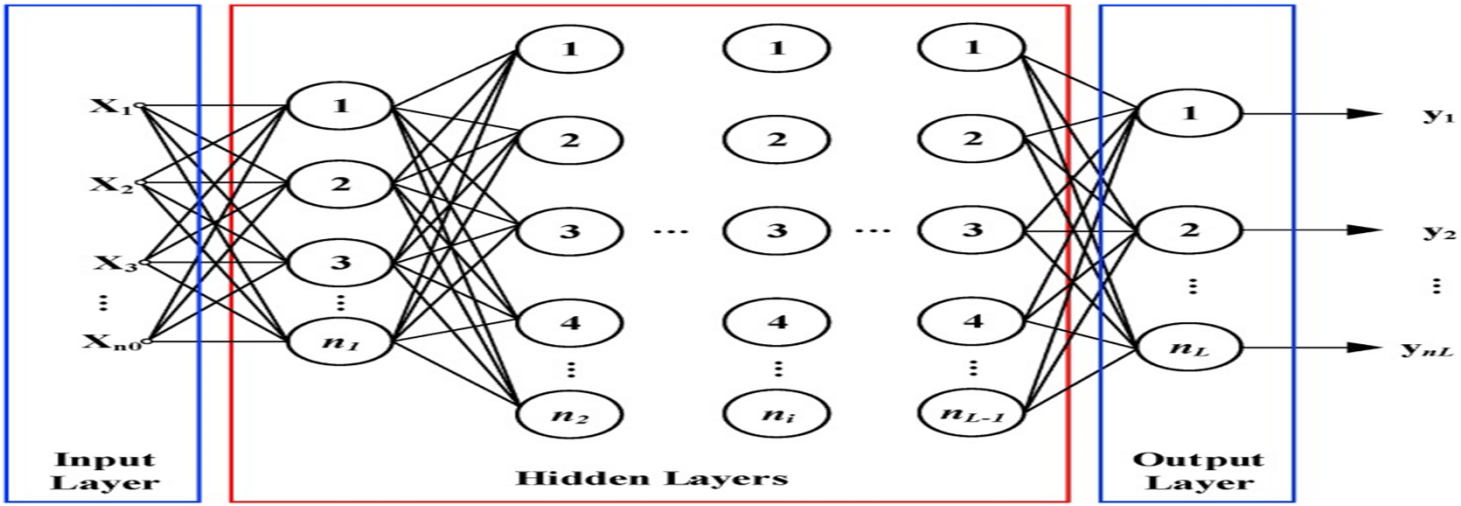
DEEP LEARNING (DL)
(APRENDIZAJE PROFUNDO)

Topológica de Deep Learning



n_0 = Número de nodo de la Capa Entrada

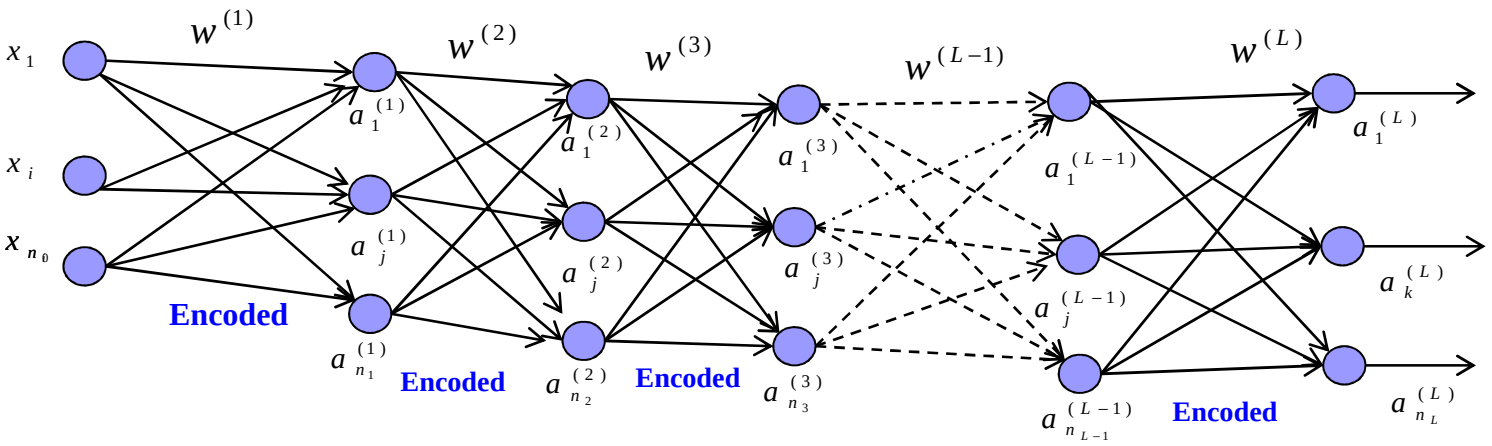
n_i = Número de nodo de la Capa Oculta i - ésima

n_L = Número de nodo de la Capa Salida

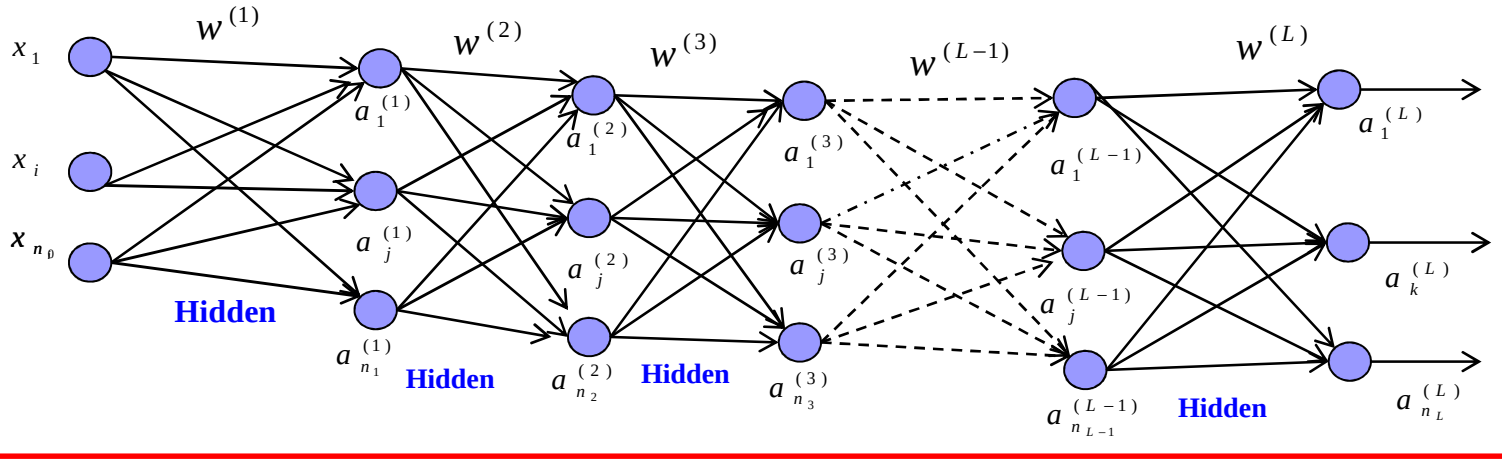
- **Es un «*nuevo*» Multi-Layer Perceptron**
- **Cuál es la diferencia con Antiguo MLP ?**

