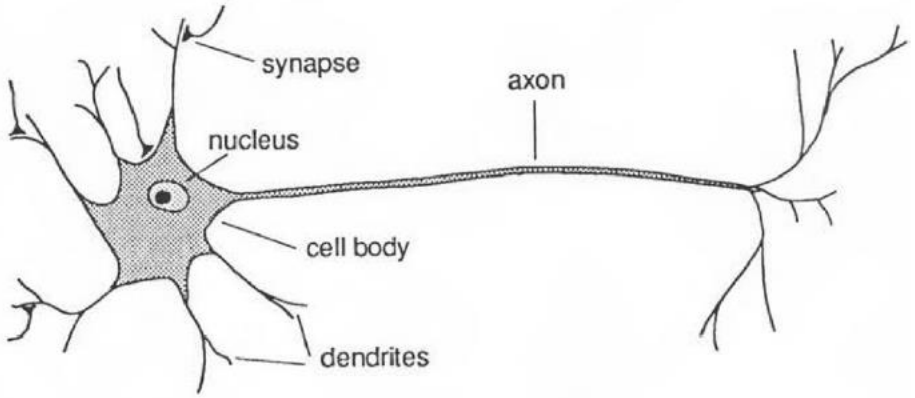
REDES NEURONALES ARTIFICIALES – DEEP LEARNING

REDES NEURONALES FEED FORWARD MULTICAPA

LAURA DIAZ DÁVILA

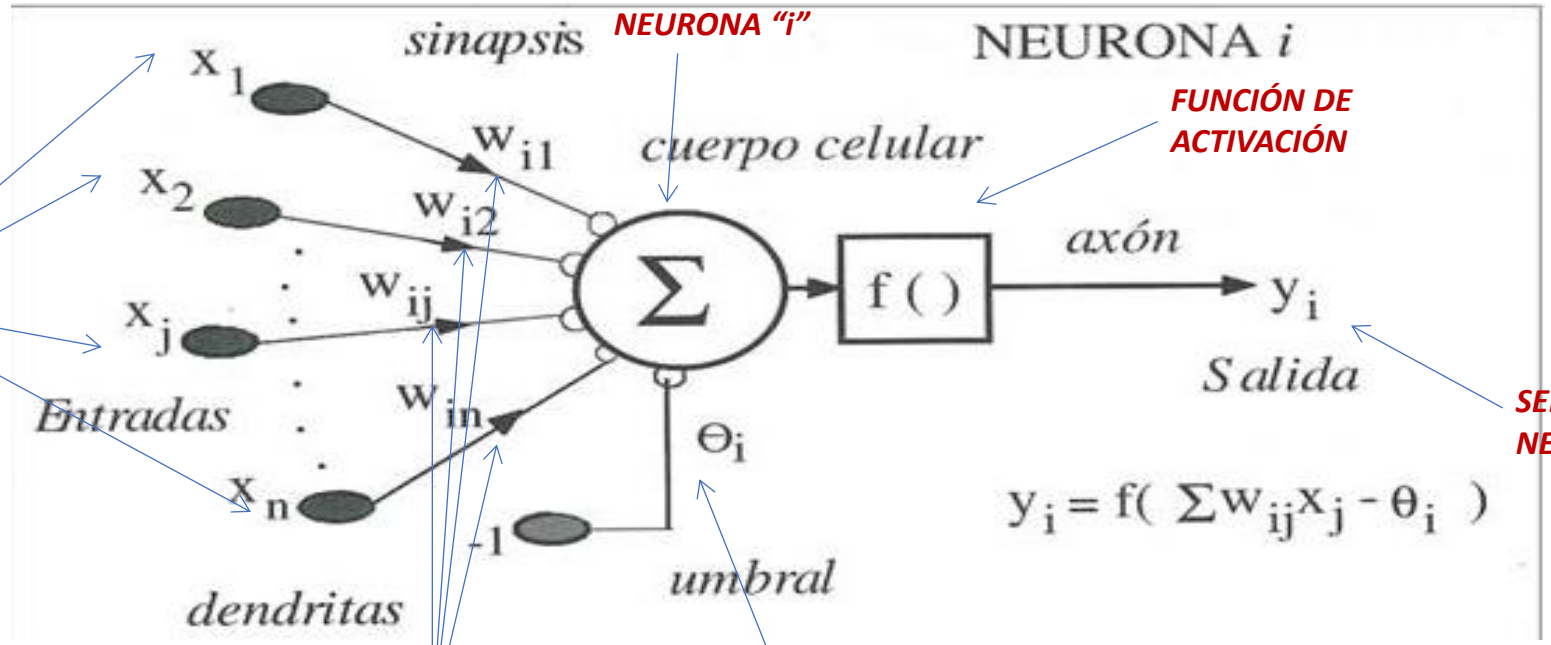
LA NEURONA ARTIFICIAL



Modelo general de neurona artificial

- Dispositivo no lineal.
- Conjunto de entradas.
- Pesos sinápticos.
- Regla de propagación.
- Función de activación.
- Función de salida.

