Inteligencia Artificial

Prof. Dra. Claudia Pons

LIFIA- UNLP

- **Temas:** Inteligencia Artificial no-simbólica

✓ Machine learning.

✓ Proceso de desarrollo de los sistemas Machine learning

✓ Redes neuronales artificiales

- **Bibliografía:**

✓ Inteligencia Artificial. Un enfoque moderno. Russell y Norvig.

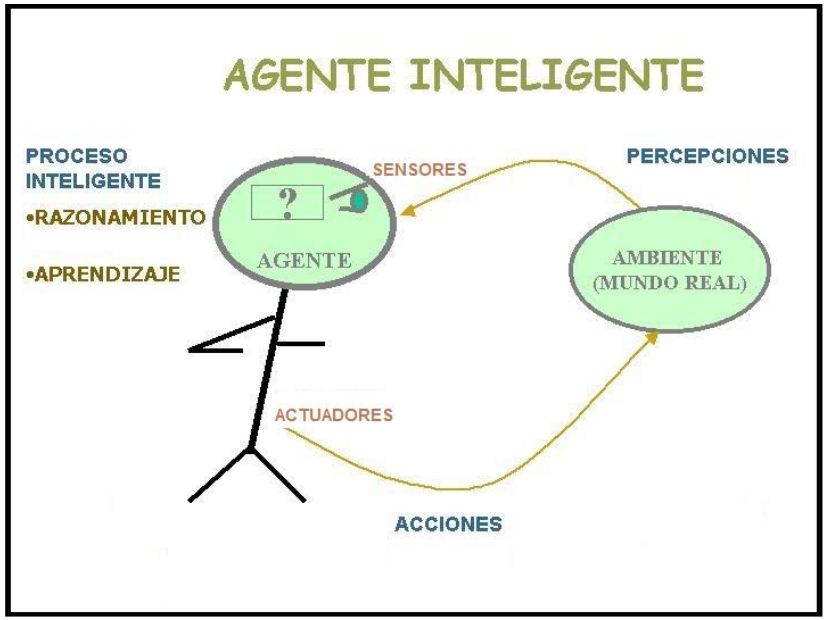
✓ Understanding Deep Learning by Simon J. D. Prince Publisher: The MIT Press.

- **Trabajo práctico**

✓ TP09

Entonces, qué es Inteligencia Artificial?

Se denomina **Inteligencia Artificial (IA)** a la rama de las ciencias de la Computación dedicada al estudio y desarrollo de **agentes racionales computacionales**
(**Artificial Intelligence: A Modern Approach** by Stuart Russell, Peter Norvig. Google Research and Stanford University)



Agente:
cualquier cosa capaz de percibir su entorno (recibir entradas), procesar tales percepciones y actuar en su entorno (proporcionar salidas).

Racionalidad:
capacidad humana de pensar y actuar conforme a principios de optimalidad y consistencia para alcanzar un objetivo. El ejercicio de la racionalidad está sujeto a mejora continua (aprendizaje).

Qué es Inteligencia Artificial?

IA = Racionalidad (actuar con optimalidad y consistencia) + Aprender

¿Son IA todos los sistemas computacionales que aprenden y realizan tareas con optimalidad y consistencia?

Es IA cuando la máquina no sigue un algoritmo (una receta) para resolver un problema, sino que encuentra su propia forma de resolverlo.

Dos enfoques diferentes

IA Simbólica.
EL PROGRAMADOR CONOCE LAS REGLAS
Ej. Sistemas expertos basados en inferencia lógica (razonan por deducción)

IA no-Simbólica.
EL PROGRAMADOR DESCONOCE LAS REGLAS
Machine learning (razonan por inducción)

- **Redes neuronales**
- Arboles de decisión
- Vectores de soporte

IA Simbólica: un ejemplo

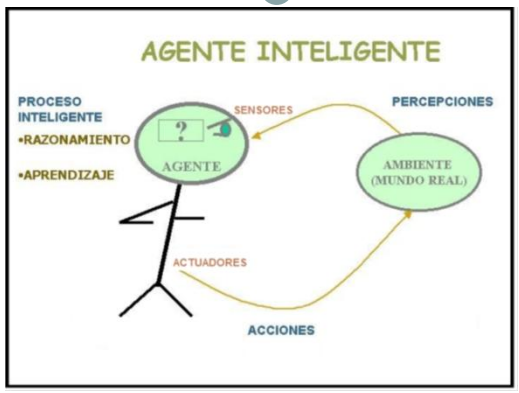
La programación lógica implementa las formas básicas del razonamiento lógico, por ejemplo el Modus ponens. A, $(A \rightarrow B) \therefore B$

Se Java y me gusta programar

¿soy apto como programador?

R1: es_apto(Persona, Rol) :- prefiere_rol(Persona, Rol), posee_skill(Persona, Skill), es_para(Skill, Rol).

Si, señor.
Es apto.

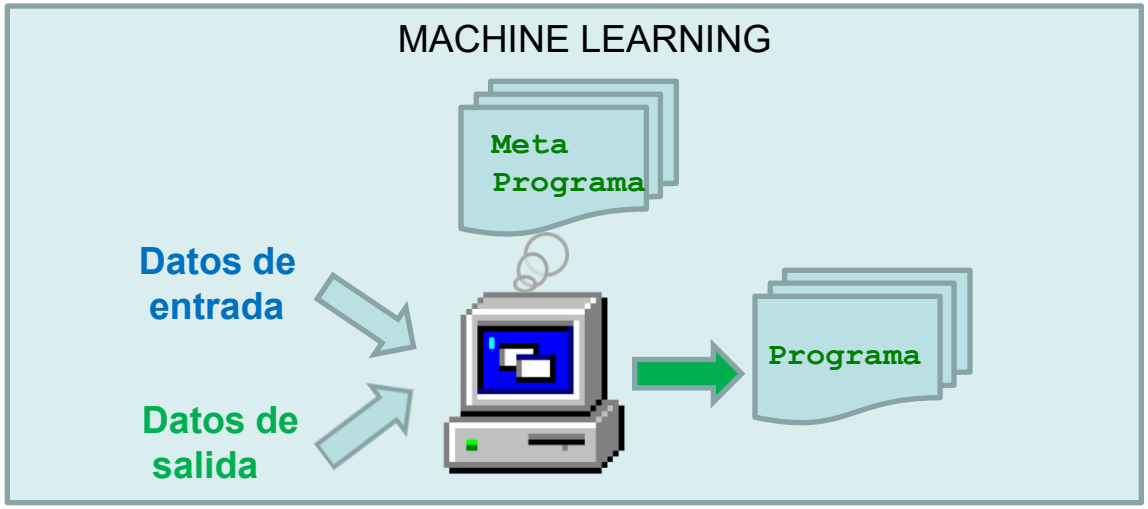


Machine Learning

El nuevo paradigma de programación

Machine Learning

Con Machine Learning la máquina construye (aprende) su propio algoritmo para resolver un problema.

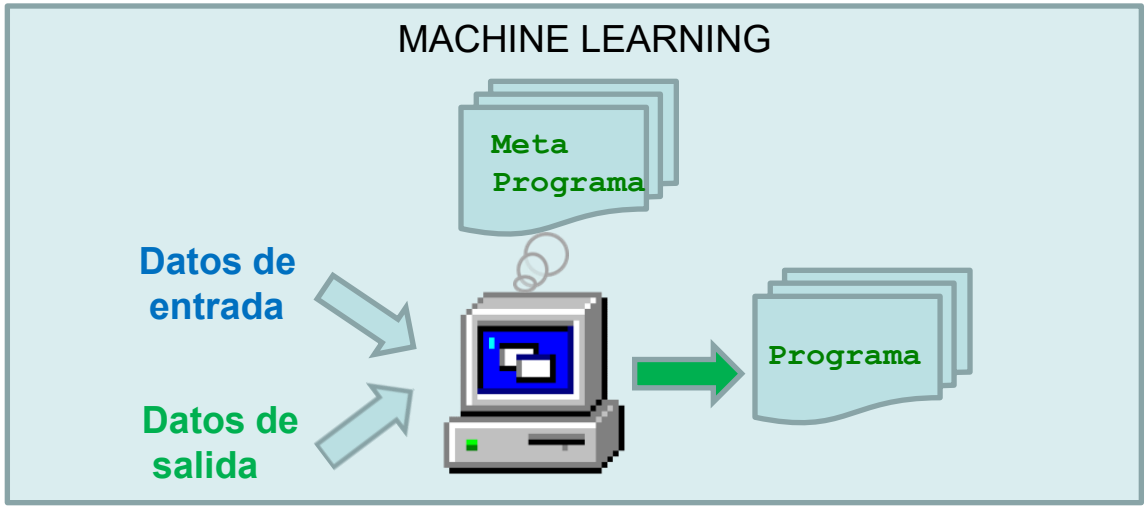


Machine Learning

¿Cuándo se usa ML?