Multi-layer Perceptron:

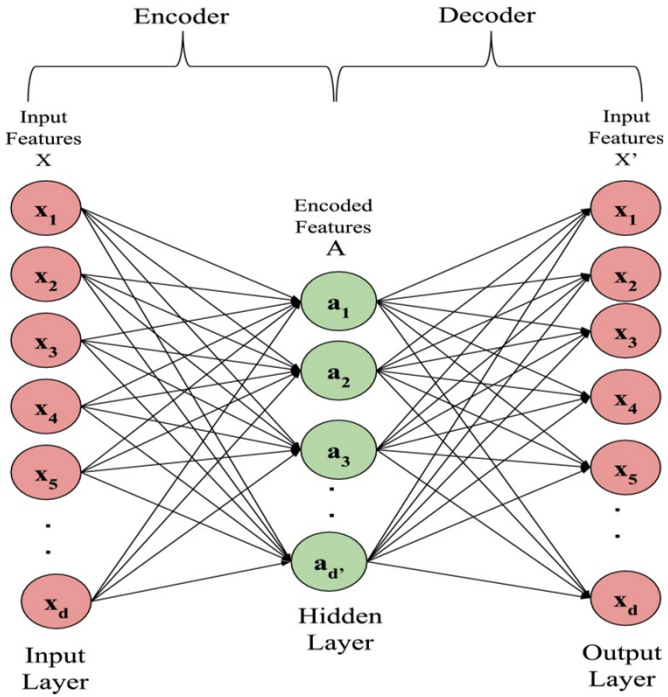
DEEP LEARNING (DL)



Shallow Neural Network (SNN)



Unidad Básica : AUTO-ENCODER (AE)



Nodos Ocultos

(1) $n_h < n_i$

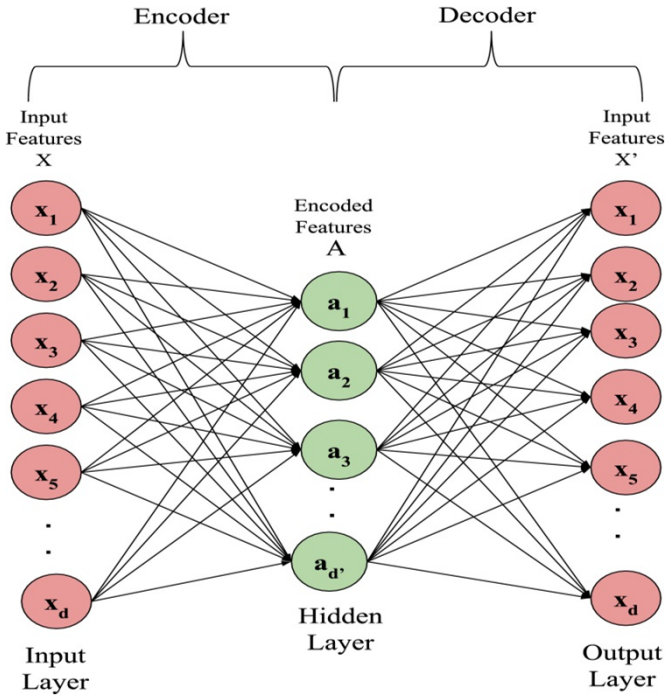
(2) $n_h = n_i$

(3) $n_h > n_i$

Proceso Deep Learning :

- **FASE 1: PRE-TRANING**
  - Aprendizaje No-supervisado *(auto-supervisado)*
    - Optimización de pesos de Auto-encoder.
- **FASE 2: FINE-TUNING**
  - Aprendizaje Supervisado
  - Optimización de todos los pesos del DL

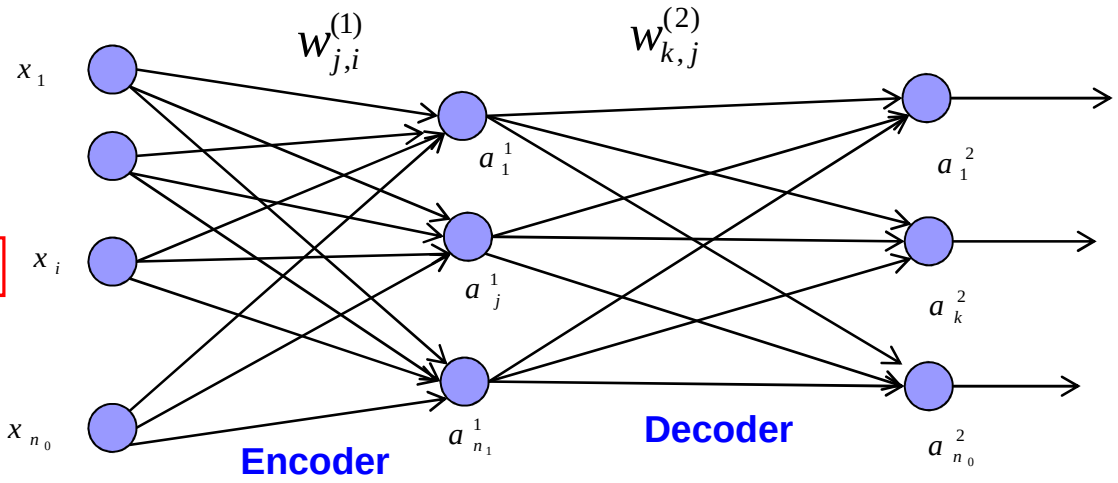
FASE 1: Entrenamiento Auto-Encoder (AE).



- **STEP 1:** Pesos Decoder**:**
  - Algoritmo Pseudo-inversa.
  - Algoritmo Back-Propagation.
- **STEP 2**: Pesos Encoder **:**
  - Algoritmo Back-Propagation.