LA NEURONA ARTIFICIAL



POTENCIAL POST SINÁPTICO DE LA NEURONA "i"

FUNCIÓN DE ACTIVACIÓN

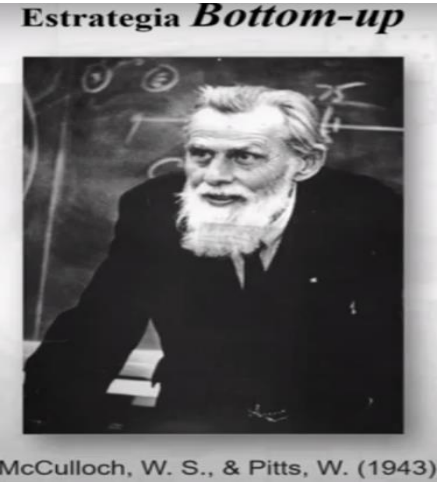
N SEÑALES DE ENTRADA

SEÑAL DE SALIDA DE LA NEURONA "i"

PARÁMETROS O PESOS SINÁPTICOS

BIAS O UMBRAL DE EXCITACIÓN

Redes Neuronales Artificiales



McCulloch, W. S., & Pitts, W. (1943)

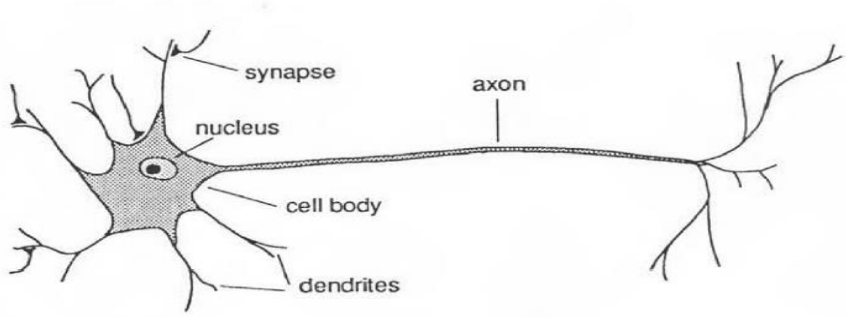
Componentes de un sistema neuronal o conexionista

- Un conjunto de procesadores elementales (neuronas).
- Un patrón de conectividad o arquitectura.
- Una dinámica de activaciones.
- Una regla o dinámica de aprendizaje.
- El entorno donde opera.

Arquitectura de las Redes Neuronales Artificiales

CONEXIONES

- Excitatorias (peso sináptico positivo)
- Inhibitorias.
- Intracapa
- Intercapa.
- Realimentadas.



ESTRUCTURAS EN CAPAS

- Monocapa.
- Multicapa.

SEGÚN EL FLUJO DE DATOS

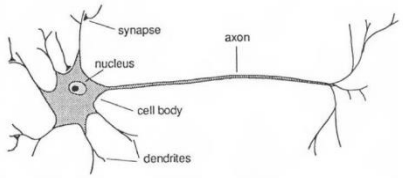
- Unidireccionales (feedforward).
- Recurrentes o realimentadas.

DINÁMICA DE ACTUALIZACIÓN DEL ESTADO DE LAS NEURONAS

- Síncronas.
- Asíncronas.
- No determinista (estocástica).

SEGÚN EL TIPO DE ASOCIACIÓN

- Hetroasociativa.
- Autoasociativa.



Aspectos de las Redes Neuronales Artificiales

TIPOS DE NEURONAS

- Binarias o continuas.
- Tipo McCulloch-Pitts.
- Tipo Ising.
- Tipo Potts.
- Continua.