- ✓ Tenemos grandes volúmenes de datos pero no conocemos las reglas a programar.
- ✓ El dominio no es comprensible para el humano.
- ✓ Existen patrones y/o reglas imperceptibles para el humano.

Ejemplo: «yo puedo reconocer tu voz» pero no puedo explicar como lo hago.



Machine Learning

ML se trata de **generalizar comportamientos** a partir de una información suministrada en forma de **ejemplos.**

Es, por lo tanto, un proceso de **inducción del conocimiento.**

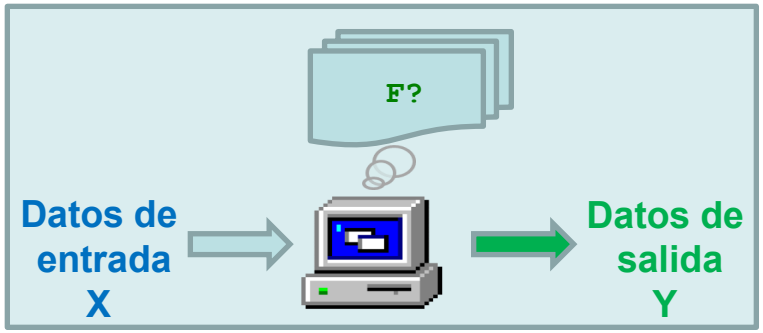
ML es:

Dado:

Training set {(xi, yi) | i = 1 … N}

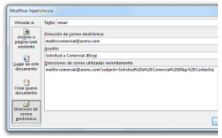
Encontrar:

una buena aproximación a F: X -> Y



Ejemplos: que son *X* e *Y* ?

Detección de Spam



Emails

{Spam, Not Spam}

Reconocimiento de dígitos

Pixels

{0,1,2,3,4,5,6,7,8,9}

Machine Learning

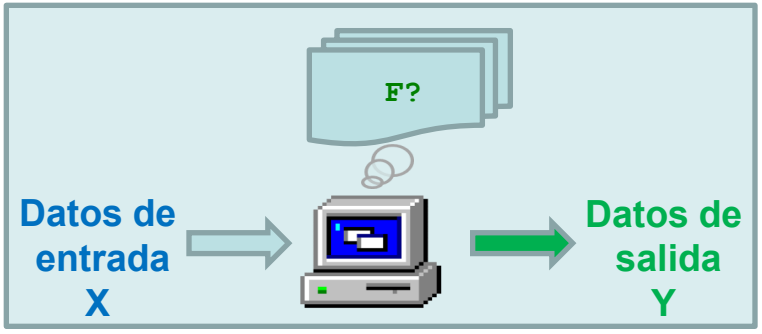
Veamos cómo es Machine Learning por dentro
y qué son los «algoritmos de ML»

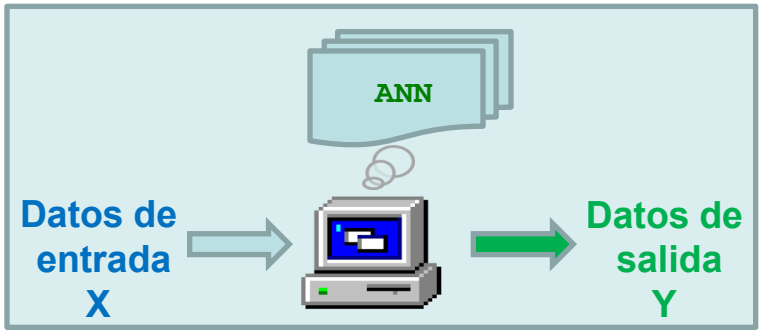
Machine Learning

Existen diferentes técnicas para representar y construir la función.

Por ejemplo:

- **Redes Neuronales Artificiales (ANN)**
- Support Vector Machine (SVM)
- Árboles de Decisión

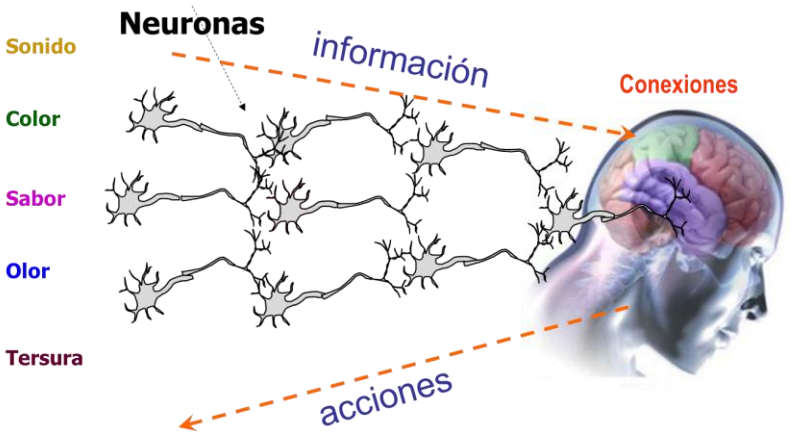


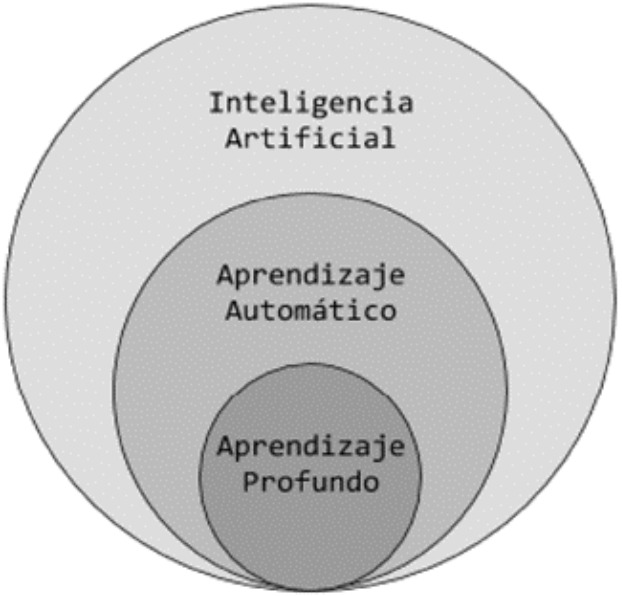


Machine Learning

.

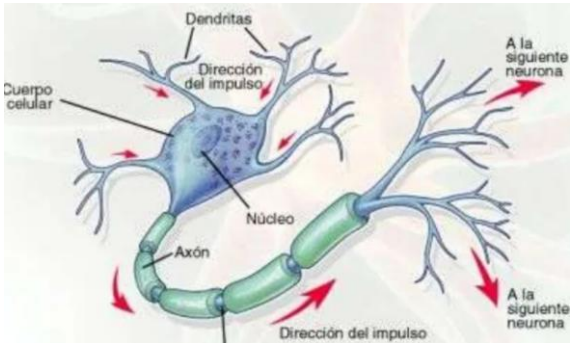
La más prometedora son las **REDES NEURONALES ARTIFICIALES**



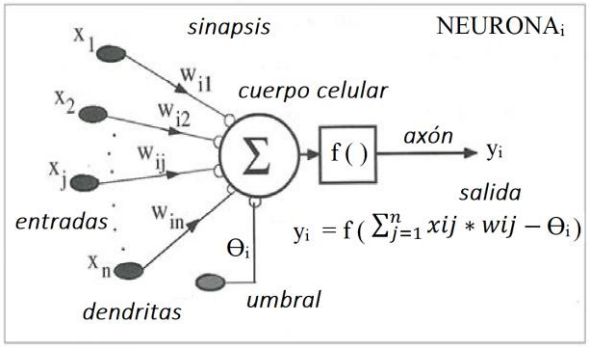


Redes Neuronales Artificiales

Las **redes neuronales** son un modelo computacional basado en un gran conjunto de unidades neuronales simples (neuronas artificiales), de forma análoga a las neuronas en los cerebros biológicos.