Estructura de 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

Entrenamiento Pesos Decoder:

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

$$r = \sqrt{\frac{6}{n_1 + n_0}}$$

$$w^{(1)} = rand\ (n_1, n_0) \times 2 \times r - r$$

Salida Encoder

$$z^{(1)} = w^{(1)} \times a^{(0)}$$

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

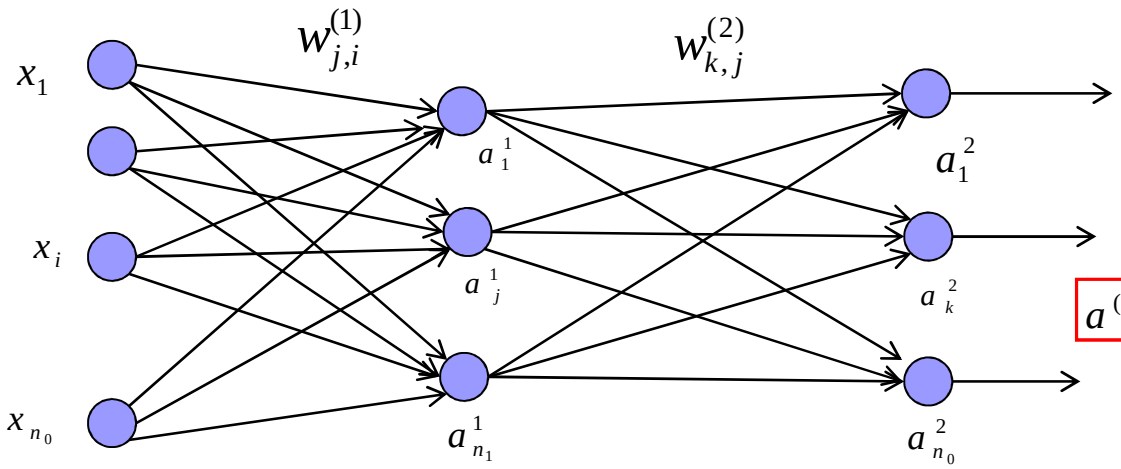
Pesos Decoder

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

Entrenamiento Pesos Encoder:

- **Algoritmo Back-propgation : Descenso del Gradiente**

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



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

$$E = \frac{1}{2}\sum_{n=1}^{N} C_n = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{n_0}\left(a^{(2)}_{k,n} - x_{k,n}\right)^2$$

Función de Costo

Gradiente Encoder:

Notación Matricial: Gradiente

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

Notación Matricial: Actualización Pesos Encoder

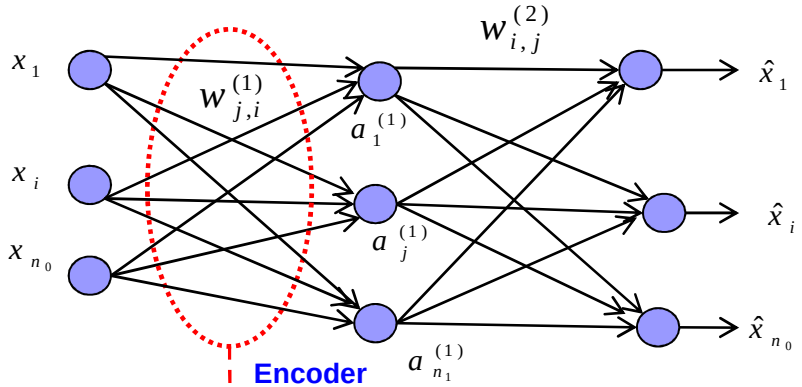
$$w^{(1)}(t) = w^{(1)}(t-1) - \mu \times \frac{\partial C}{\partial w^{(1)}}, \quad \mathrm{t} = 1, \cdots, MaxIter$$

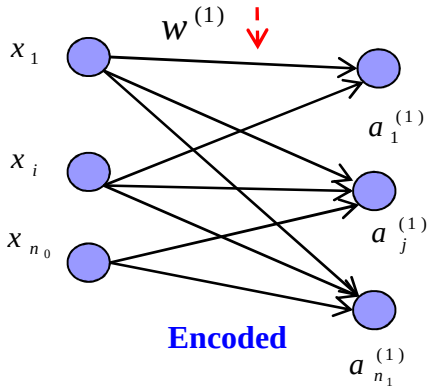
Construcción de DL usando AE Apilados (AEA)

AE_1 :

INPUT:

$X(n_0, N)$

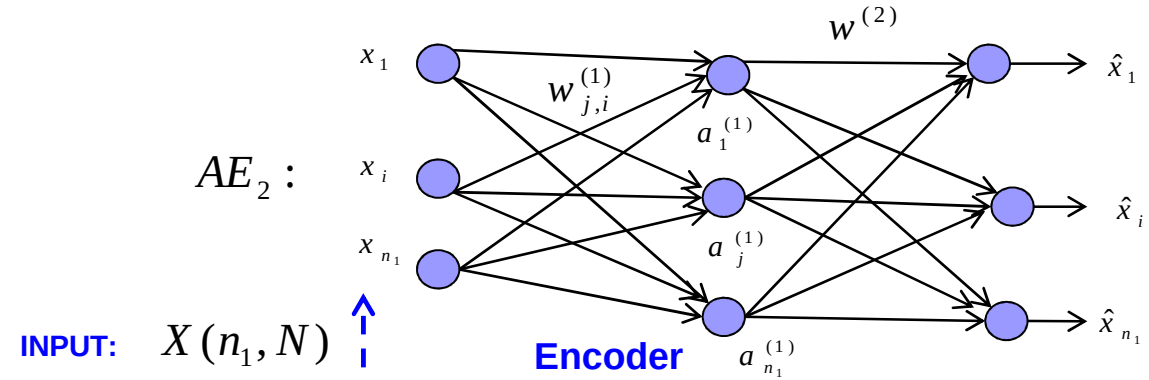


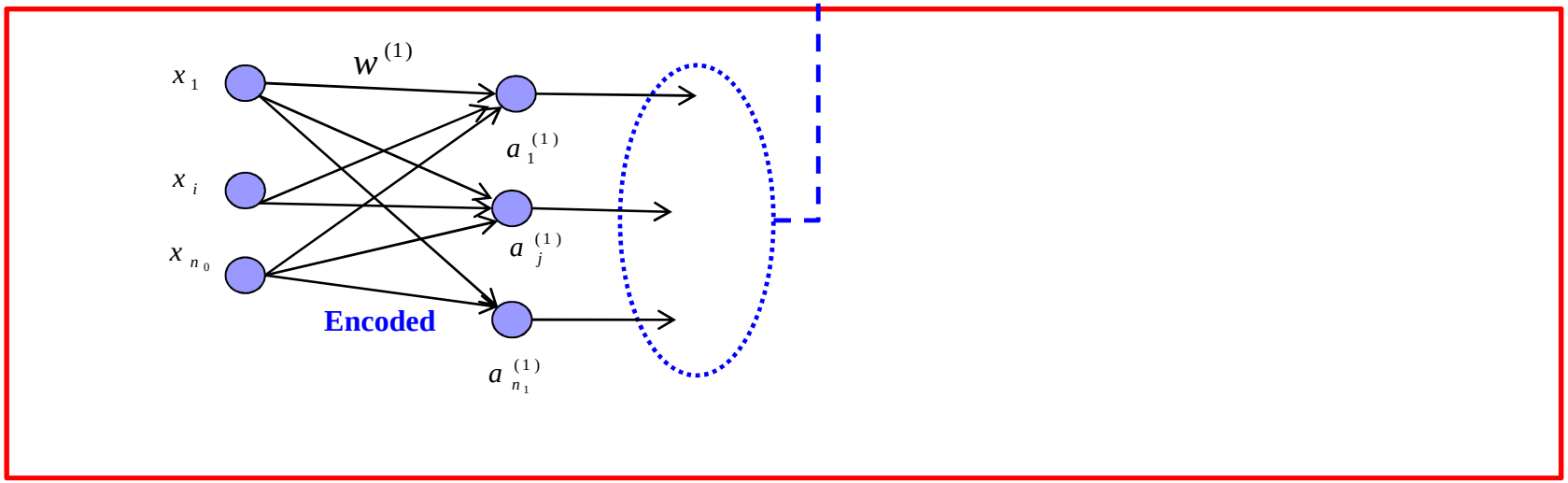


Steps:

