DEEP LEARNING :

mini-Batch Stochastic Gradient Descent (mB-SGD):

+

Pseudo-Inversa (Pinv)

Aprendizaje Deep Learning (DL)

FASE I: Pre-Tuning

Training Auto-Encoder:

- **Pesos Salida : Pseudo-Inversa (Pinv)**
- **Pesos Ocultos: Back-Propgation (BP) usando mini-Batch-SGD**

Training Softmax: Gradient Descent (GD)

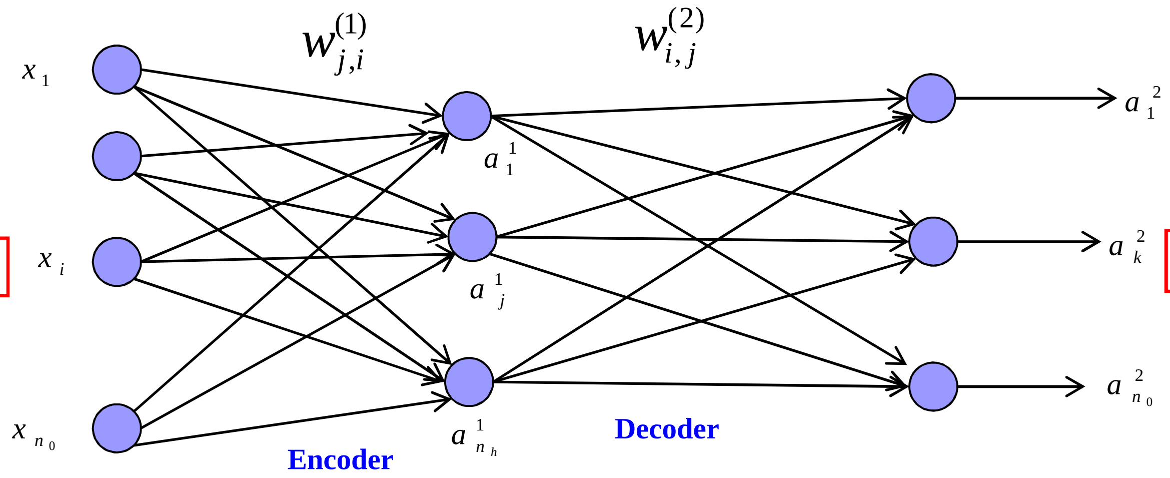
FASE II: Fine-Tuning

Training DL (o MLP):

- **Pesos Ocultos y Pesos de Salidas:**
  - **Back-Propgation (BP) Convencional usando método GD.**

AUTO-ENCODER (AE)

$$X = a^{(0)} \in \Re^{(n_0 \times N)}$$



$$a^{(2)} = \hat{X} \in \Re^{(n_0 \times N)}$$

$$z^{(1)} = w^{(1)} \times a^{(0)}$$
$$a^{(1)} = f\left(z^{(1)}\right)$$
$$a^{(1)} = \frac{1}{1 + \exp\left(-z^{(1)}\right)}$$

Activación Oculta: No-Lineal

$$z^{(2)} = w^{(2)} \times a^{(1)}$$
$$a^{(2)} = f\left(z^{(2)}\right)$$
$$a^{(2)} = z^{(2)}$$

Activación Salida: Lineal

Pesos de Salida : AE

- **Los Pesos Encoder son inicializados con valores aleatorios:**

Pesos Encoder:

$$r = \sqrt{\frac{6}{n_0 + n_h}}$$
$$w^{(1)} = rand\ (n_h, n_0) \times 2 \times r - r$$

Salida Encoder:

$$z^{(1)} = w^{(1)} \times a^{(0)}$$
$$H = \frac{1}{1 + \exp\left(- z^{(1)}\right)}$$

Pesos Decoder:

$$w^{(2)} = X \times H^T \times \left(H \times H^T + \frac{I}{C} \right)^{-1}$$

Pesos Ocultos AE:

Aprendizaje Back-Propagation usando Método mini-Batch SGD

Pesos Ocultos: BP+mB-SGD

- M: Tamaño del mini-Batch-SGD , con M<<N, donde N: Tamaño Data training

$$C = \frac{1}{2M}\sum_{n=1}^{M}\left\| d_n - a_n^{(2)} \right\|^2 = \frac{1}{2M}\sum_{n=1}^{M}\sum_{k=1}^{n_0}\left(d_{n,k} - a_{n,k}^{(2)}\right)^2 = \frac{1}{2M}\sum_{n=1}^{M}\sum_{k=1}^{n_0}\left(e_{n,k}^{(2)}\right)^2$$

Notación Matricial: Gradiente

$$\frac{\partial C}{\partial w^{(1)}} = \left\{\left(w^{(2)}\right)^T \times e^{(2)}\right\} \otimes f'\left(z^{(1)}\right) \times \left(a^{(0)}\right)^T$$