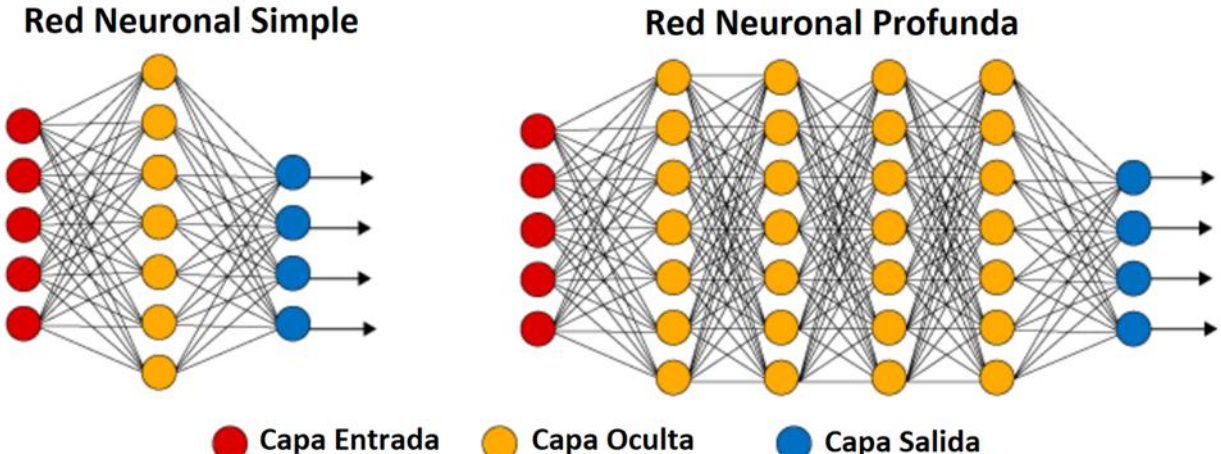


Neurona biológica



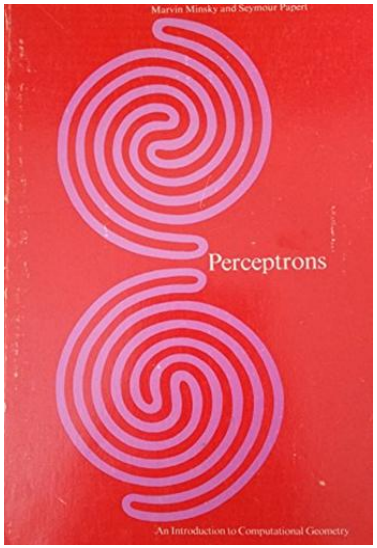
Neurona artificial

- Cada unidad neuronal está conectada con muchas otras y los enlaces entre ellas pueden incrementar o inhibir el estado de activación de las neuronas adyacentes.



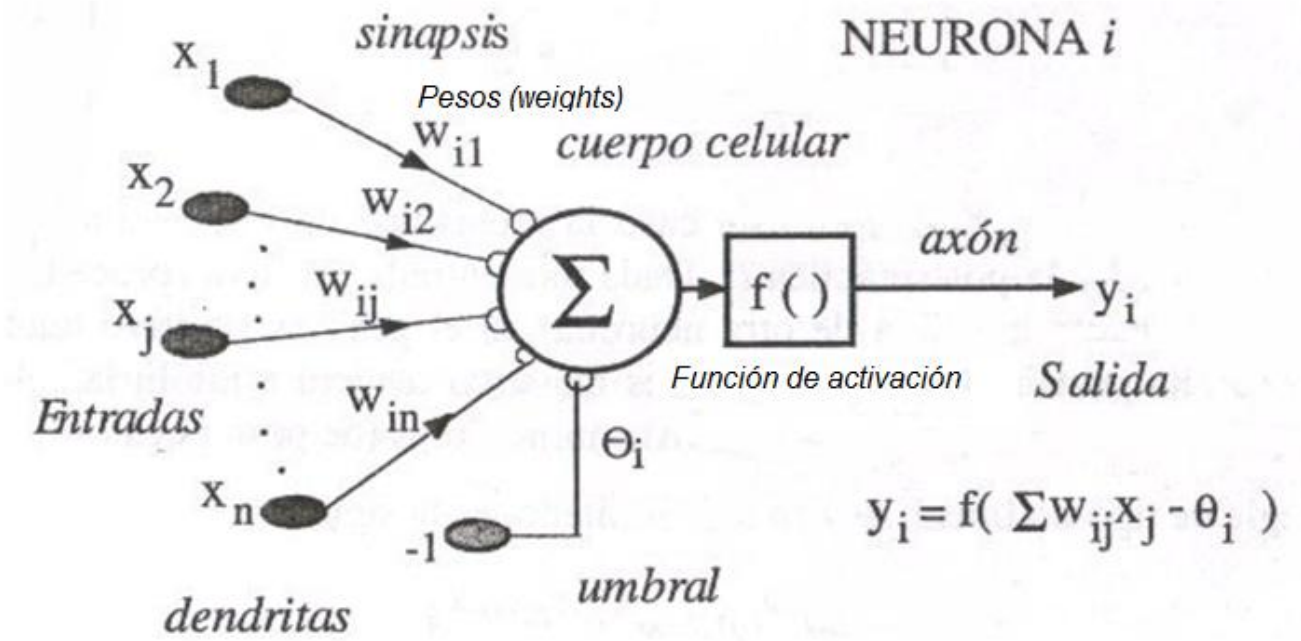
Redes Neuronales Artificiales

- Nada nuevo!!! Se basan en ideas de 1958.



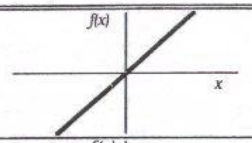
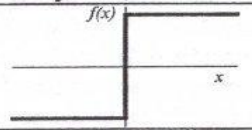
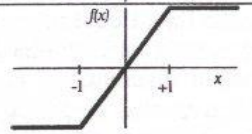
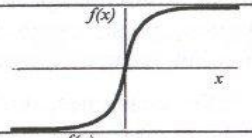
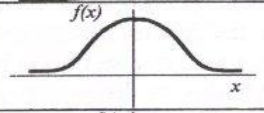
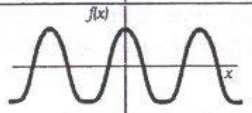
- Perceptrons: an introduction to computational geometry by Marvin Minsky and Seymour Papert (1969)

Redes Neuronales. La neurona



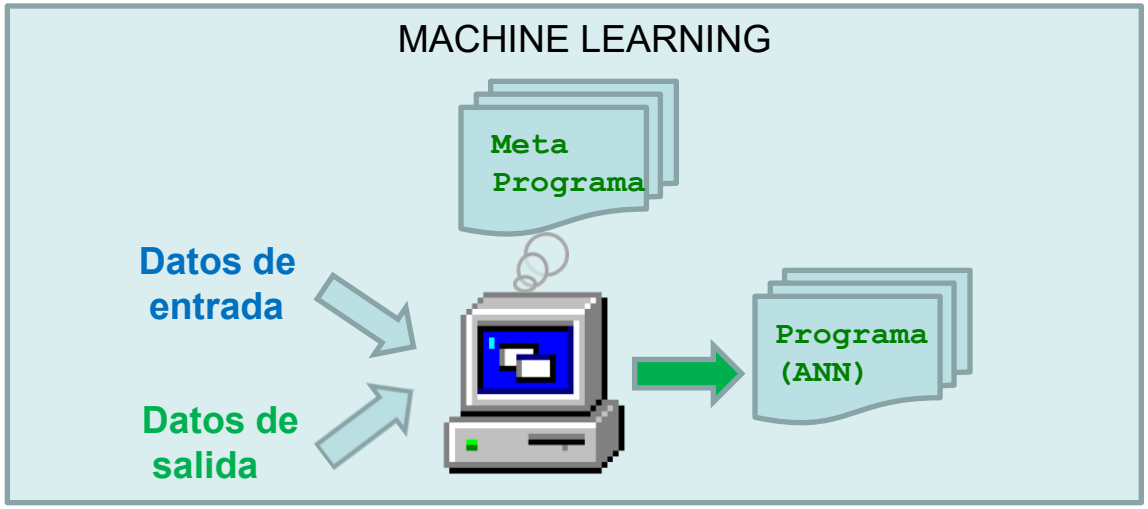
- Las entradas tienen pesos (weights).
- Cada unidad neuronal, de forma individual, opera empleando funciones de suma.
- Existe una función limitadora o umbral, de tal modo que la señal debe sobrepasar un límite antes de propagarse a otra neurona.
- También aplica una función de activación f().

Redes Neuronales. Función de activación

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Machine Learning

Con Machine Learning la máquina construye (aprende) su propio algoritmo para resolver un problema.



Y como hace la máquina para aprender el algoritmo?
O sea, cómo construye la ANN?