1. Entrenar el AE
2. Retornar los pesos del Encoder para diseñar la primera capa del DL

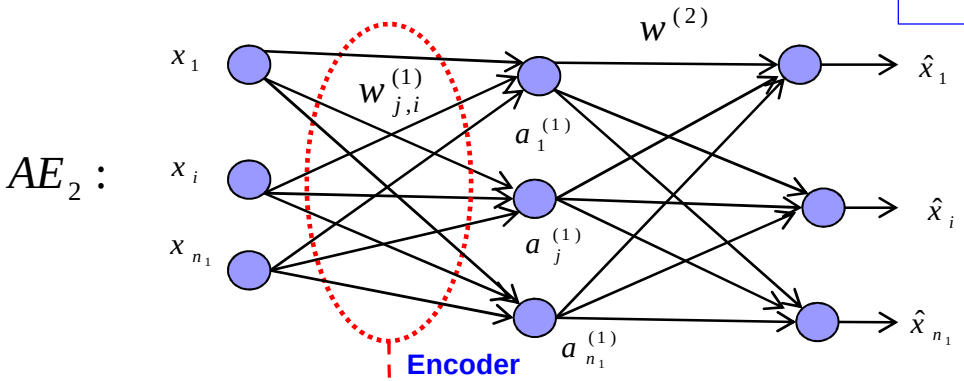
DL usando AE

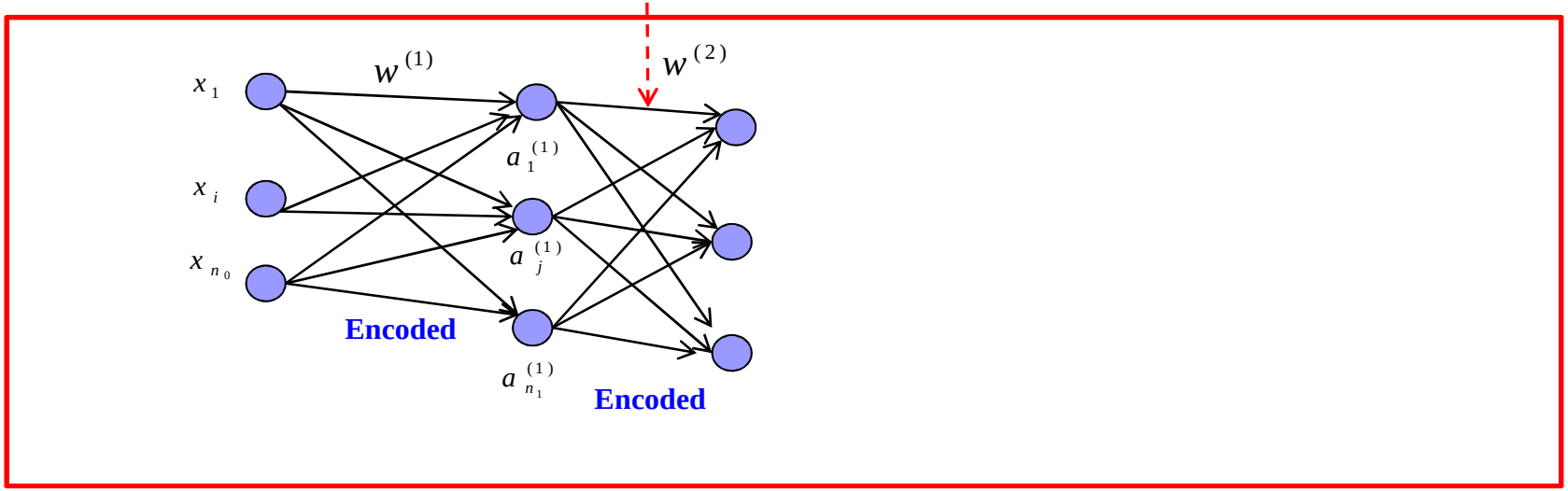

AE_2 :
INPUT: $X(n_1, N)$
x_1
x_i
x_{n_1}
$w^{(1)}_{j,i}$
$w^{(2)}$
$a^{(1)}_1$
$a^{(1)}_j$
$a^{(1)}_{n_1}$
$\hat{x}_1$
$\hat{x}_i$
$\hat{x}_{n_1}$
Encoder