Notación Matricial: Update Pesos Ocultos

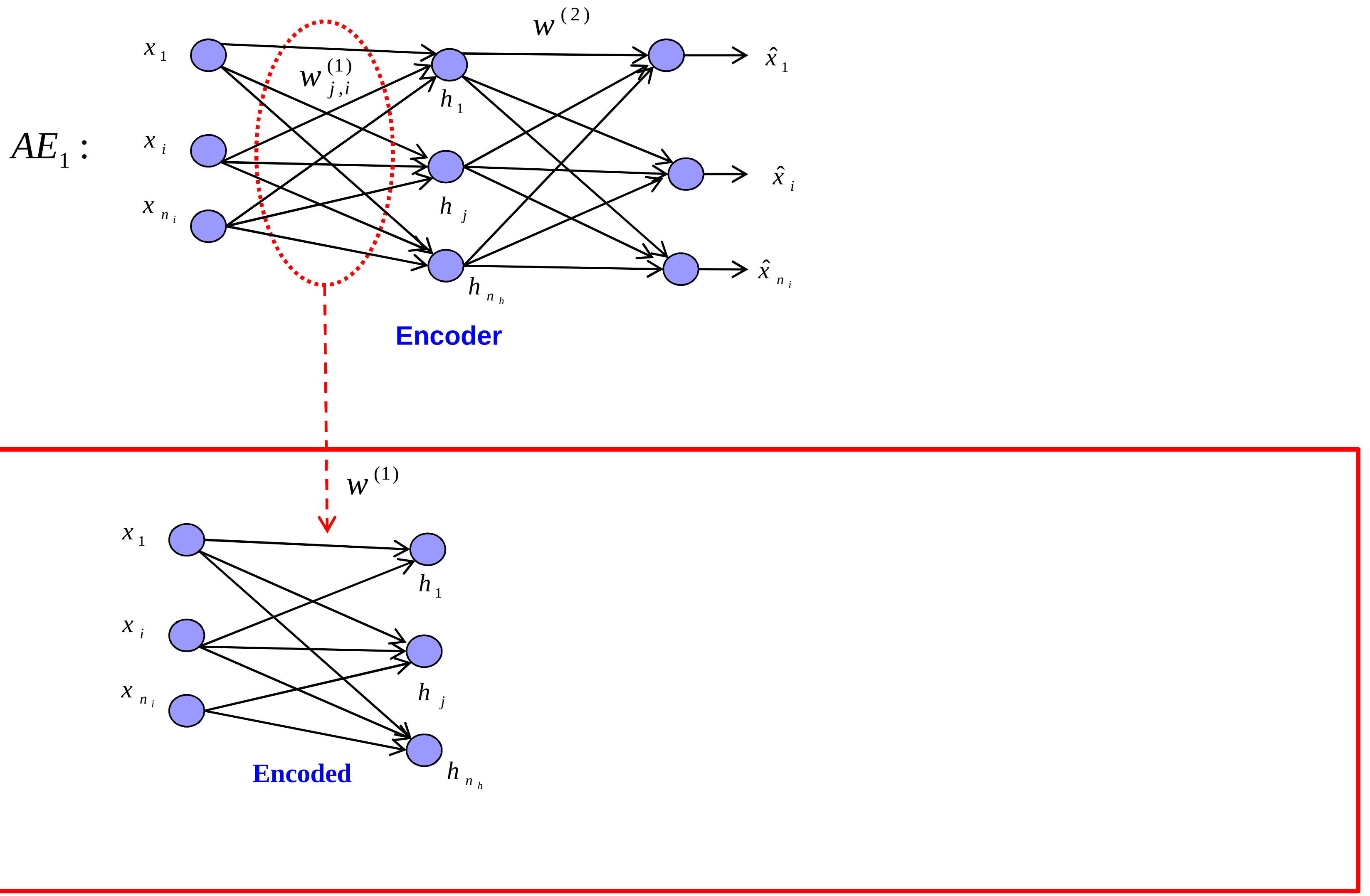
$$w^{(1)}(k) = w^{(1)}(k-1) + \mu \times \frac{\partial C}{\partial w^{(1)}}, \quad k = 1, \cdots, MaxEpoch$$

Tasa aprendizaje:

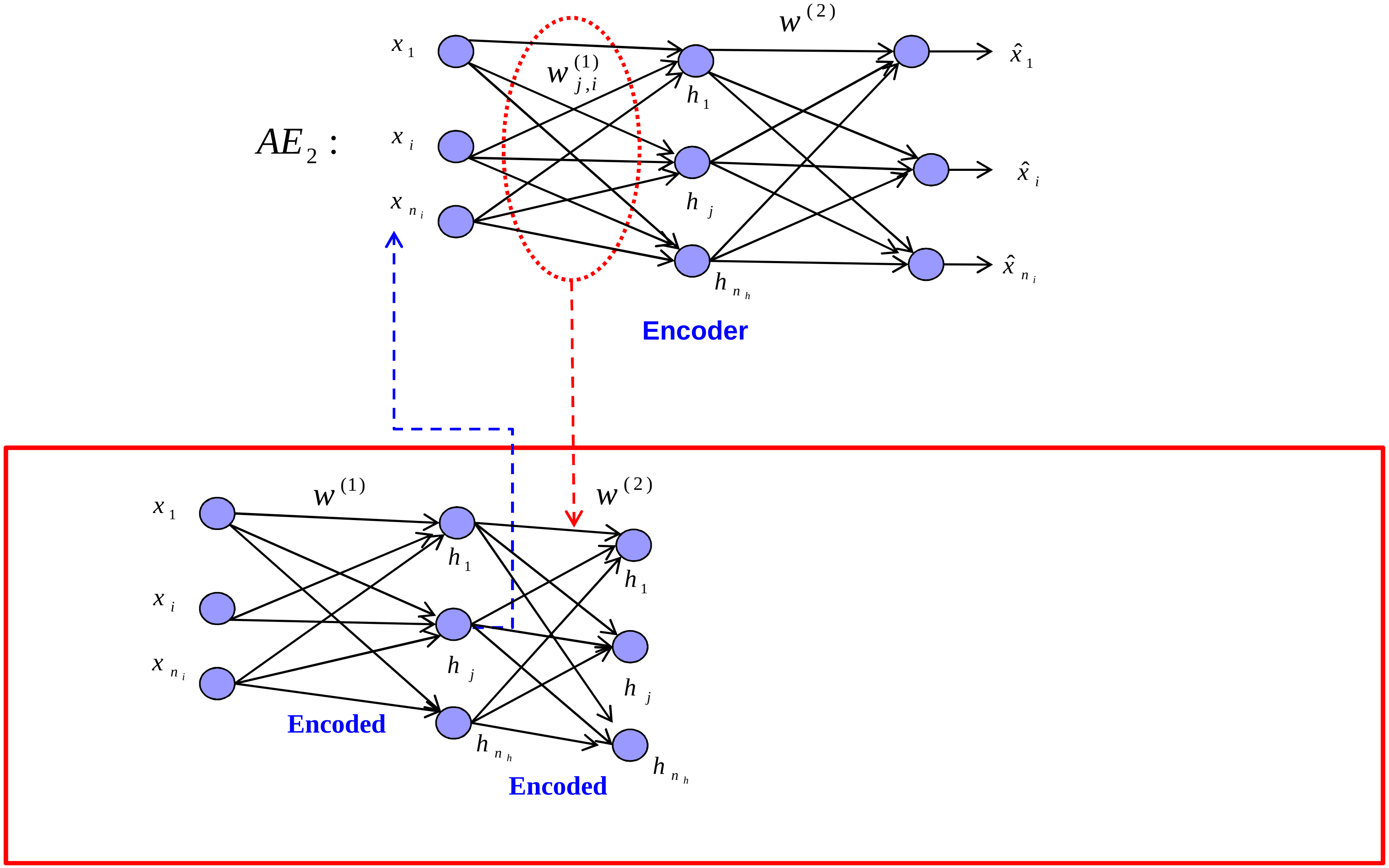
$$\tau = \frac{\mu_0}{MaxEpoch}$$

$$\mu = \frac{\mu_0}{1 + \tau \times \mathrm{Epoch(k)}}$$

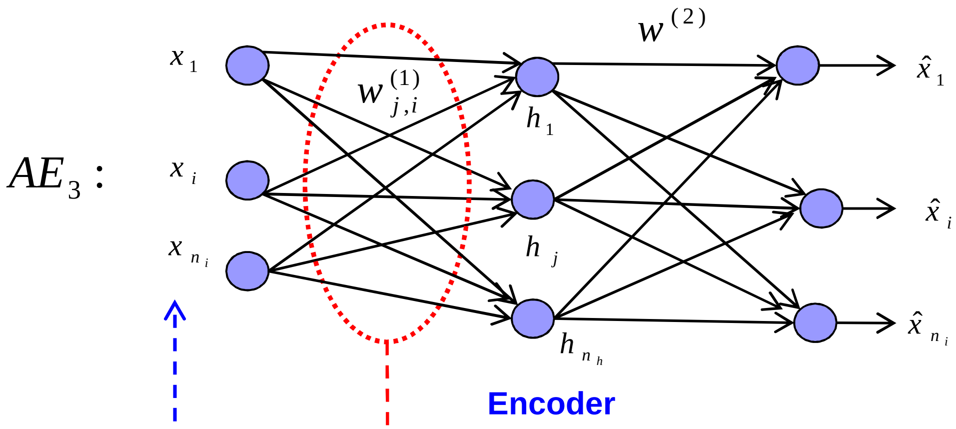
Método 2: Auto-Encoder Apilados (SAE)

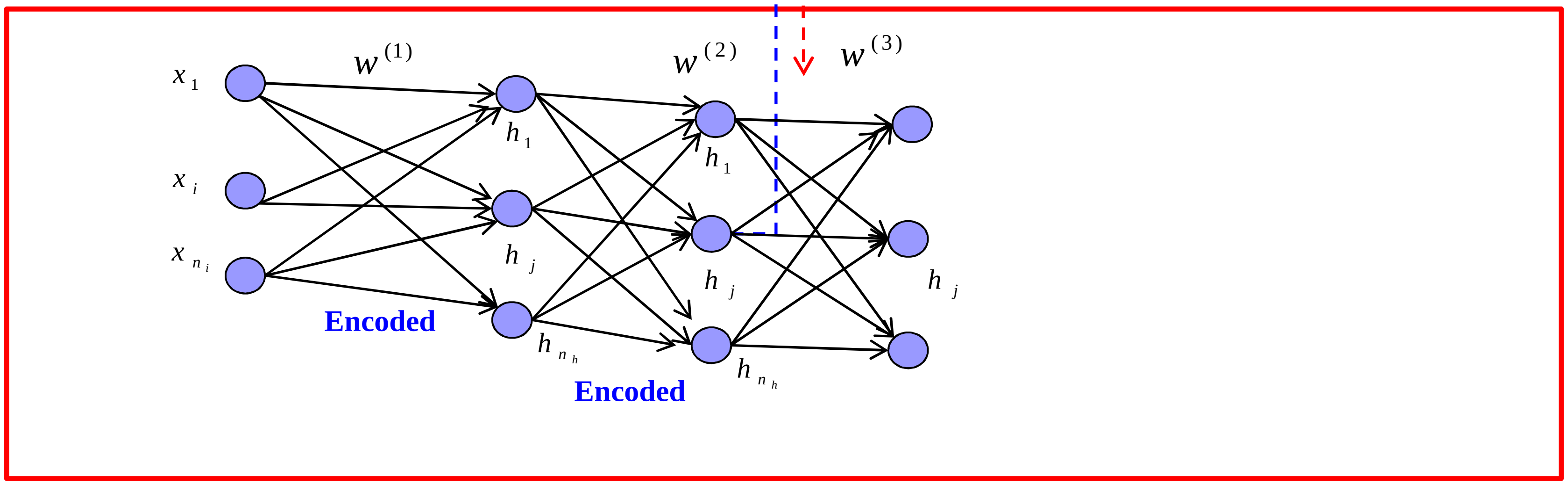


Método 2: (SAE)


AE_2 :
x_1
x_i
x_{n_i}
$w^{(1)}_{j,i}$
$w^{(2)}$
h_1
h_j
h_{n_h}
$\hat{x}_1$
$\hat{x}_i$
$\hat{x}_{n_i}$
Encoder
$w^{(1)}$
$w^{(2)}$
Encoded
Encoded

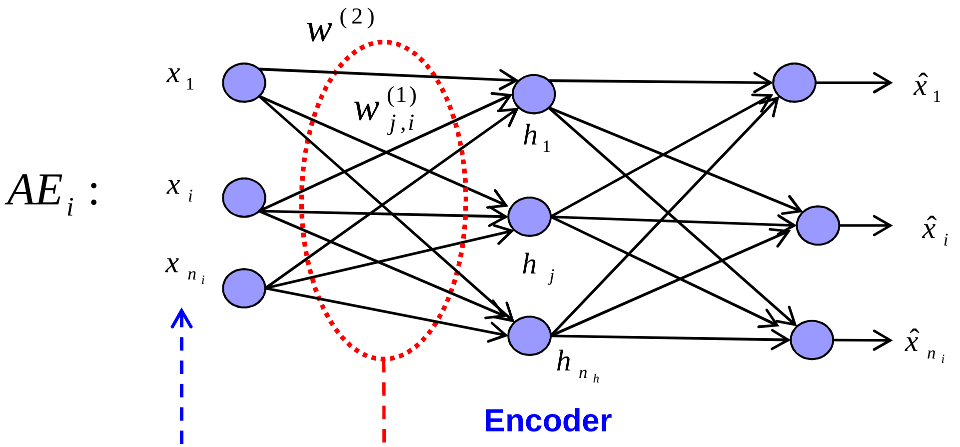
Método 2: (SAE)