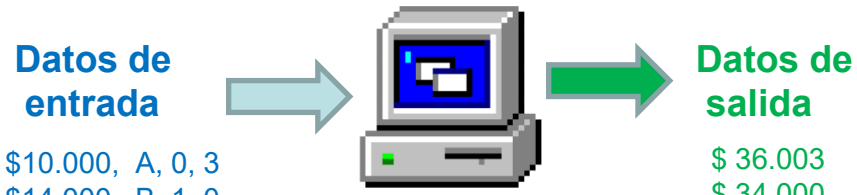
Lo vemos en acción?

Ejemplo. Sin ML

En la programación tradicional:

- El programador escribe el algoritmo que transforma las entradas en las salidas.
- La máquina ejecuta «ciegamente» dicho algoritmo.

```
##  ALGORITMO EN PYTHON QUE CALCULA SUELDOS
def sueldo (basico, categoria, ausencias, cantidad_hijos):
  if categoria == 'A': monto_por_categoria = 10000
  elif categoria == 'B': monto_por_categoria = 20000
  elif categoria == 'C': monto_por_categoria = 30000
  if ausencias == 0: presentismo = 7000
  else: presentismo= 0
  salario_familiar= cantidad_hijos*3001
  sueldo= basico+ monto_por_categoria + presentismo + salario_familiar
  return sueldo
```



Datos de entrada

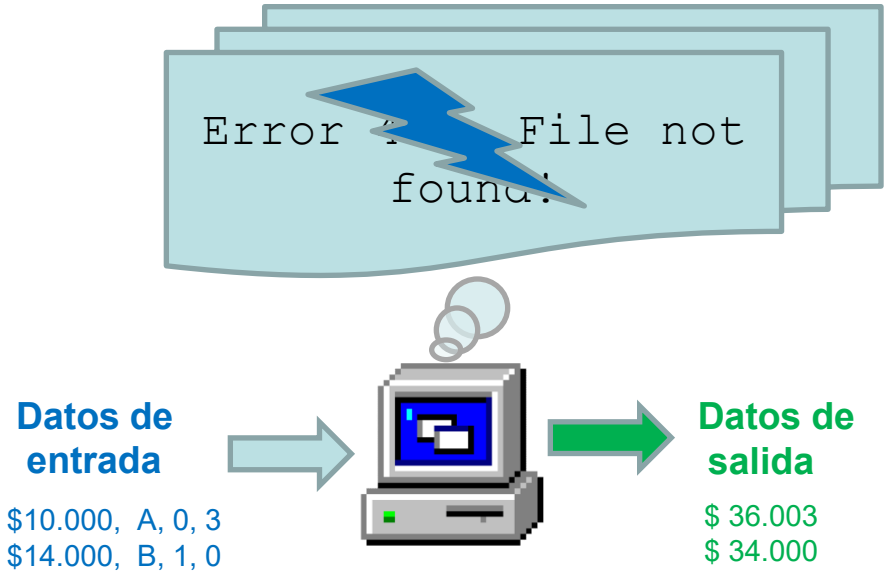
$10.000, A, 0, 3
$14.000, B, 1, 0

Datos de salida

$ 36.003
$ 34.000

Programación tradicional vs. Machine learning.

- ✓Imaginemos que mi algoritmo que calculaba los sueldos se borró.
- ✓¿Cómo hará mi máquina para calcular los sueldos este mes?
- ✓ No conozco el algoritmo. Pero conozco los datos.
- ✓Y mi maquina será capaz de aprender (construir) el algoritmo a partir de los datos!



Ejemplo. Datos para entrenar

datos_empleados_50000.csv

1 to 25 of 50000 entries Filter

	sueldo_basico	categoria	ausencias	cantidad_hijos	sueldo
0	12200	C	0	3	17020
1	26100	B	1	1	28850
2	41700	C	2	2	45700
3	38400	B	2	1	41150
4	71800	C	0	3	76620
5	60600	A	0	0	61970
6	73700	A	1	0	75000
7	88600	B	2	2	92100
8	36000	B	0	3	40320
9	41000	C	0	1	44320
10	70900	C	0	1	74220
11	65800	B	2	2	69300
12	79700	A	2	0	81000
13	33400	B	1	0	35400
14	75800	A	2	3	79350
15	12100	A	2	3	15650
16	14500	C	1	0	17000
17	39600	A	2	0	40900
18	41900	B	0	1	44720
19	57500	A	1	2	60300
20	45800	C	2	3	50550

Ejemplo. Separando los Datos

```python
import numpy as np
from sklearn.model_selection import train_test_split

data = pd.read_csv(nombre_archivo)

#separo los datos de entrada X de los datos de salida Y
X = data.drop(['sueldo'], axis=1)
X = np.array(X.drop(data.columns[[0]], axis=1))
y = np.array(data['sueldo'])

#Separo los datos en training y testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2)
```

X_train

X_test

y_train

y_test

datos_empleados_50000.csv

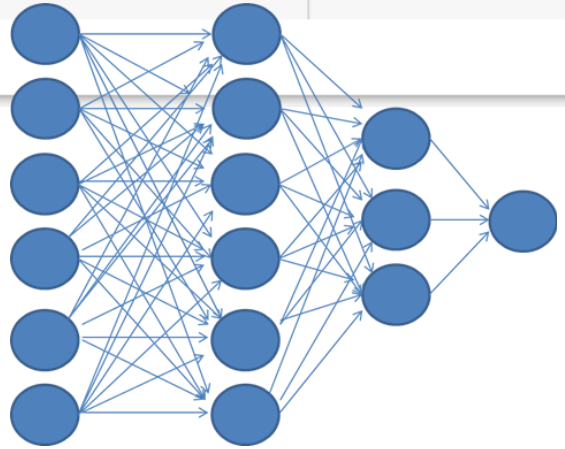
1 to 25 of 50000 entries Filter

	sueldo_basico	categoria	ausencias	cantidad_hijos	sueldo
0	12200	C	0	3	17020
1	26100	B	1	1	28850
2	41700	C	2	2	45700
3	38400	B	2	1	41150
4	71800	C	0	3	76620
5	60600	A	0	0	61970
6	73700	A	1	0	75000
7	88600	B	2	2	92100
8	36000	B	0	3	40320
9	41000	C	0	1	44320
10	70900	C	0	1	74220
11	65800	B	2	2	69300
12	79700	A	2	0	81000
13	33400	B	1	0	35400
14	75800	A	2	3	79350
15	12100	A	2	3	15650
16	14500	C	1	0	17000
17	39600	A	2	0	40900
18	41900	B	0	1	44720
19	57500	A	1	2	60300
20	45800	C	2	3	50550