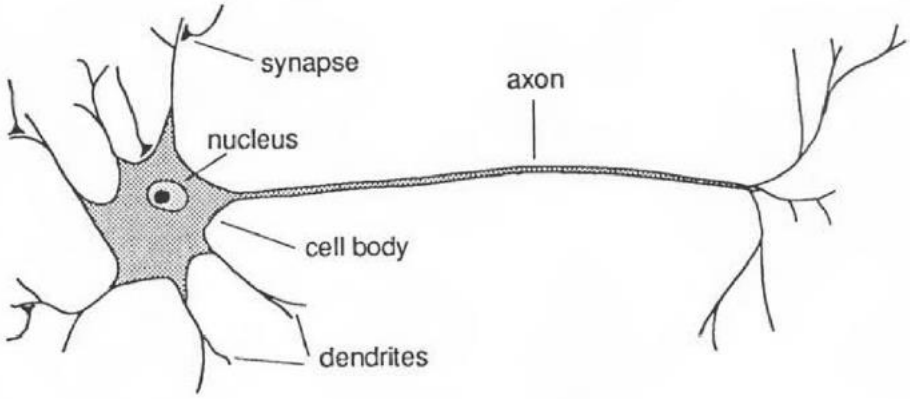
FUNCIONES DE ACTIVACIÓN

- Función identidad (Adalina).
- Función escalón (Perceptrón simple y Hopfield discreta).
- Lineal a tramos.
- Tipo sigmoide (BP, derivable).
- Gaussiana.
- Sinusoidales (periodicidad temporal)

CARACTERIZACIÓN

1. Arquitectura: Número de capas, número de neuronas por capa, grado de conectividad y tipo de conexiones entre neuronas.
2. Mecanismos de aprendizaje y recuerdo o ejecución.
3. Representación de la información de entrada y salida

Arquitectura de las Redes Neuronales Artificiales



Capas de Neuronas Artificiales

1) Aquellas que reciben estímulos externos, relacionadas con el aparato sensorial, que tomarán la información de entrada.

2) Dicha información se transmite a ciertos elementos internos que se ocupan de su procesado. Es en las sinapsis y neuronas correspondientes a *este segundo nivel donde se genera cualquier tipo de representación* interna de la información. Puesto que no tienen relación directa con la información de entrada ni con la de salida, estos elementos se denominan unidades ocultas.

3) Una vez finalizado el período de procesado, la información llega a las unidades de salida, *cuya misión es dar la respuesta del sistema*.

PERCEPTRON MLP - BACKPROPAGATION

REDES NEURONALES APRENDIZAJE SUPERVISADO – UNIDIRECCIONALES -MULTICAPA
(WIDROW 1960, RUMELHART 1986)

EL PERCEPTRON GENERALIZADO

ESPACIO DE SOLUCIONES NO LINEALMENTE SEPARABLES

1. **MLP BACK PROPAGATION**
2. **DEEP LEARNING**

- Rosenblatt (1961)
- Minsky (1969)
- Hopfield (1982)
- Rumelhart (1986)

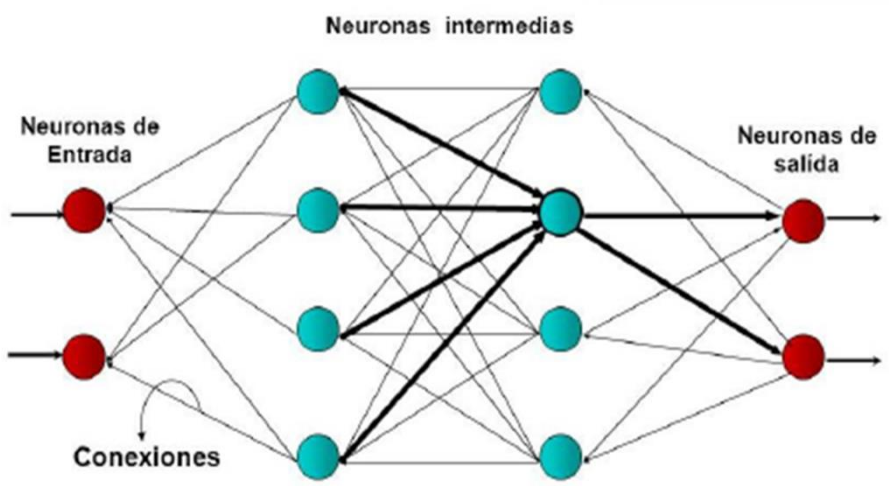
DEFINIENDO a RNA MLP BACK-PROPAGATION

NÚMERO DE CAPAS

TIPOS DE NEURONAS

MODO DE APRENDIZAJE

DINÁMICA DE ACTIVACIÓN

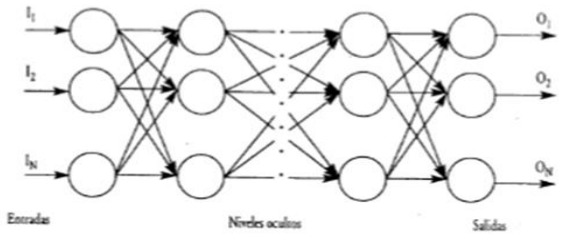


CONECTIVIDAD

PROPAGACIÓN DE LA SEÑAL

DISEÑO EL MODELO

Espacios de los patrones de entrada no linealmente separables



Cantidad de patrones -capacidad de generalización-.