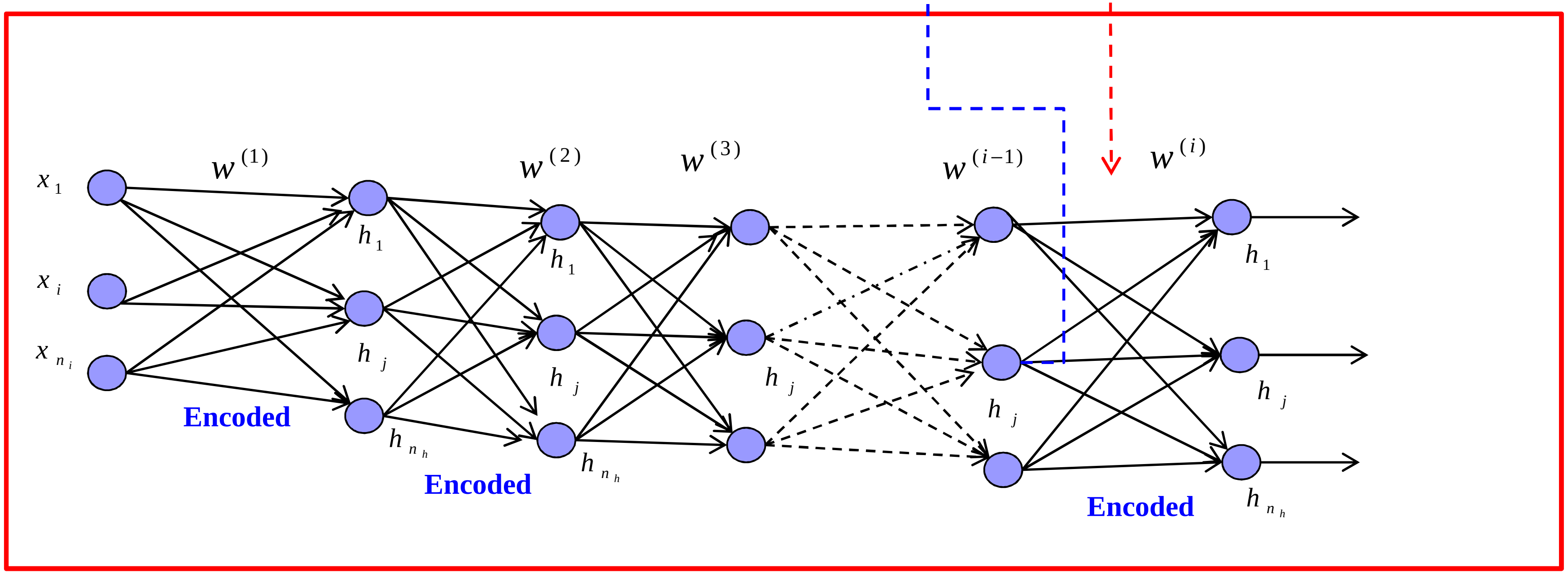
DEEP LEARNING

Método 2: (SAE)