x_1
x_i
x_{n_0}
$w^{(1)}$
$a^{(1)}_1$
$a^{(1)}_j$
$a^{(1)}_{n_1}$
Encoded

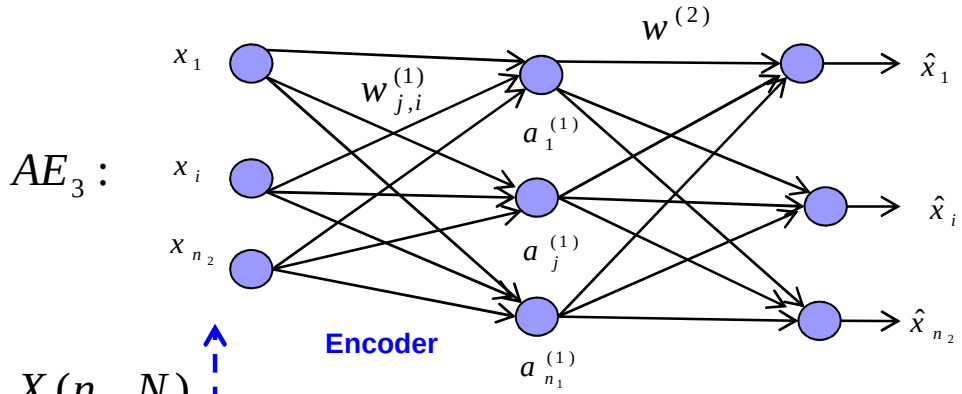
DL usando AE

1. Entrenar el AE
2. Retornar los pesos del Encoder para diseñar la segunda capa del DL

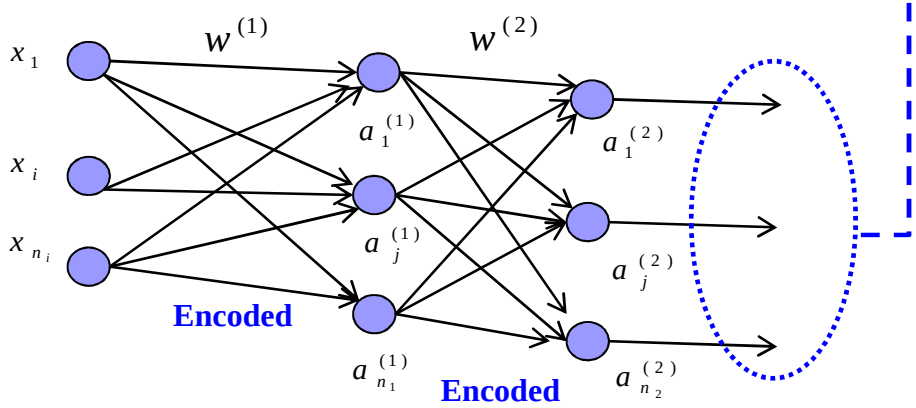




DL usando AE



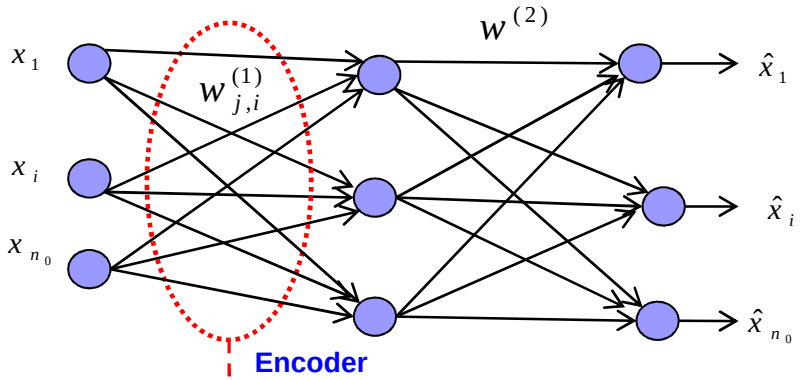
INPUT: $X(n_2, N)$

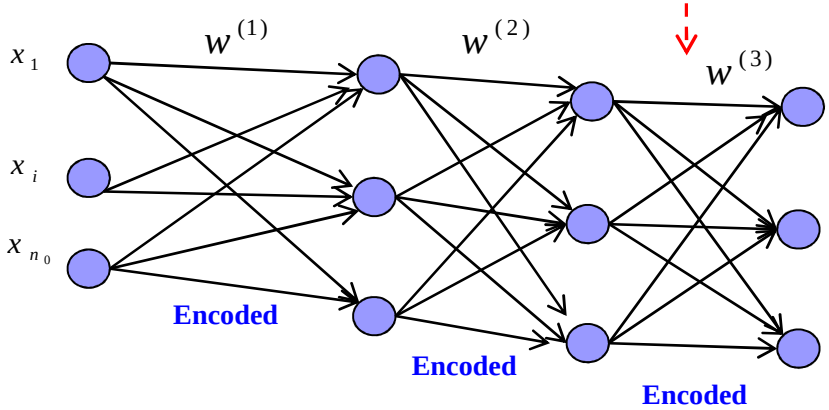


DL usando AE

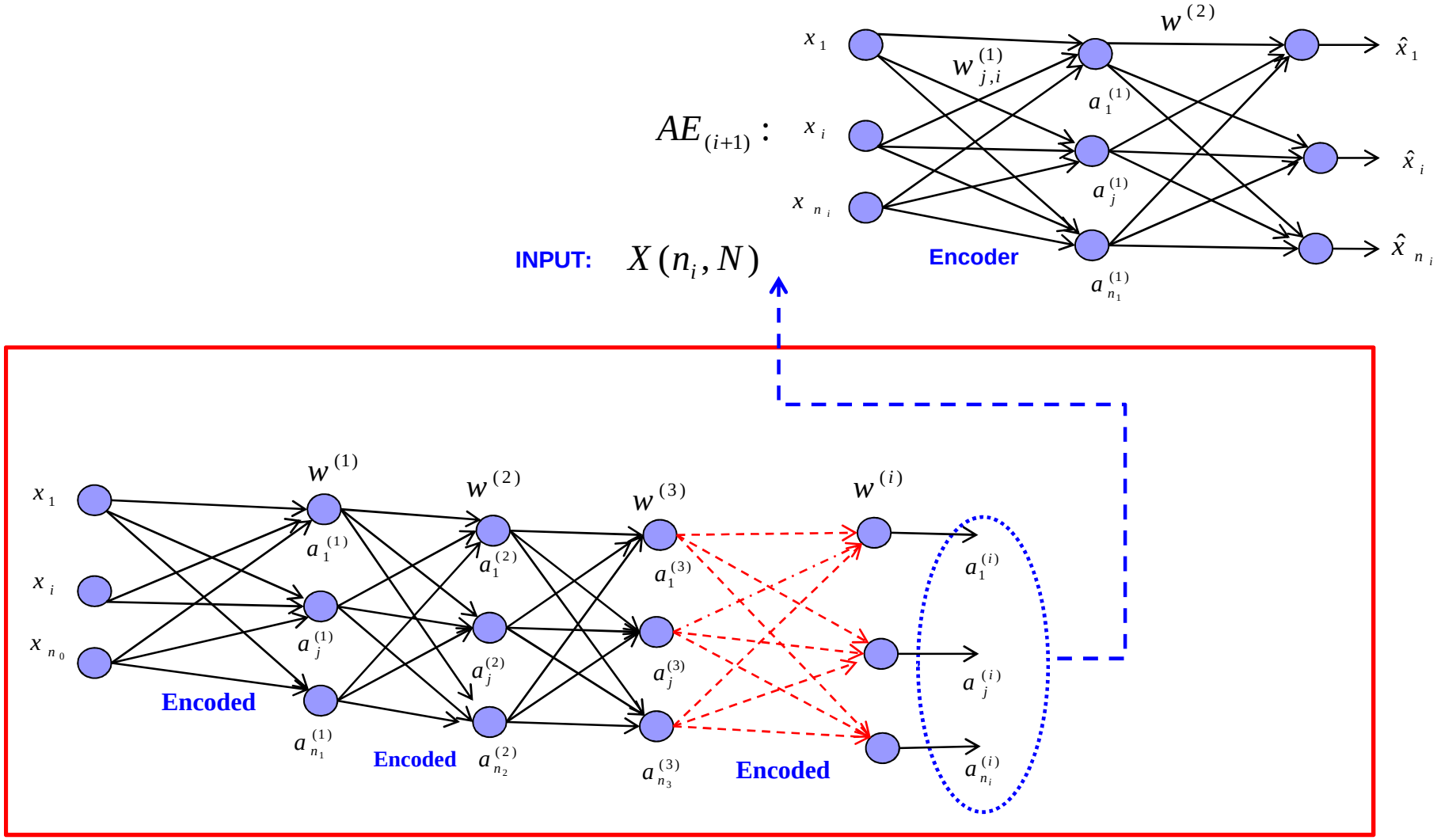
- Entrenar el AE
- Retornar los pesos del Encoder para diseñar la tercera capa del DL