Número de capas ocultas.

Número de neuronas por capa oculta.

Representación de la información de salida.

Función de transferencia.

Parámetros de ajuste de la red.

La red Backpropagation.

La regla delta generalizada.

Aspectos del funcionamiento del algoritmo.

Incorporación de momentum.

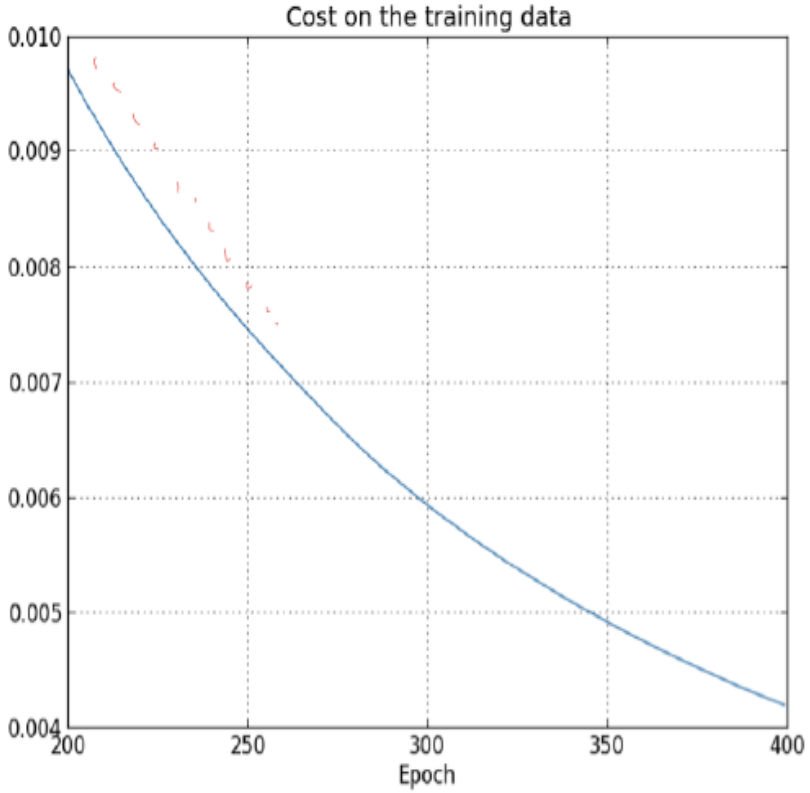
Dimensionamiento de la red.

PERCEPTRON MLP - BACKPROPAGATION

APRENDIZAJE

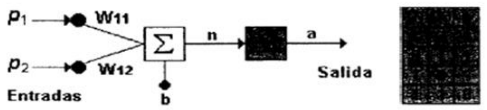
Este algoritmo utiliza una función o superficie de error asociada a la red, buscando el estado estable de mínimo error a través del camino descendente de la superficie de error. Para ello, realimenta el error del sistema para realizar la modificación de los pesos en un valor proporcional al gradiente decreciente de dicha función de error.

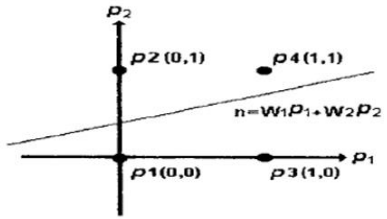
U_j. El error que se produce en una neurona oculta es la suma de todos los errores cometidos por las neuronas a las que está conectada su salida multiplicados por el peso de la conexión correspondiente.