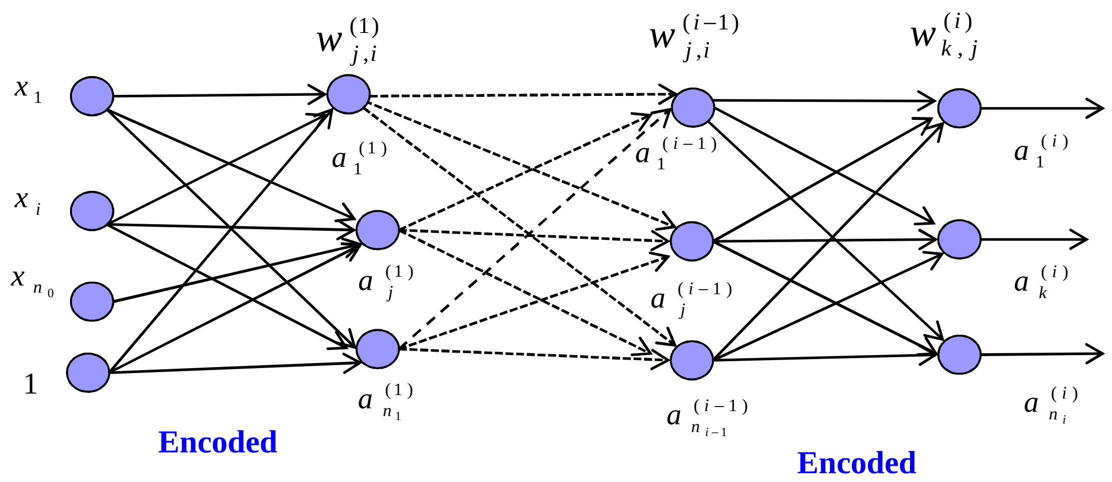
$$H^{(0)} = X$$

$$H^{(i)} = f\left(w^{(i)} \times H^{(i-1)}\right)$$

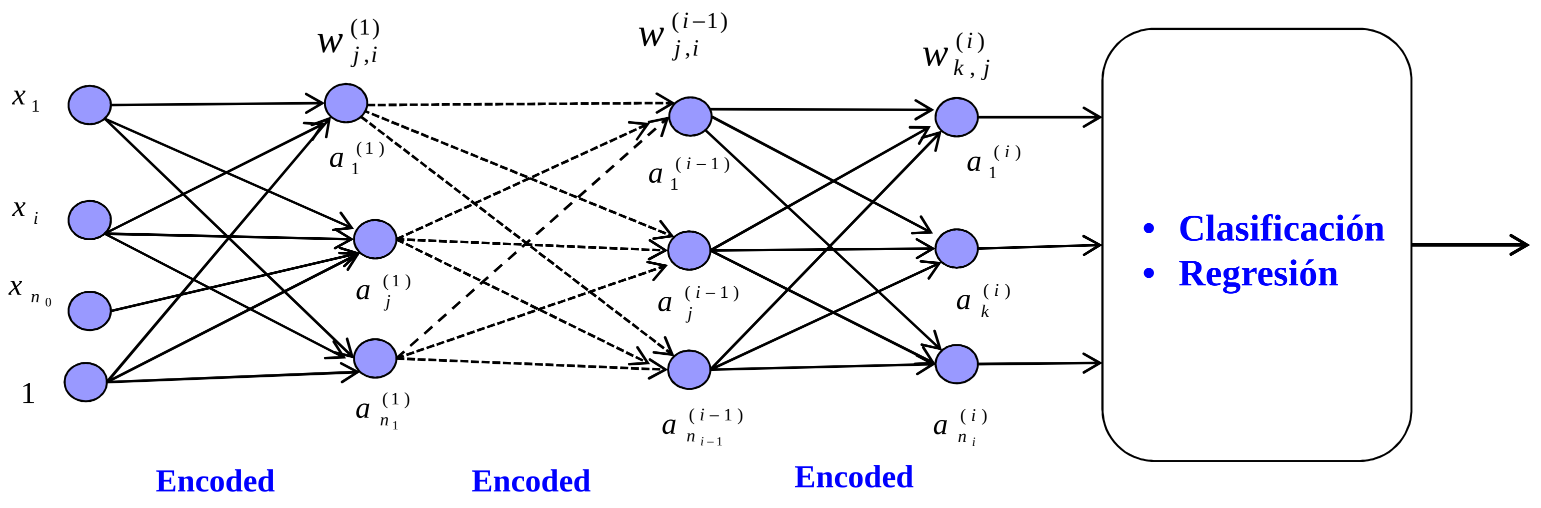


DEEP LEARNING

Training: Auto-Encoder Apilados (SAE)



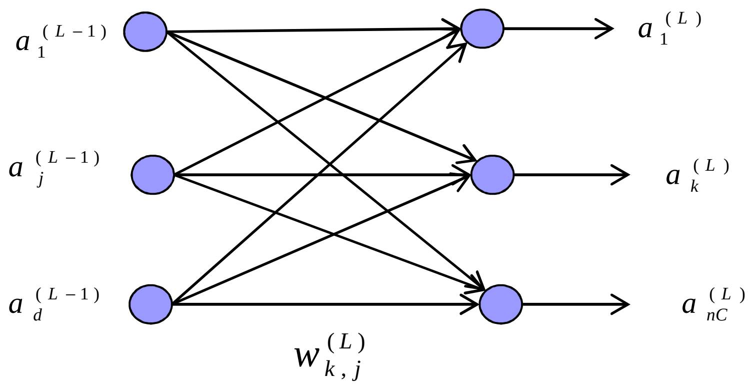
Training Clasificador Softmax



Clasificación Softmax

$X = a^{(L-1)} \in \Re^{(d \times N)}$

Data
Entrada



$a^{(L)} \in \Re^{(nC \times N)}$

Data
Salida

Algoritmo de Aprendizaje: SOFTMAX

Salida Softmax de la *n*-ésima muestra

$$\boxed{\begin{aligned} z &= w \times x \\ y &= \frac{\exp(z)}{\sum_{k=1}^{nC} \exp(z_k)} \end{aligned}}$$

$$\boxed{Cost = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{nC} t_{n,k} \log\left(y_{n,k}\right) + \frac{\lambda}{2}\|w\|_2^2}$$

- **N**: número de muestras, **nC**: numero de clases,
- **t**: valor deseado, **y:** salida softmax, ***lambda***: penalidad de pesos.

Notación Matricial Modificación de W

$$\boxed{\begin{aligned} w(k) &= w(k-1) - \mu \frac{\partial Cost}{\partial w}, \quad k = 1, \ldots, MaxIter \\ \frac{\partial Cost}{\partial w} &= -\frac{1}{N}\left((T - Y) \times X^T\right) + \lambda \times w(k-1) \end{aligned}}$$

Pseudo-code Deep Learning:

mini-Batch SGD

+

Pseudo-inversa

Pseudo-code: SAE_TRAIN

```
Function   sae_train(xe, argList)

for i=1  to numAEs
    wAE{i}          =  ae_train(xe, argList);          # W1: Encoded
     xe             =  Sigmoid(wAE{i}*xe);             # New Inputs
end

 return(wAE, xe)
```

Pseudo-code: AE_TRAIN

```
function  ae_train(xe,  argList)
w1              =  random_W(argList);
numBatch   =  floor(N/miniBatchSize);  tau = mu/MaxEpoch;
X               = xe; N=xe.shape[1];
For  Epoch=1  to  MaxEpoch
    Idx = np.random.permutation(N);   X=X(:,Idx);
    for iter=1: numBatch
          idx    = (iter-1)*miniBatchSize+1:iter*miniBatchSize;
          xe     = X(:,idx);

          w2   = ae_pinv(xe,w1,Cpar);
          mu   = mu_0/(1+tau*Epoch);
          w1  = ae_bp(xe,w1,w2, mu);
    endFor
endFor
return(w1)
```

Pseudo-code: AE_Pinv

```
function  ae_pinv(x, w1, C)

H     =  sigmoid(w1*x);

# Calcular  w2 con Pseudo-inversa

Completar code….

return(w2)
```

Pseudo-code: AE_BP

```
function  ae_bp(x, w1,w2,mu)

a   =  ae_forward(x,w1,w2);

E   =  x - act{3};

Delta_out        =  E;
Delta_hidden     =  (w2.T*Delta_out).*derivate_act(a{2});

gradW1           = Delta_hidden*act{1}.T;

w1               =  w1+mu*gradW1;

return(w1)
```