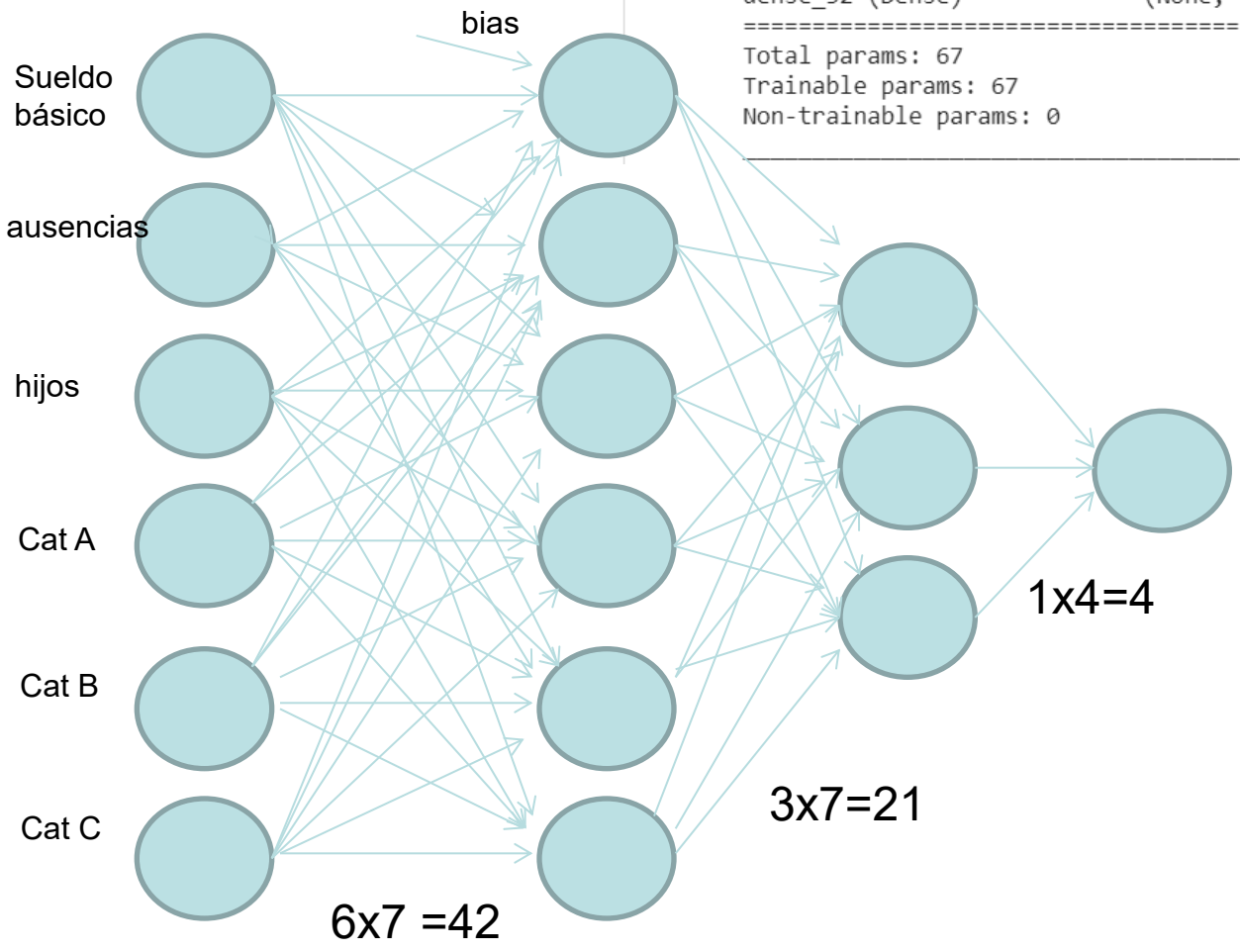
Ejemplo. Creando y entrenando una ANN

```
## ESTIMO CON UNA RED NEURONAL Secuencial Densa (chica)
##
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense

# creo el modelo
model = Sequential()

#tiene 6 datos de entrada por el sueldo basico, antiguedad, hijos, A, B y C
model.add(Dense(6, input_dim=6,input_shape=(6,), activation='relu', kernel_initializer='uniform'))
model.add(Dense(3, activation='relu'))
model.add(Dense(1, activation='relu'))

## compilar
model.compile(loss='mean_squared_error', optimizer='adam',                    )

#se entrena el modelo con los datos X_train Y_train
model.fit(X_train, y_train, verbose=0, batch_size = 10, epochs=10)
```

```
<tensorflow.python.keras.callbacks.History at 0x7f61e12551d0>
```



Ejemplo. Estructura de la ANN

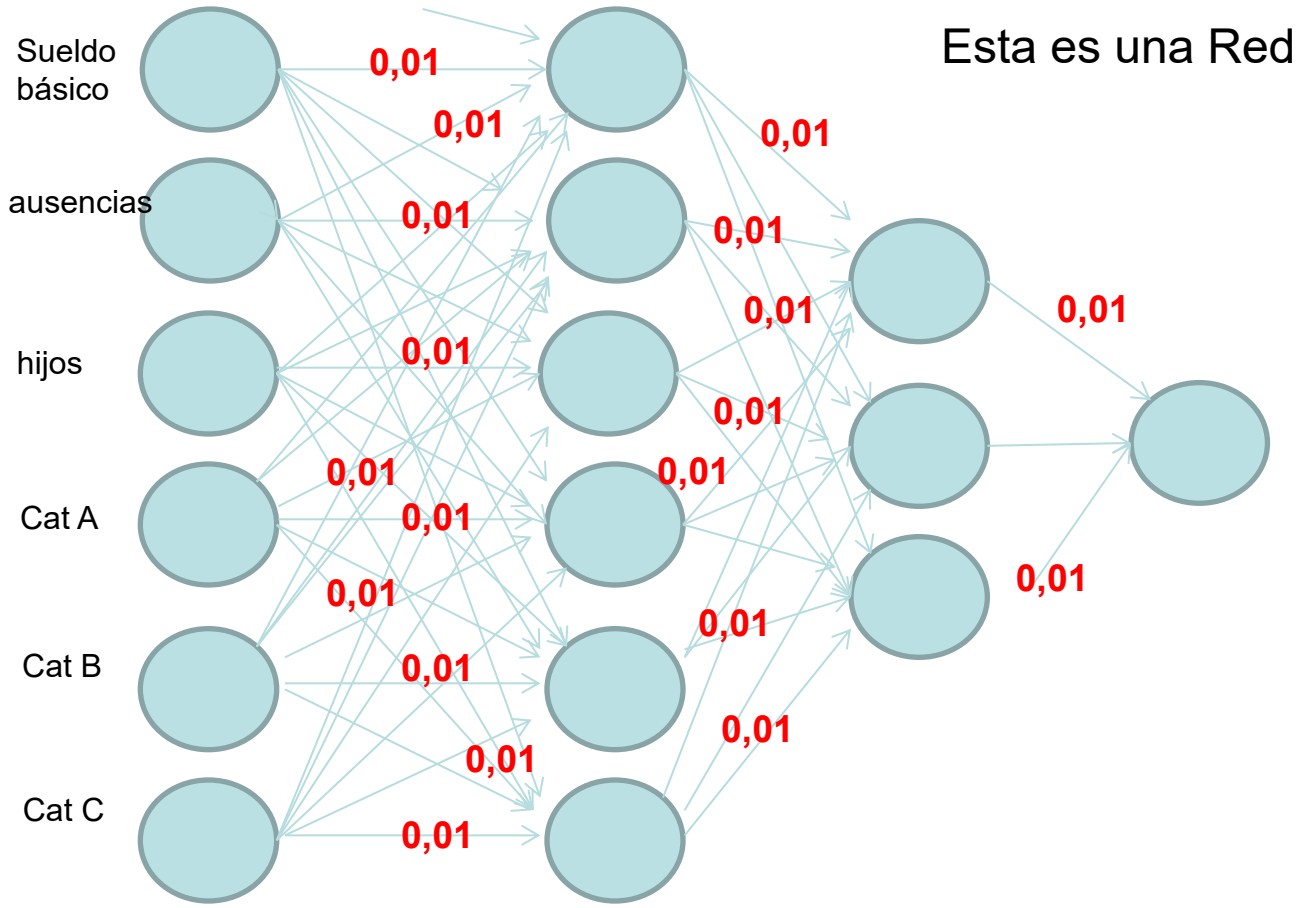


Layer (type)	Output Shape	Param #
dense_30 (Dense)	(None, 6)	42
dense_31 (Dense)	(None, 3)	21
dense_32 (Dense)	(None, 1)	4

Total params: 67
Trainable params: 67
Non-trainable params: 0

Ejemplo. Inicialización de la ANN

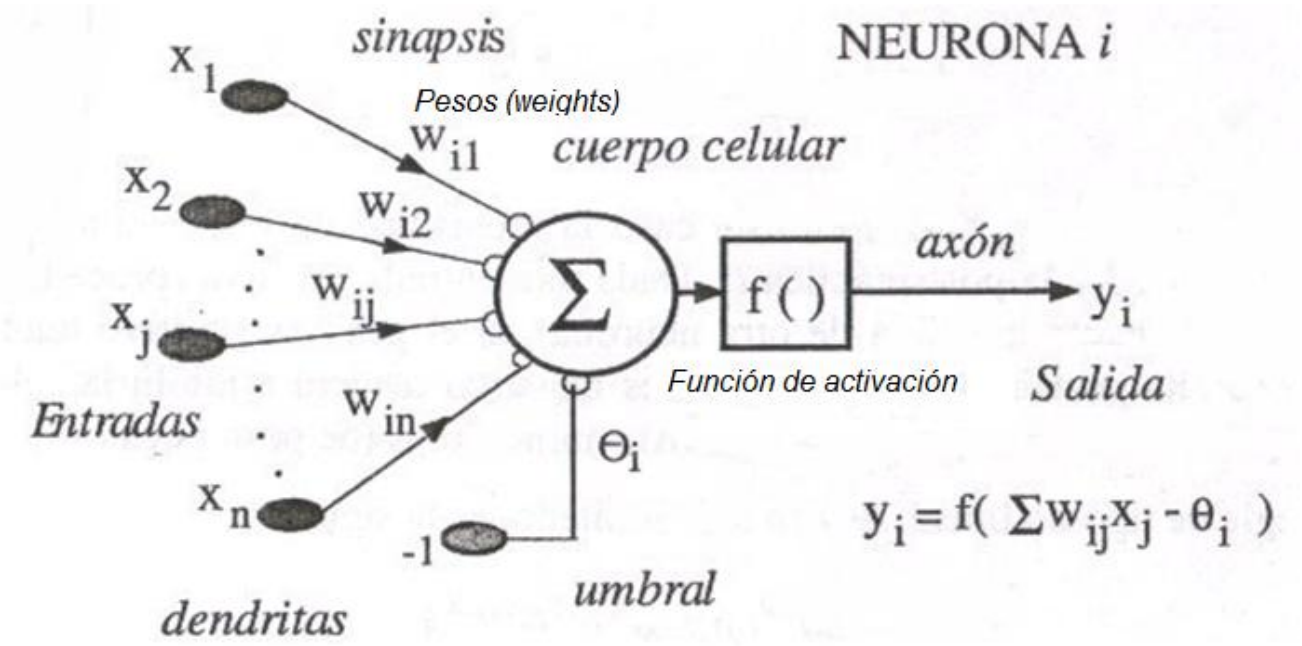
Las redes neuronales artificiales se crean e inicializan con valores aleatorios y luego se someten a un proceso de entrenamiento.