MLP – BACKPROPAGATION: ALGORITMO DE APRENDIZAJE

Puerta lógica XOR





ENTRENAMIENTO "EN LÍNEA"

ENTRENAMIENTO "EN BATCH"

ENTRENAMIENTO "EN MINI BATCHES"

Paso 1: Iniciar los pesos sinápticos -random-

Paso 2: Presentar un patrón -valores de las neuronas de entrada y valores de salida deseados-

Paso 3: Echar a andar la red para calcular los valores de salida para ese patrón -pasando por las capas ocultas-

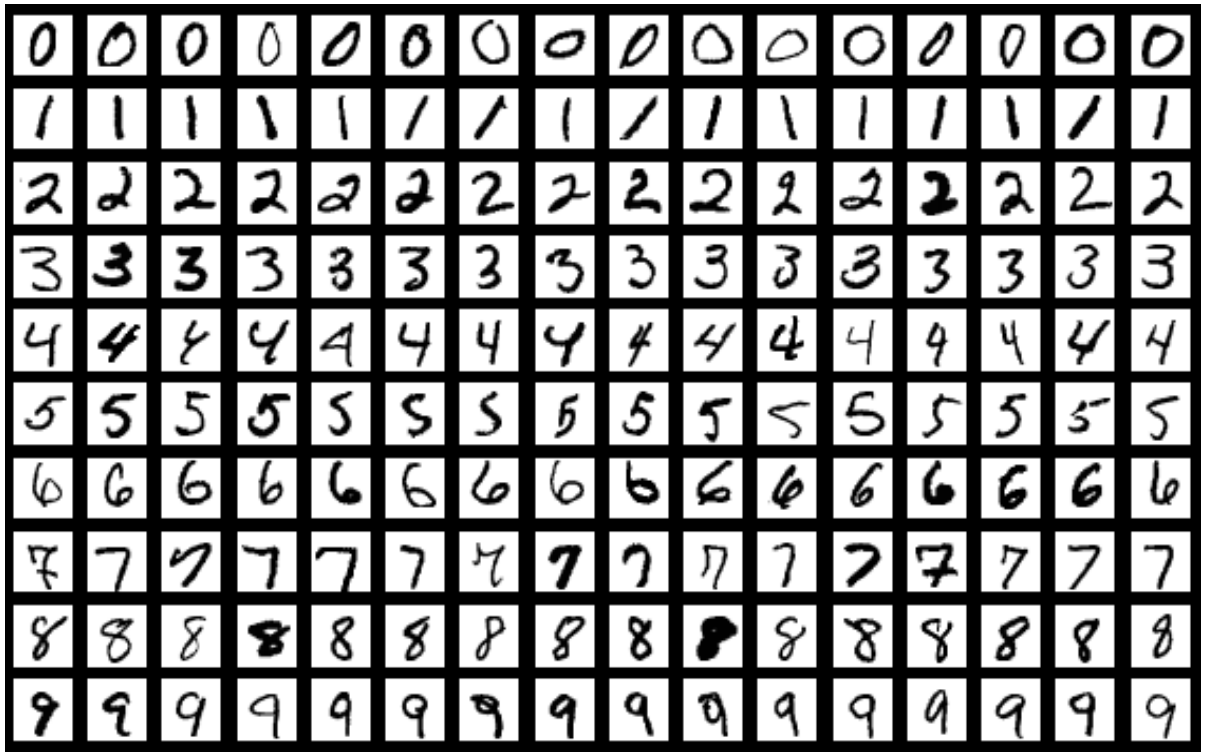
Paso 4: Determinar el error delta, comenzando desde la salida y ajustando hacia atrás pasando por todas las ocultas.

Paso 5: Actualizar los pesos sinápticos desde la capa de salida hasta la última oculta.

Paso 6: Repetir desde P. 3 para todos los patrones y hasta que el error medio resulte admisible.