Pseudo-code: AE_forward

```
fucntion  ae_forward(x, w1, w2):

a{1}  =  x;
a{2}  =  sigmoid(w1*a{1});
a{3}  =  w2*a{2};

return(a)
```

Ejemplo:
CLASIFICACIÓN USANDO DEEP LEARNING

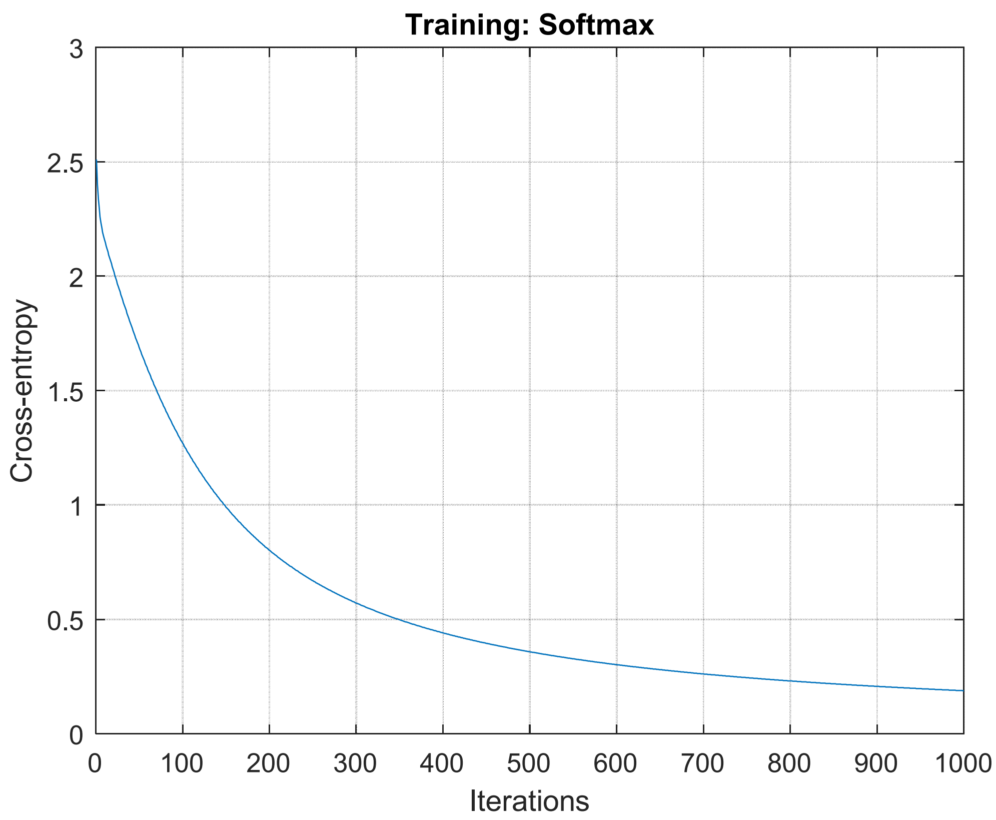
Ejemplo 1: Deep Learning con mini-Batch-SGD

- FASE I: Pre-Tuning
  - Número de muestras : 1600
  - Input : 256 puntos de amplitud de señal
  - Clases : 10 tipos de fallos mecánicos

  - Stack Auto-Encoder:
    - % Training : 0,8
    - Parámetro Pinv. (C) : 10
    - Máx. Epoch : 20
    - Tasa Aprendizaje (mu) : 0.01
    - AE1 Nodos Ocultos : 192
    - AE2 Nodos Ocultos : 128

  - Softmax-Clasificador:
    - Máx. Iteraciones : 10000
    - Tasa aprendizaje (mu) : 0.1
    - Penalidad (Lambda) : 0.0001

Fase I: Deep Learning



DL(256,192,128,10)

FASE I: Deep Learning

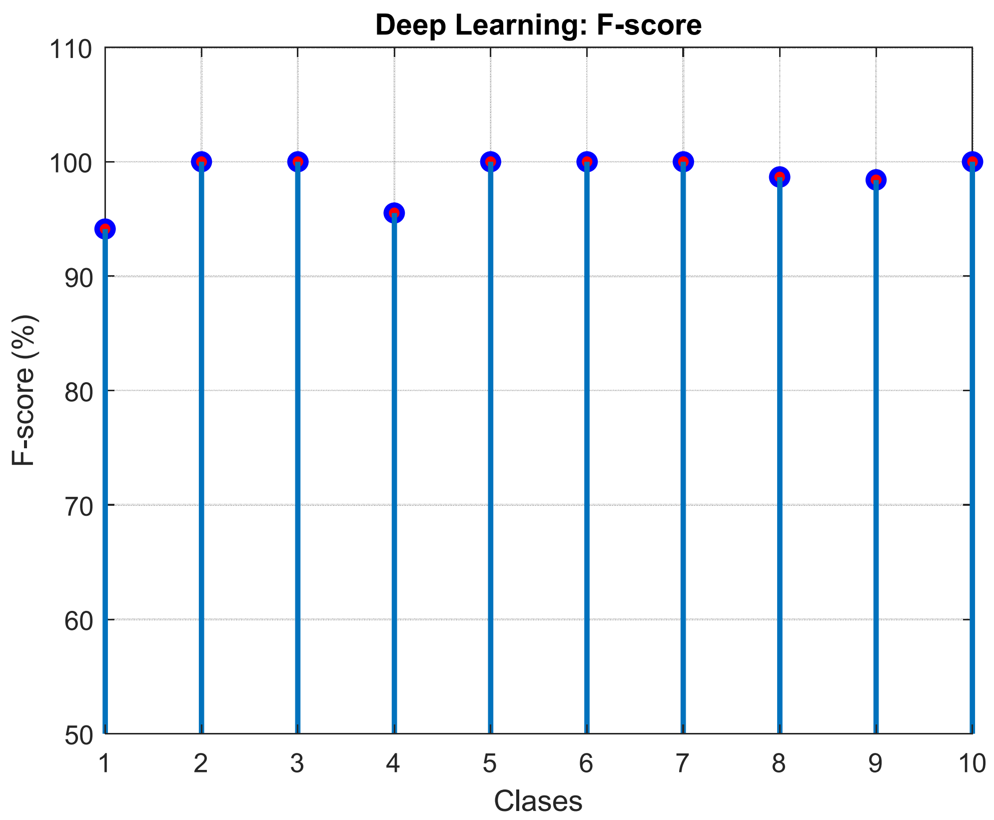
Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	
1	**24** 7.5%	**0** 0.0%	**0** 0.0%	**2** 0.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.3%	**0** 0.0%	88.9% 11.1%
2	**0** 0.0%	**28** 8.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
3	**0** 0.0%	**0** 0.0%	**34** 10.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
4	**0** 0.0%	**0** 0.0%	**0** 0.0%	**32** 10.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**1** 0.3%	**0** 0.0%	**0** 0.0%	97.0% 3.0%
5	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**33** 10.3%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
6	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**36** 11.3%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
7	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**26** 8.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
8	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**37** 11.6%	**0** 0.0%	**0** 0.0%	100% 0.0%
9	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**31** 9.7%	**0** 0.0%	100% 0.0%
10	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**35** 10.9%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	94.1% 5.9%	100% 0.0%	100% 0.0%	100% 0.0%	97.4% 2.6%	96.9% 3.1%	100% 0.0%	**98.8%** **1.2%**