Esta es una Red “recién nacida”

Ejemplo. Comportamiento de la ANN

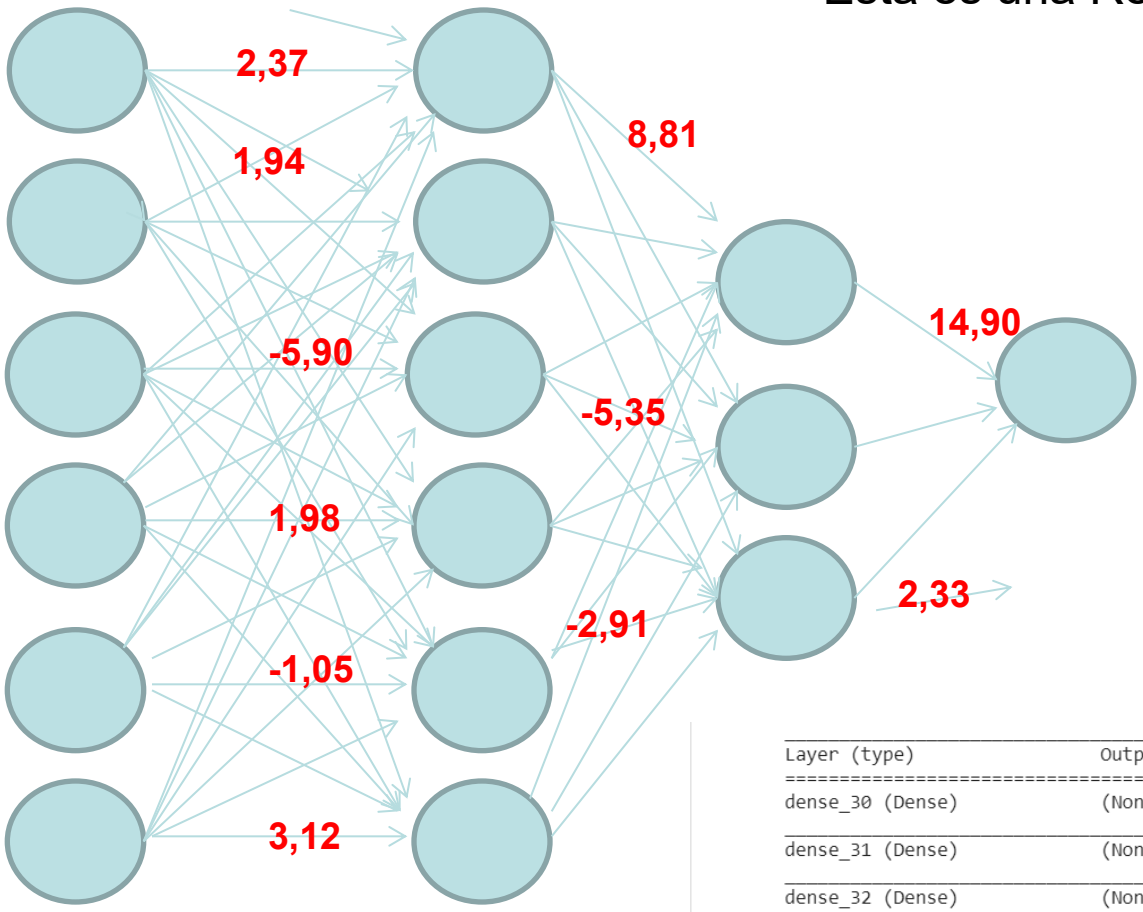
Recordemos como funciona la ANN:



Ejemplo. Aprendizaje de la ANN

Luego del entrenamiento la red “aprendió” los pesos de sus conexiones (parámetros de la red).

Esta es una Red “entrenada”



Layer (type)	Output Shape	Param #
dense_30 (Dense)	(None, 6)	42
dense_31 (Dense)	(None, 3)	21
dense_32 (Dense)	(None, 1)	4

Total params: 67
Trainable params: 67
Non-trainable params: 0

Ejemplo. Parámetros de la ANN

Estos s lo que la ANN «aprendió» a partir de los ejemplos de entrenamiento.

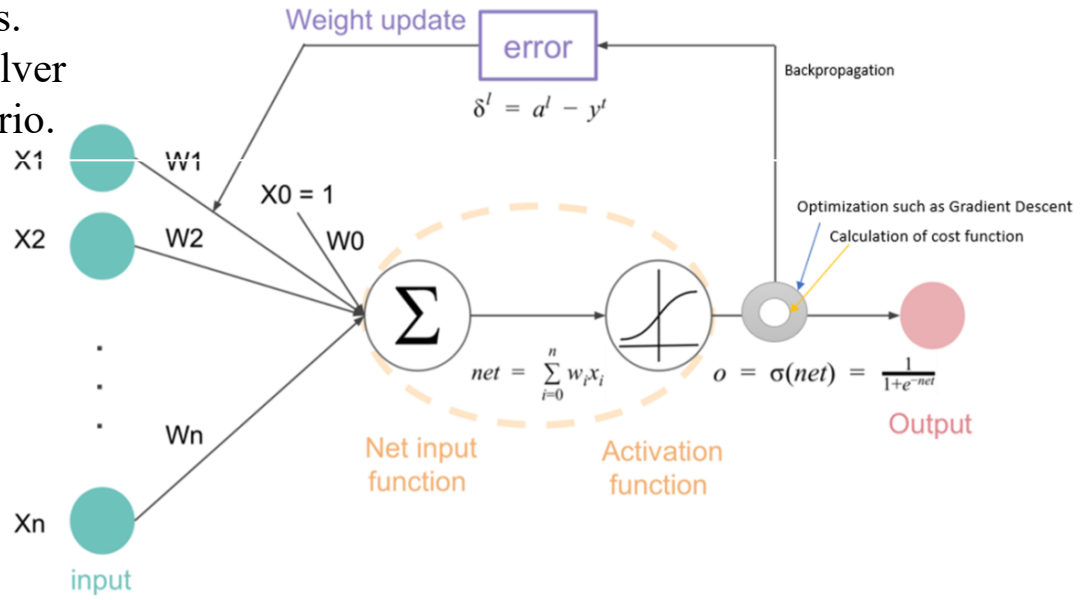
```
=================================================================
Total params: 67
Trainable params: 67
Non-trainable params: 0
_________________________________________________________________
None
[array([[ 2.3753208e-01,  1.9455378e-01, -5.9044957e-03, -1.9834493e-03,
         -1.0555386e-02, -3.1221554e-02],
        [-2.8605638e+00, -2.8482444e+00,  6.5158382e-03, -8.7555784e-01,
          -2.5664544e-02,  1.5504365e-02],
        [-1.4048452e+00, -1.5574017e+00, -1.4083646e-02,  1.8405057e-01,
          -7.9109110e-02,  7.6739595e-04],
        [ 6.0894132e-01,  4.3434203e-01, -3.4527075e-02, -9.0832837e-02,
          -7.3695451e-02, -3.2011069e-02],
        [ 8.9847927e+00,  9.0873766e+00,  2.0794943e-04,  1.0417400e+01,
          -3.3930805e-02, -4.3792557e-02],
        [ 1.5403311e+01,  1.5708423e+01, -2.7559115e-02,  1.7888594e+01,
          -4.4545386e-02, -4.5525804e-03]], dtype=float32), array([15.789694  , 15.987958  ,
        -0.01948806], dtype=float32), array([[ 8.8130021e-01, -5.3563658e-02, -2.9117650e-01
        [ 1.2855804e+00,  1.4941083e-01, -6.5364009e-01],
        [ 2.2938561e-01, -3.2682377e-01, -2.2147667e-01],
        [ 8.8367844e+00,  7.2431450e+00, -4.8270121e-01],
        [-2.6399300e-01,  2.2706485e-01,  6.1077350e-01],
        [-5.6976724e-01,  3.6380875e-01,  3.4164190e-03]], dtype=float32), array([15.376709
        [5.8587036 ],
        [0.91822344]], dtype=float32), array([14.906073], dtype=float32)]
```

Algoritmo de Aprendizaje back-propagation

1. **Inicializar** los pesos y los umbrales iniciales de cada neurona. Hay varias posibilidades de inicialización siendo las mas comunes las que introducen valores aleatorios pequeños.