AE_3 :





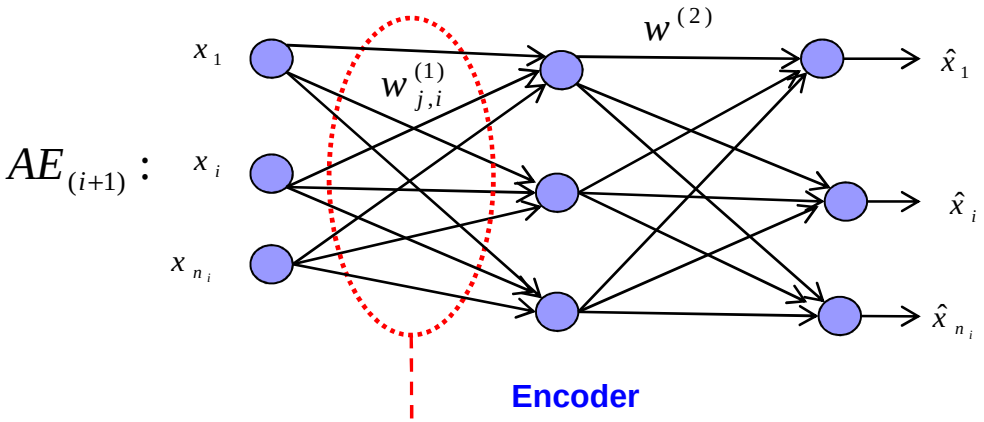
DL usando AE

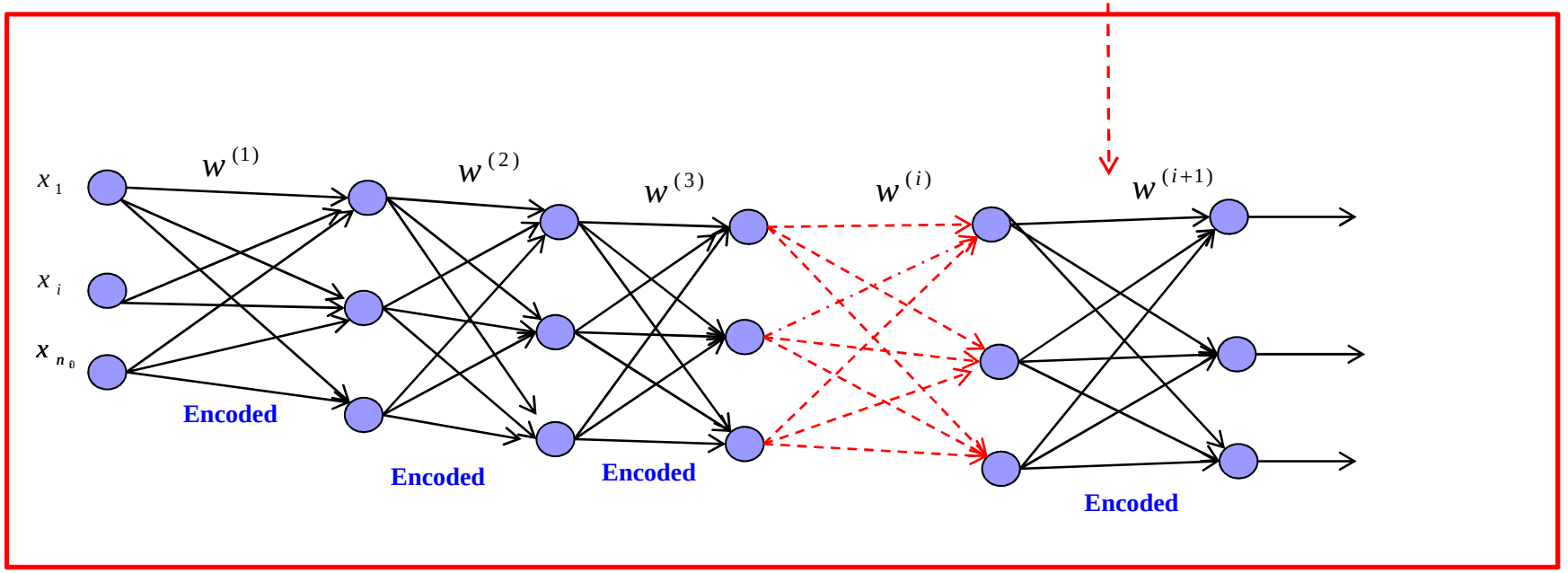


DEEP LEARNING

DL usando AE

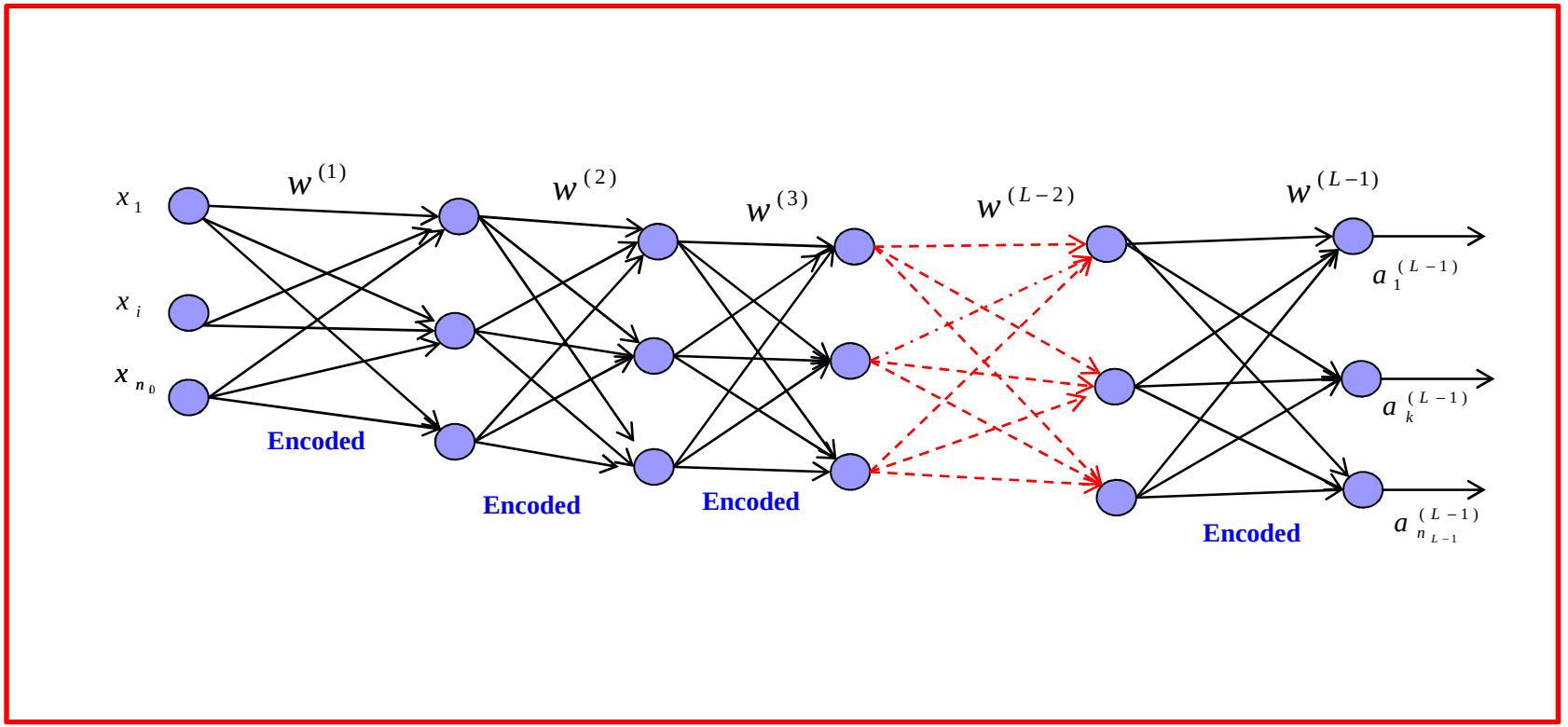
- Entrenar el AE
- Retornar los pesos del Encoder
  para diseñar la (***i+1)-ésima*** capa del DL