Target Class

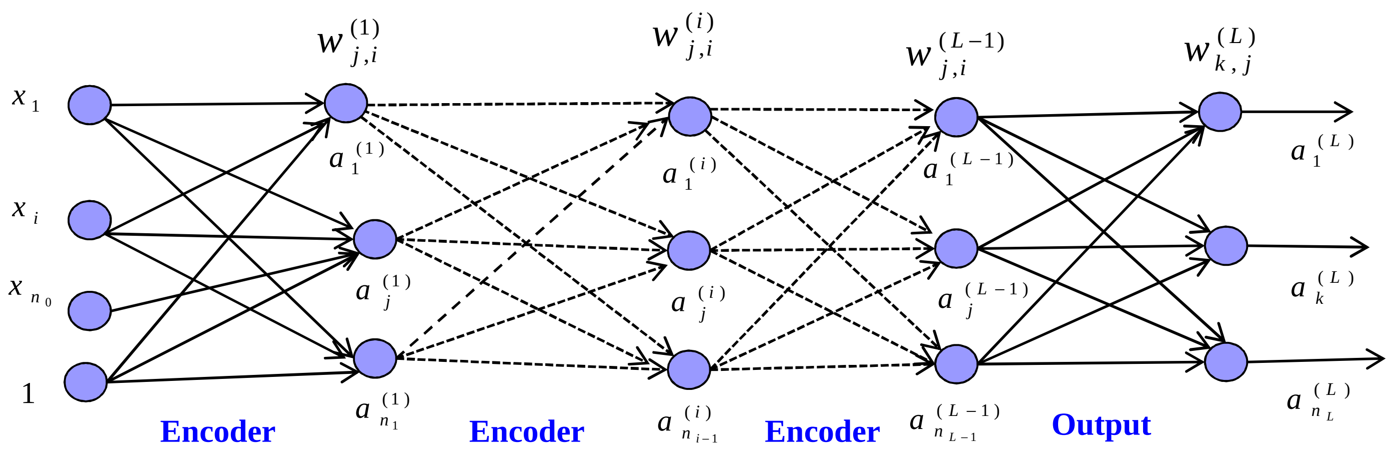
DL(256,192,128,10)

FASE I: Deep Learning



Average F-scores (%): 98.67

Fase II Fine-Tuning : Back-propagation Convencional



Fine Tuning: Pesos de Salida con BP

Capa de Salida del DL:

$$z_k^{(L)} = w_{k,j}^{(L)} a_j^{(L-1)}$$

$$a_k^{(L)} = \frac{\exp(z_k^{(L)})}{\sum_{i=1}^{nC} \exp(z_i)}$$

Función de Costo:

$$E = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{nC} T_{n,i} \log\left(a_{n,i}^{(L)}\right)$$

- **N**: número de muestras, **nC**: numero de clases, **T**: valor deseado.

Update Pesos de Salida:

$$w^{(L)}(m) = w^{(L)}(m-1) - \mu \frac{\partial E}{\partial w^{(L)}}, \quad m = 1, \dots, MaxIter$$

$$\delta^{(L)} = a^{(L)} - T$$

$$\frac{\partial E}{\partial w^{(L)}} = \frac{1}{N}\left\{\delta^{(L)} \times \left(a^{(L-1)}\right)^T\right\}$$

Fine Tuning: Pesos Ocultos con BP

- Para Cada Oculta: desde (L-1) hasta la Capa 1

Update Pesos Ocultos:

$$w^{(l)}(m) = w^{(l)}(m-1) - \mu \frac{\partial E}{\partial w^{(l)}},$$

$$l = L-1, L-2, \ldots, 1$$

$$m = 1, \ldots, MaxIter$$

$$\delta^{(l)} = \left\{ \left(w^{(l+1)}\right)^T \times \delta^{(l+1)} \right\} \otimes f'\left(z^{(l)}\right)$$