2. Para cada patrón del conjunto de los datos de entrenamiento:
   a) Obtener la predicción de la red ante ese patrón. Esta parte se consigue propagando la entrada hacia adelante (red **feedforward**)..
   b) Calcular la **Función de error**: comprobando qué tan lejos está la predicción del valor verdadero conocido.
   c) Propagación hacia atrás con descenso de gradiente: calcula derivadas parciales de la función de error para encontrar el conjunto de pesos que minimizan la función de error.
   d) Actualizar pesos y umbrales.
   e) Calcular el error actual y volver al paso 2 si no es satisfactorio.

.



Algoritmo de Aprendizaje back-propagation

El aprendizaje que se suele usar en este tipo de redes recibe el nombre de retropropagacion del error (backpropagation). Como funcion de coste global, se usa el error cuadrático medio. Es decir, que dado un par ($\mathbf{x}_k$, $\mathbf{d}_k$) correspondiente a la entrada k de los datos de entrenamiento y salida deseada asociada se calcula la cantidad:

$$E(w_{ij}, \theta_j, w'_{kj}, \theta'_k) = \frac{1}{2} \sum_p \sum_k \left[d_k^p - f\left(\sum_j w'_{kj} y_j^p - \theta'_k \right) \right]^2$$

Es la suma de los errores parciales debido a cada patrón (índice p), resultantes de la diferencia entre la salida deseada d_p y la salida que da la red f()

Sobre esta función de coste global se aplica algún procedimiento de minimización (ej. descenso por gradiente).

$$\delta w'_{kj} = -\epsilon \frac{\partial E}{\partial w'_{kj}}$$

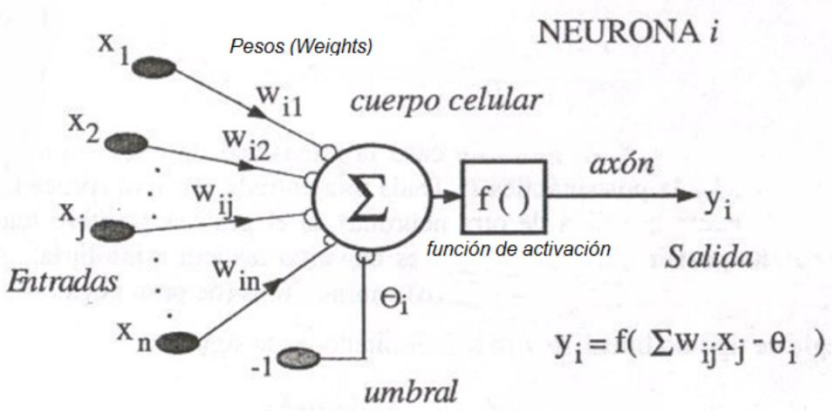
$$\delta w'_{ji} = -\epsilon \frac{\partial E}{\partial w_{ji}}$$

$$\delta w'_{kj} = \epsilon \sum_p \Delta'^p{}_k y_j^p \qquad con \qquad \Delta'^p_k = [d_k^p - f(v'^p{}_k)] \frac{\partial f(v'^p{}_k)}{\partial v'^p{}_k}$$

$$\delta w_{ij} = \epsilon \sum_p \Delta_j^p x_i^p \qquad con \qquad \Delta_j^p = \left(\sum_k \Delta'^p{}_k w'_{kj} \right) \frac{\partial f(v_j^p)}{\partial v_j^p}$$

Parámetros vs. hiperparámetros

- **Los parámetros del modelo** son los pesos y los umbrales. El objetivo del entrenamiento es aprender los valores de estos parámetros.
- **Los hiperparámetros** son parámetros externos establecidos por el operador de la red neuronal.



Hiperparámetros relacionados con la estructura de la red neuronal:

- Cantidad de capas ocultas ej. 1
- Función de activación ej relu
- Inicialización de pesos ej: valores chicos

Hiperparámetros relacionados con el entrenamiento:

- Épocas,
- Cantidad de iteraciones
- tamaño de lote.
- Algoritmo optimizador (ej. sgd descenso del gradiente, adam)
- Función de costo (o loss) ej. mse mean squared error

Recordemos el código

```
## ESTIMO CON UNA RED NEURONAL Secuencial Densa (chica)
##
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense

# creo el modelo
model = Sequential()

#tiene 6 datos de entrada por el sueldo basico, antiguedad, hijos, A, B y C
model.add(Dense(6, input_dim=6,input_shape=(6,), activation='relu', kernel_initializer='uniform'))
model.add(Dense(3, activation='relu'))
model.add(Dense(1, activation='relu'))

## compilar
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['binary_accuracy'])

#se entrena el modelo con los datos X_train Y_train
model.fit(X_train, y_train, verbose=0, batch_size = 10, epochs=10)
```

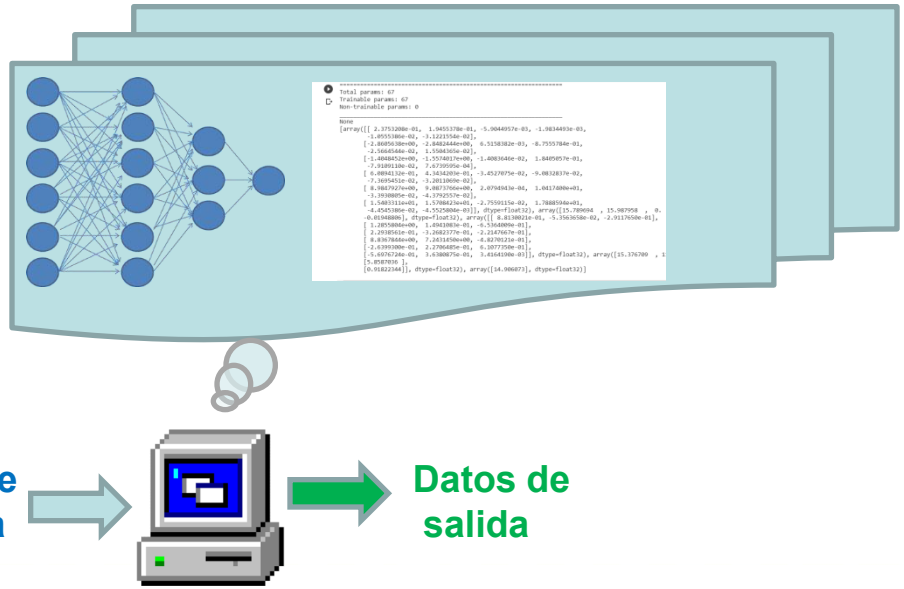
```
<tensorflow.python.keras.callbacks.History at 0x7f61e12551d0>
```

Métodos de ajuste de hiperparámetros

- **Ajuste manual**: un operador experimentado puede adivinar valores. Esto requiere prueba y error.
- **Búsqueda de cuadrícula:** esto implica probar sistemáticamente múltiples valores de cada hiperparámetro y volver a entrenar el modelo para cada combinación.
- **Búsqueda aleatoria:** un estudio de investigación mostró que el uso de valores de hiperparámetros aleatorios es en realidad más efectivo que la búsqueda manual o la búsqueda de cuadrícula.
- **Optimización bayesiana:** métodos estadísticos han mostrado ser mas efectivos en algunos dominios.

