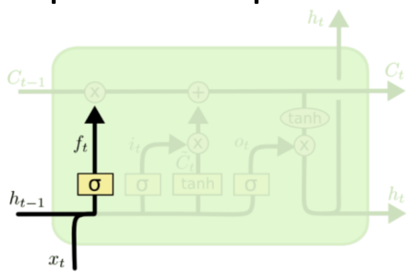
Long Short Term Memory (LSTM)

- Un tipo de red neuronal recurrente capaz de aprender dependencia de largo plazo



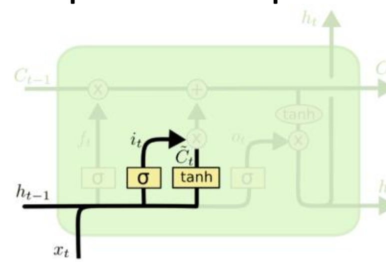
Long Short Term Memory (LSTM)

Capa de compuertas para olvidar



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

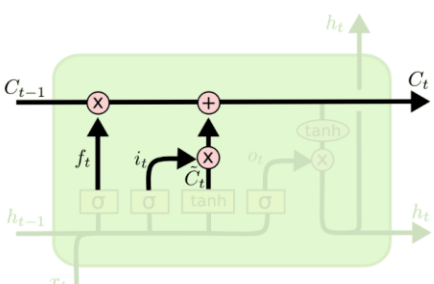
Capa de compuertas de entrada



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

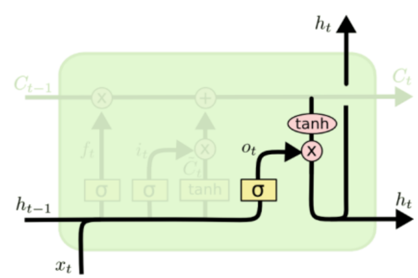
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Cálculo del estado de la celda



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Cálculo de la salida



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

ITAM

TensorFlow

- TensorFlow es una biblioteca de software de código abierto para el cálculo numérico usando diagramas de flujo de datos.
- Desarrollada originalmente por Google Brain para conducir investigaciones de aprendizaje máquina y, en específico, deep neural networks.

Instalación de TensorFlow

- pip install --upgrade pip
- pip install tensorflow==1.13.1

Regresión Softmax

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

import tensorflow as tf
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
#mnist.train.images
#mnist.train.labels

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

Keras

- Keras es una biblioteca de redes neuronales minimalista, altamente modular, escrita en Python capaz de ejecutarse en utilizando TensorFlow o Theano.
- Fue desarrollado con el objetivo de facilitar la experimentación rápida.
  - Ser capaz de ir de la idea al resultado con el menor retraso posible es clave para hacer una buena investigación
- pip install keras

Word Embeding

- Método de aprendizaje no supervisado que transforma un conjunto de palabras en vectores de altas dimensiones
- Basado en el modelo distribucional
  - Palabras que aparecen en los mismos contextos comparten significado semántico
- Los vectores tiene la propiedad de que conceptos similares se agrupan

  Similitud entre Hombre y Mujer: 0.74
- Se pueden realizar operaciones entre los vectores

  Rey - Hombre + Mujer = Reina

IMDB corpus

- 25,000 opiniones de películas de IMDB etiquetadas con su sentimiento asociado (positivo con calificación >=7 / negativo con calificación <= 4).
- Cada revisión se codifica como una secuencia de índices de palabras (enteros).
- Por conveniencia, las palabras son indexados por frecuencia global en el conjunto de datos.
  - Se puede filtrar la información rápidamente

ITAM