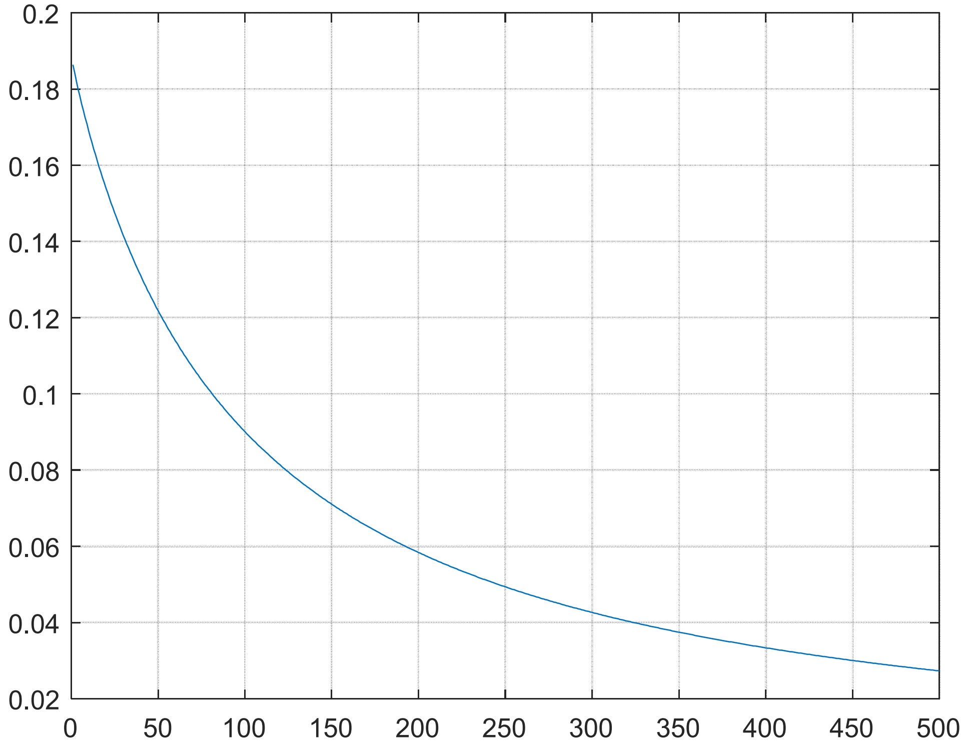
$$\frac{\partial E}{\partial w^{(l)}} = \delta^{(l)} \times \left(a^{(l-1)}\right)^T$$

Ejemplo 2: Deep Learning con mini-Batch-SGD

- FASE I: Pre-Tuning
  - Stack Auto-Encoder:
    - % Training : 0,8
    - Parámetro Pinv. (C) : 10
    - Máx. Epoch : 20
    - Tasa Aprendizaje (mu) : 0.01
    - AE1 Nodos Ocultos : 192
    - AE2 Nodos Ocultos : 128
  - Softmax-Clasificador:
    - Máx. Iteraciones : 10000
    - Tasa aprendizaje (mu) : 0.1
    - Penalidad (Lambda) : 0.0001

- FASE II: Fine-Tuning
  - Deep-Learining +Back-propagation:
    - % Training : 0,8
    - Máx. Epoch : 500
    - Tasa aprendizaje(mu) : 0.5*1e-3

Deep Learning



DL(256,192,128,10)

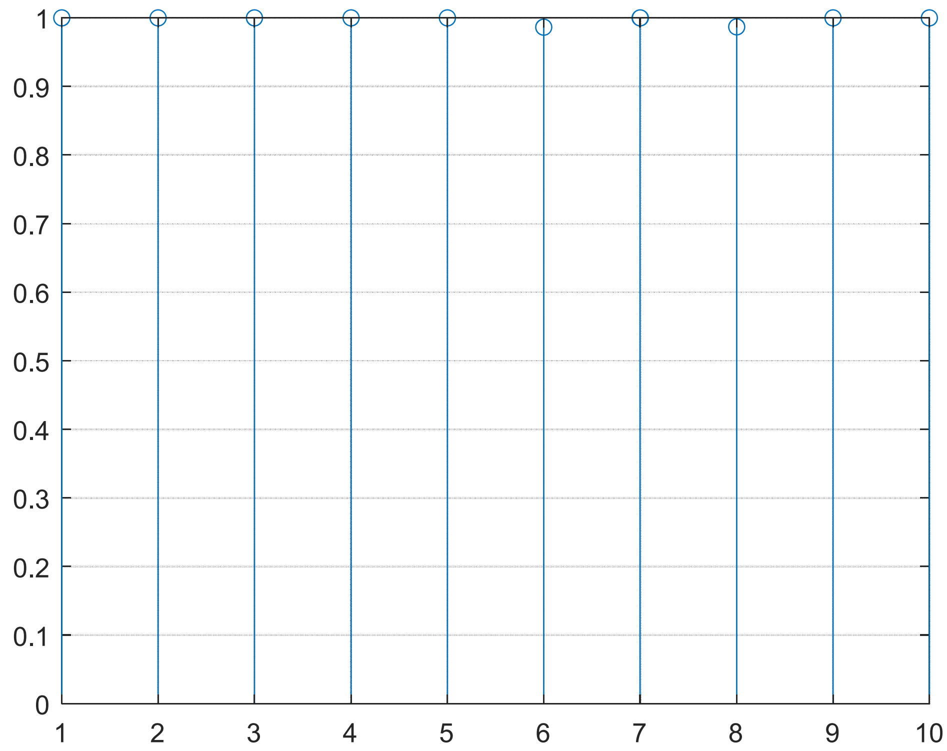
Deep Learning

Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	6	7	8	9	10	
1	**24** 7.5%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
2	**0** 0.0%	**28** 8.8%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
3	**0** 0.0%	**0** 0.0%	**34** 10.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
4	**0** 0.0%	**0** 0.0%	**0** 0.0%	**34** 10.6%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
5	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**33** 10.3%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
6	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**36** 11.3%	**0** 0.0%	**1** 0.3%	**0** 0.0%	**0** 0.0%	97.3% 2.7%
7	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**26** 8.1%	**0** 0.0%	**0** 0.0%	**0** 0.0%	100% 0.0%
8	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**37** 11.6%	**0** 0.0%	**0** 0.0%	100% 0.0%
9	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**32** 10.0%	**0** 0.0%	100% 0.0%
10	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**0** 0.0%	**35** 10.9%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	100% 0.0%	97.4% 2.6%	100% 0.0%	100% 0.0%	**99.7%** **0.3%**

DL(256,192,128,10)

Deep Learning



Average F-scores (%): 99.75

CONTINUARÁ....