EJEMPLO: MLP - BACKPROPAGATION

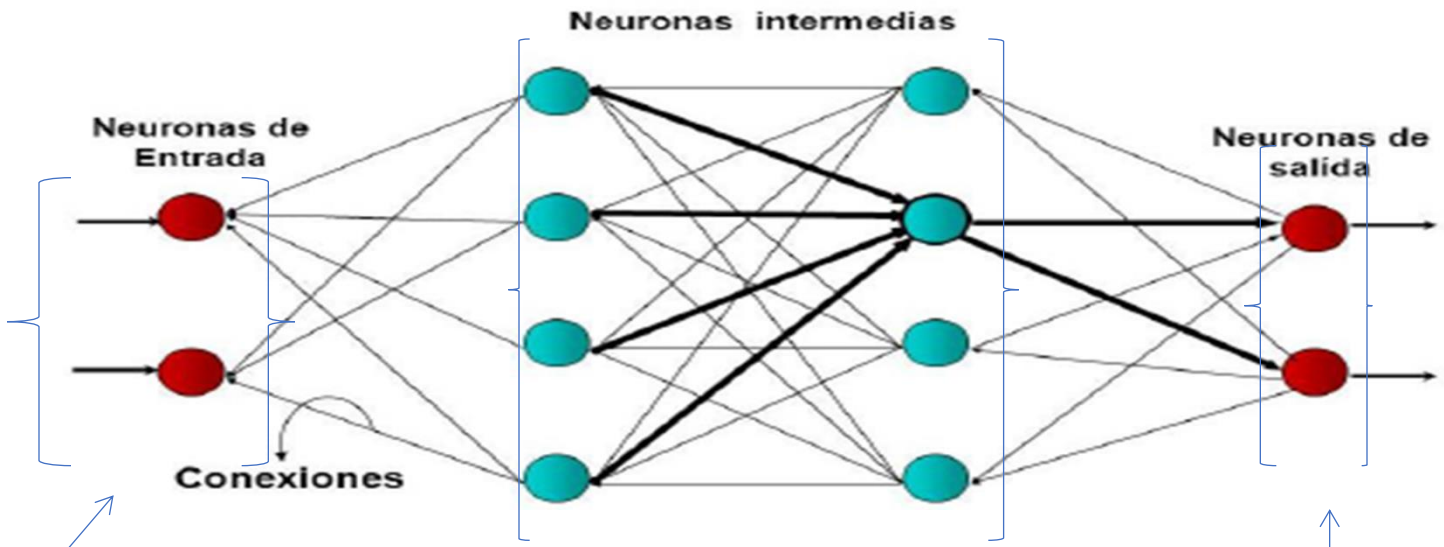
MNIST



PERCEPTRON MLP - BACKPROPAGATION

MNIST

DISEÑO DEL MODELO: ARQUITECTURA



28 * 28 PIXELES:
784 NEURONAS
EN LA CAPA DE
ENTRADA

DOS CAPAS OCULTAS
256 NEURONAS
CADA UNA

CAPA DE SALIDA: 10
NEURONAS
DISCRETAS

<u>Parámetros de entrenamiento</u>

training epochs: 15

batch size: 200

MLP - BACKPROPAGATION

```
#Network parameters

n_hidden1 = 256 # 1st layer number of neurons
n_hidden2 = 256 # 2nd layer number of neurons
n_input = 784 # MNIST data input img shape is 28*28
n_output = 10 # The MNIST dataset has 10 classes, representing the digits 0 through 9.

#Learning parameters

learning_constant = 0.01
number_epochs = 1000
batch_size = 100
```

```
#tf Graph input: assign X and Y

X = tf.placeholder("float", [None, n_input])
Y = tf.placeholder("float", [None, n_output])
```

```
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
Epoch: 0
Epoch: 100
Epoch: 200
Epoch: 300
Epoch: 400
Epoch: 500
Epoch: 600
Epoch: 700
Epoch: 800
Epoch: 900
Accuracy: 0.9187
```

```
#DEFINING WEIGHTS AND BIASES

#Biases first hidden layer
b1 = tf.Variable(tf.random_normal([n_hidden1]))
#Biases second hidden layer
b2 = tf.Variable(tf.random_normal([n_hidden2]))
#Biases output layer
b3 = tf.Variable(tf.random_normal([n_output]))

#Weights connecting input layer with first hidden layer
w1 = tf.Variable(tf.random_normal([n_input, n_hidden1]))
#Weights connecting first hidden layer with second hidden layer
w2 = tf.Variable(tf.random_normal([n_hidden1, n_hidden2]))
#Weights connecting second hidden layer with output layer
w3 = tf.Variable(tf.random_normal([n_hidden2, n_output]))
```