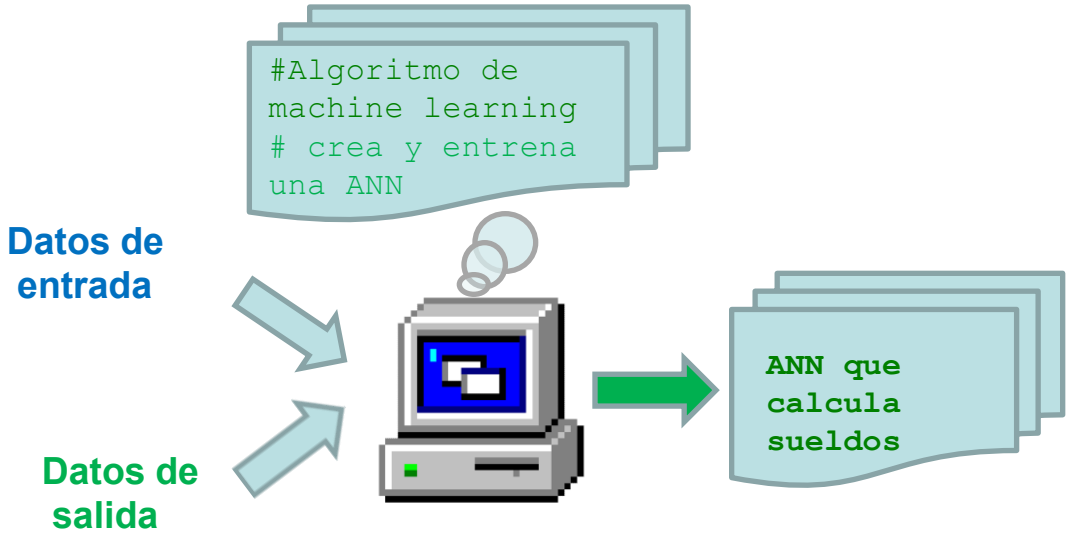
Programación tradicional vs. ML

```
##  ALGORITMO EN PYTHON QUE CALCULA SUELDOS
def sueldo (basico, categoria, ausencias, cantidad_hijos):
  if categoria == 'A': monto_por_categoria = 10000
  elif categoria == 'B': monto_por_categoria = 20000
  elif categoria == 'C': monto_por_categoria = 30000
  if ausencias == 0: presentismo = 7000
  else: presentismo= 0
  salario_familiar= cantidad_hijos*3001
  sueldo= basico+ monto_por_categoria + presentismo + salario_familiar
  return sueldo
```



Programación tradicional vs. ML

✓El programador escribió un meta-algoritmo que genera algoritmos.
Ej: en Keras, el objeto Sequential con sus métodos add(), compile(), fit() y predict()



✓La máquina ejecuta «ciegamente» dicho meta-algoritmo.

Y retomemos la pregunta ¿Qué es la Inteligencia Artificial?

Coloquialmente, el término inteligencia artificial se aplica cuando una máquina **imita las funciones cognitivas que los humanos** consideramos como propias de nuestras mentes, en especial: **comunicarnos**, resolver problemas, tomar decisiones y **aprender**.

¿ Son IA todos los **sistemas computacionales que imitan nuestras funciones cognitivas**?

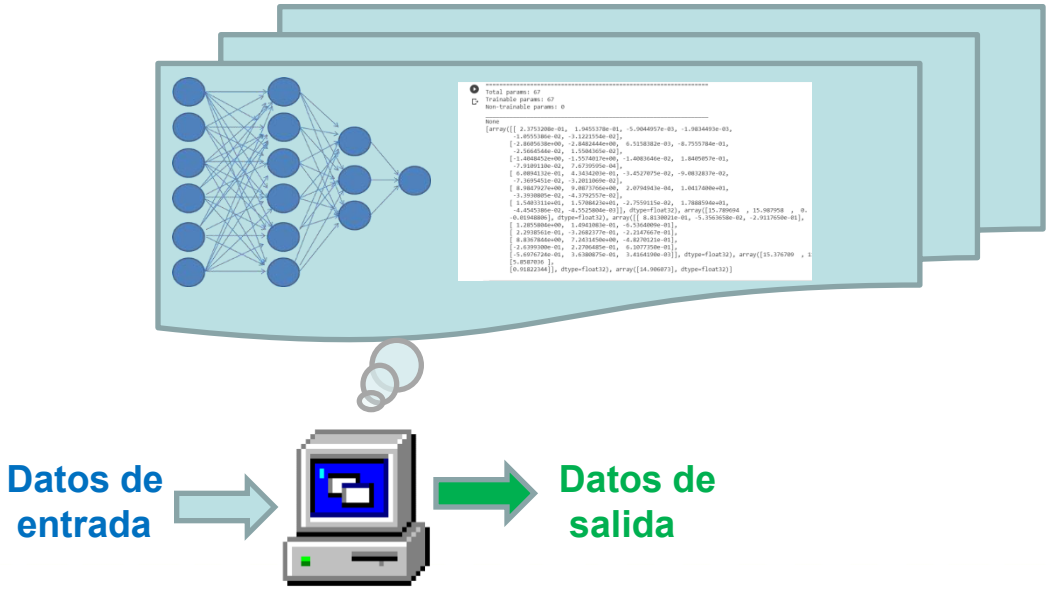
Es IA cuando la máquina no sigue un algoritmo (una receta) para resolver un problema,

sino que **encuentra su propia forma** de resolverlo.

Programación tradicional vs. ML

```python
##  ALGORITMO EN PYTHON QUE CALCULA SUELDOS
def sueldo (basico, categoria, ausencias, cantidad_hijos):
  if categoria == 'A': monto_por_categoria = 10000
  elif categoria == 'B': monto_por_categoria = 20000
  elif categoria == 'C': monto_por_categoria = 30000
  if ausencias == 0: presentismo = 7000
  else: presentismo= 0
  salario_familiar= cantidad_hijos*3001
  sueldo= basico+ monto_por_categoria + presentismo + salario_familiar
  return sueldo
```

Comparemos!!!



En resumen…

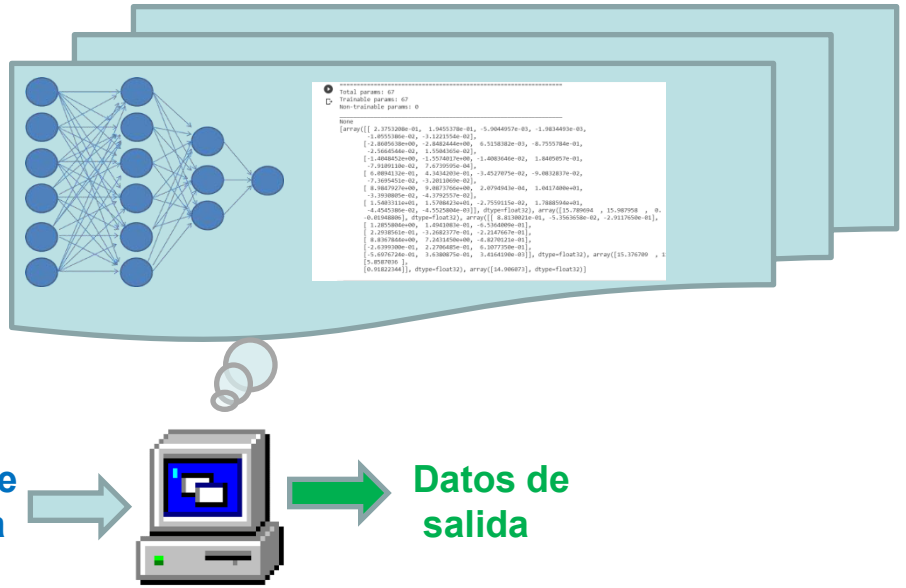
- Las máquinas nos sorprenden resolviendo problemas para los cuales nadie les escribió el algoritmo. Ellas por su cuenta son capaces de aprender el camino.
- Pero **no son conscientes** de lo que están haciendo, continúan siguiendo al pie de la letra otros algoritmos que los humanos hemos programado, los algoritmos de aprendizaje.
- Sin ver el código no se puede distinguir un sistema de IA de un sistema programado tradicionalmente.
- Debe legislarse sobre el **software**, no sobre la **Inteligencia Artificial**.

Y la «ingenieria de soft» donde va?

Hay buenas prácticas?

- ✓ Design patterns
- ✓ Reuso
- ✓ Cohesion,
- ✓ Acoplamiento,
- ✓ encapsulamiento
- ✓ Code smells

```
##  ALGORITMO EN PYTHON QUE CALCULA SUELDOS
def sueldo (basico, categoria, ausencias, cantidad_hijos):
  if categoria == 'A': monto_por_categoria = 10000
  elif categoria == 'B': monto_por_categoria = 20000
  elif categoria == 'C': monto_por_categoria = 30000
  if ausencias == 0: presentismo = 7000
  else: presentismo= 0
  salario_familiar= cantidad_hijos*3001
  sueldo= basico+ monto_por_categoria + presentismo + salario_familiar
  return sueldo
```



¿Cómo es el ciclo de vida?