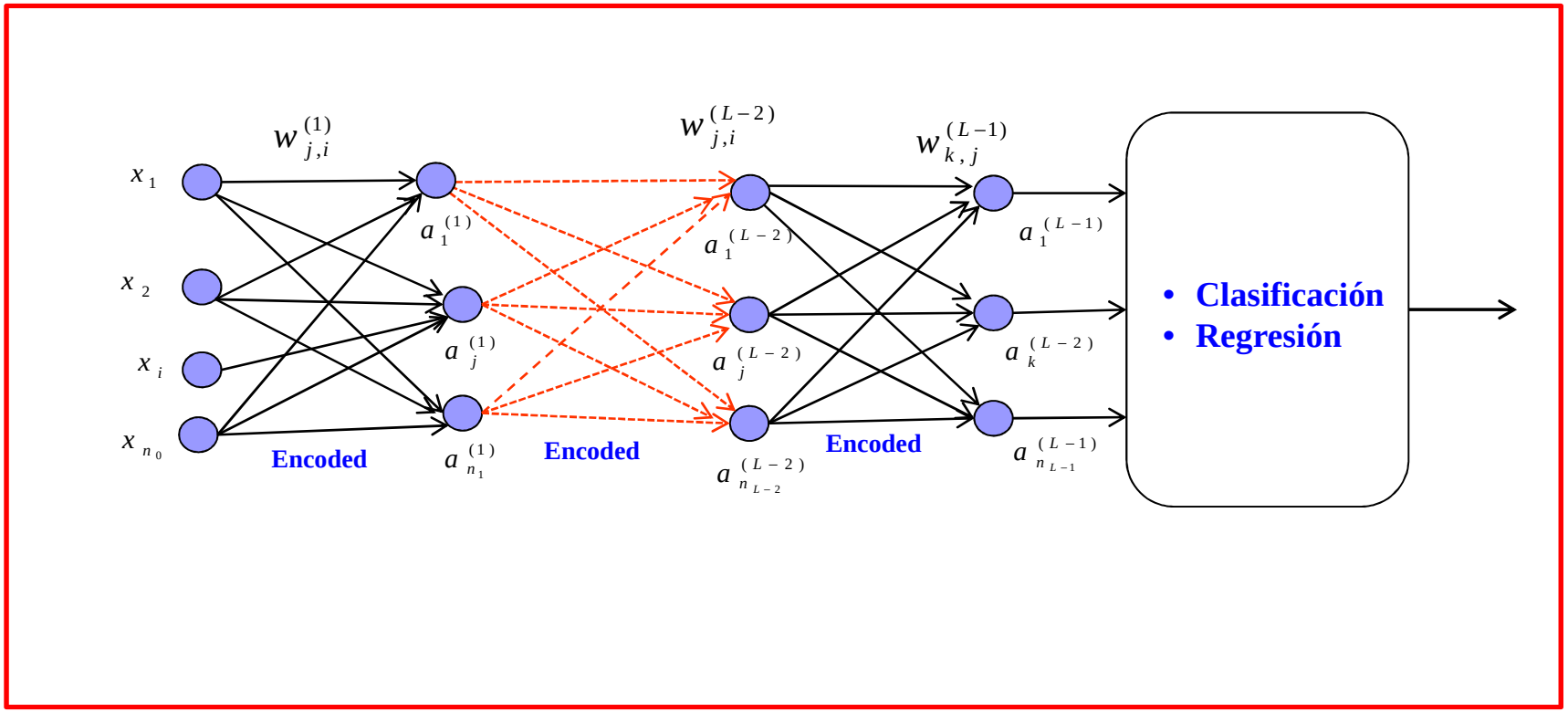


DEEP LEARNING

Construción de DL usando AE Apilados

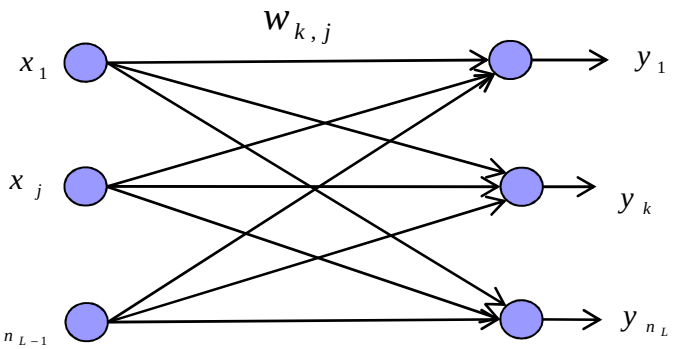


Deep Learning

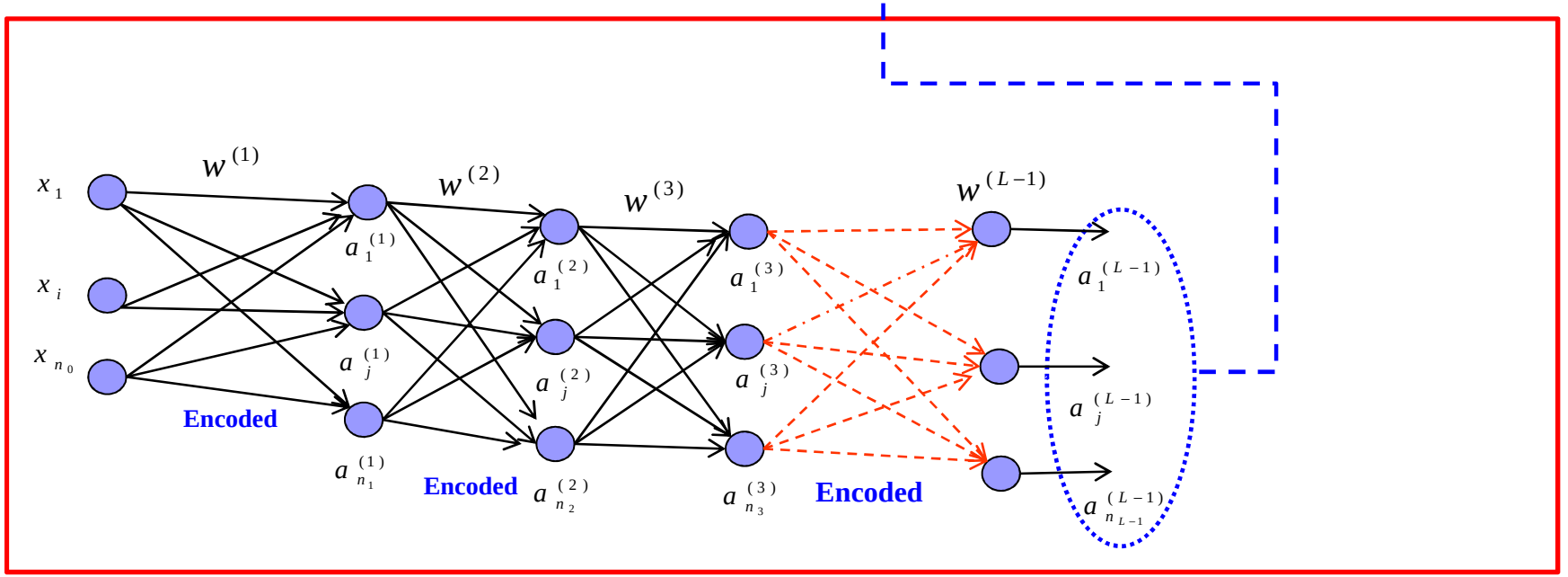


Clasificador Softmax:

Número de Clases: n_L
(nodos salidas)