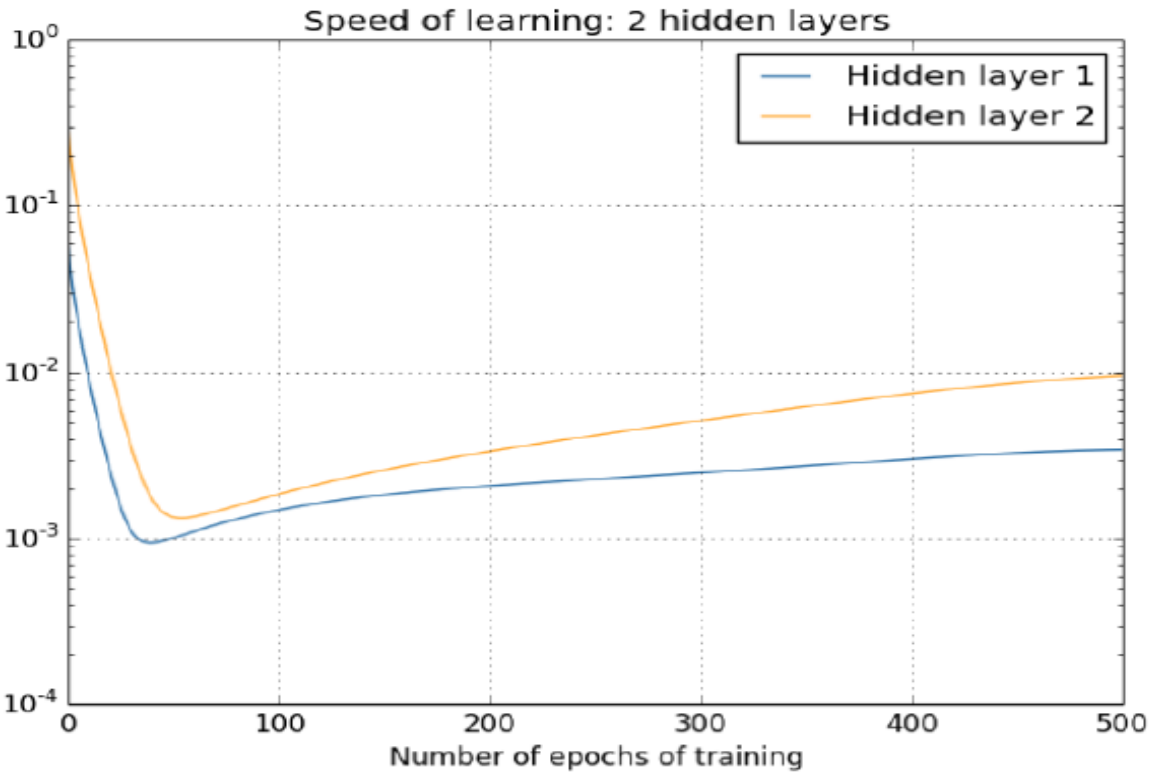
```
#Define loss and optimizer
loss_op = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=neural_network,labels=Y))
#Define how to fix it
optimizer = tf.train.GradientDescentOptimizer(learning_constant).minimize(loss_op)
```

```
with tf.Session() as sess:
    sess.run(init)
    #Training epoch
    for epoch in range(number_epochs):
        #Get one batch of images
        batch_x, batch_y = mnist.train.next_batch(batch_size)
        #Run the optimizer feeding the network with the batch
        sess.run(optimizer, feed_dict={X: batch_x, Y: batch_y})
        #Display the epoch
        if epoch % 100 == 0:
            print("Epoch:", '%d' % (epoch))
    # Test model
    pred = tf.nn.softmax(neural_network)  # Apply softmax to logits
    correct_prediction = tf.equal(tf.argmax(pred, 1), tf.argmax(Y, 1))
    # Calculate accuracy
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    print("Accuracy:", accuracy.eval({X: mnist.test.images, Y: mnist.test.labels}))
```

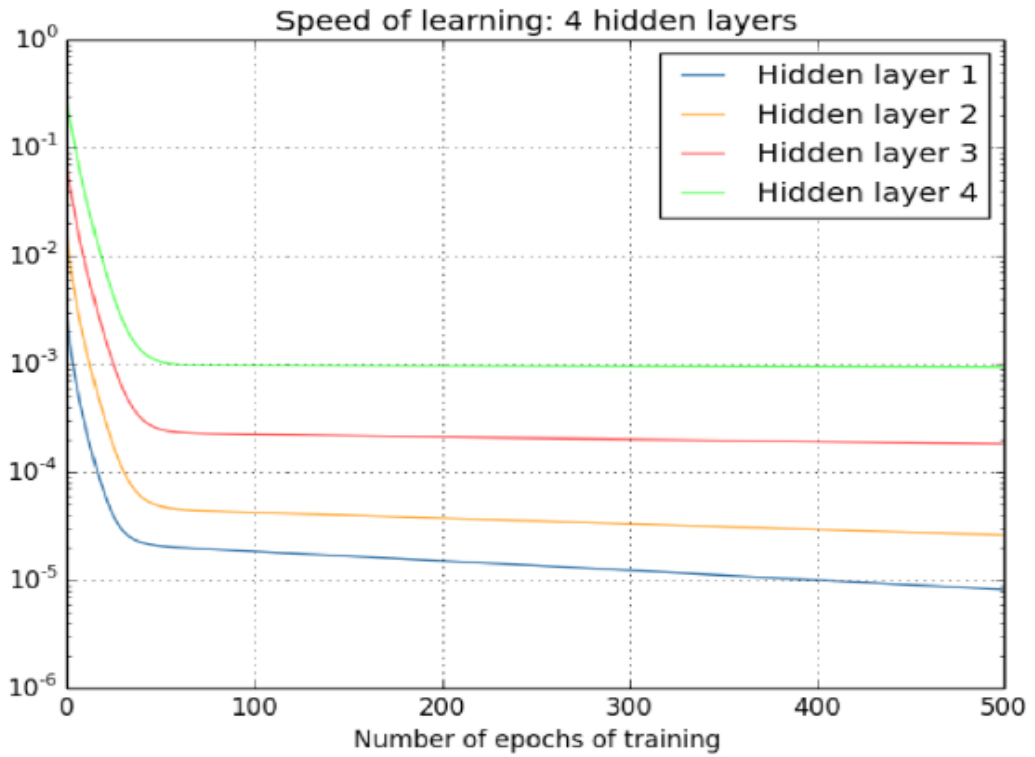
MLP – BACKPROPAGATION: SUPRESIÓN DEL GRADIENTE