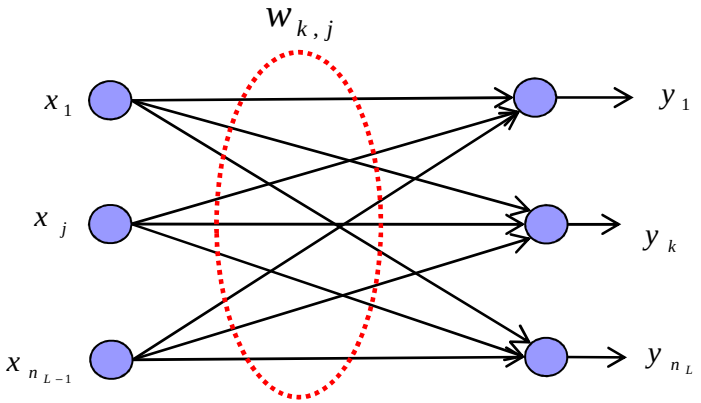


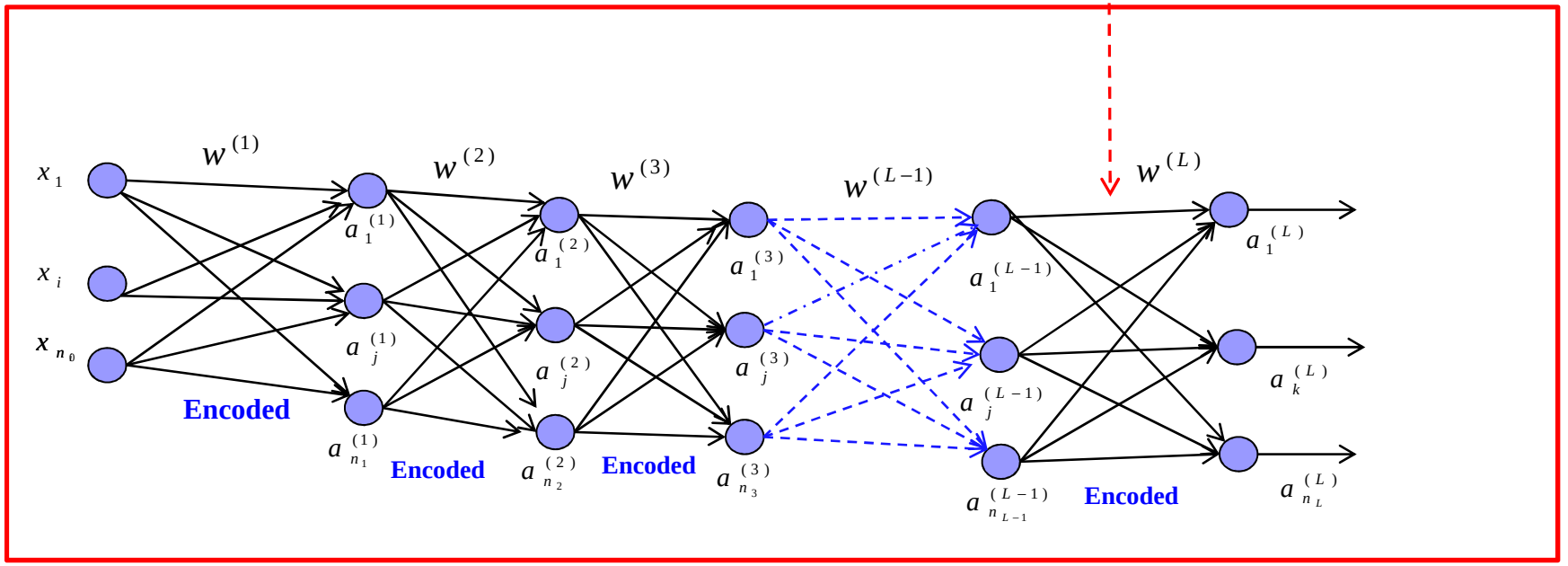
INPUT: $X(n_{L-1}, N)$



Deep Learning+ Clasificador Softmax:

- Entrenar el Softmax
- Retornar los pesos del Softmax
  para diseñar la **L-*ésima*** capa del DL





DEEP LEARNING

CONTINUARÁ….