MNIST

DOS CAPAS OCULTAS
DE 100 NEURONAS

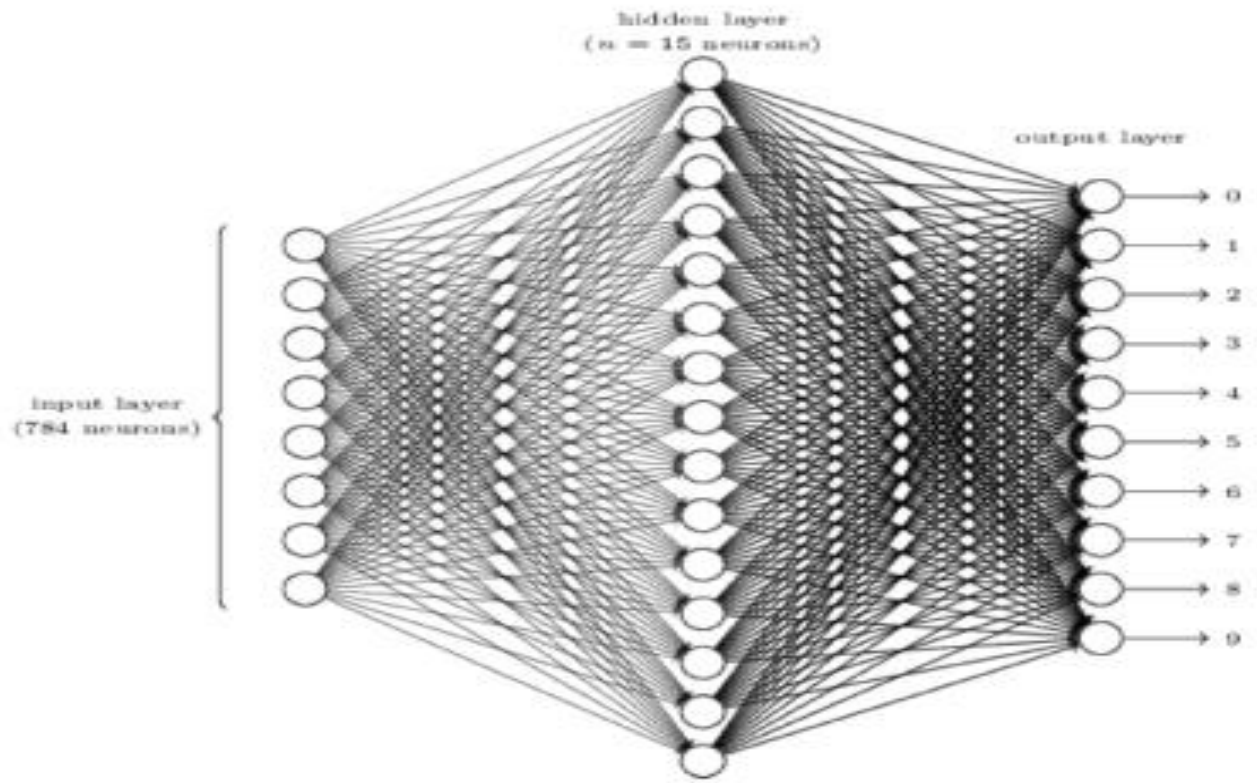


CUATRO CAPAS OCULTAS
DE 100 NEURONAS



MLP – BACKPROPAGATION: SUPRESIÓN DEL GRADIENTE

MNIST



SOLUCIÓN: AUTOENCODER

¡Gracias!