DEEP LEARNING :

Algoritmo de Aprendizaje:

miniBatch RMSprop

+

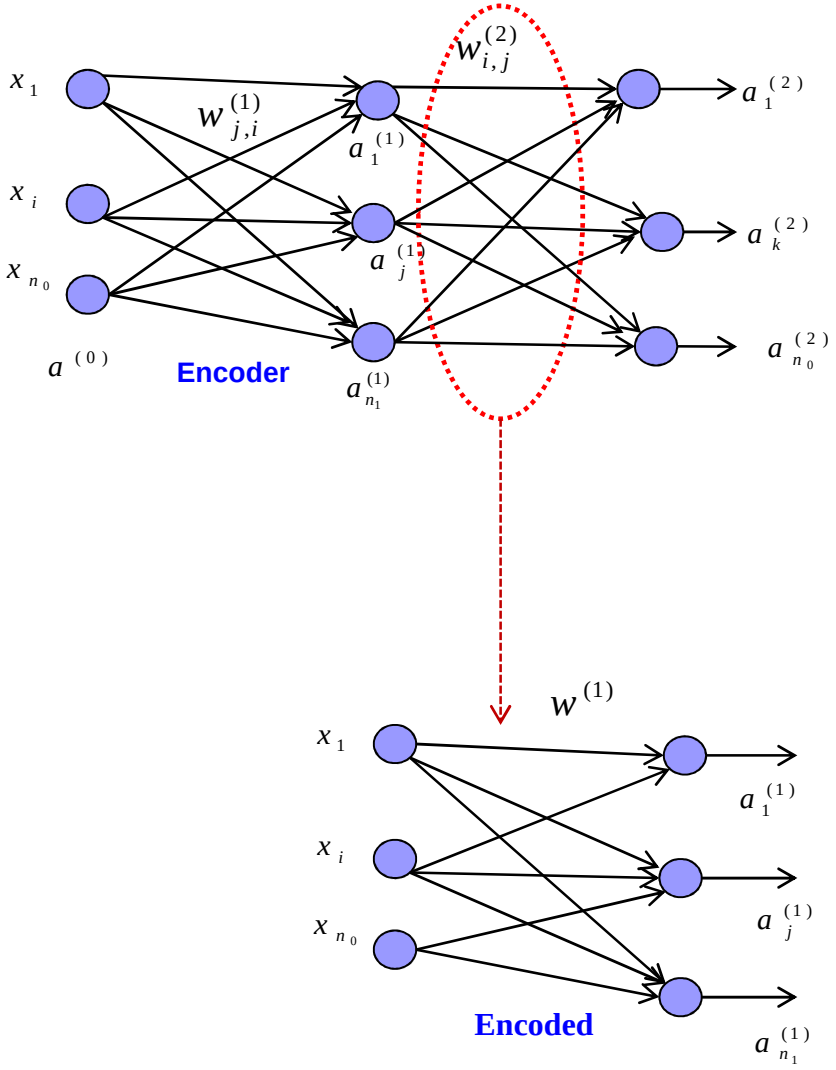
Pseudo-Invesa

Aprendizaje AE: RMSprop+Pseudo-inversa

AE_1 :

INPUT:

$X(n_0, N)$



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

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

g : Función Identidad

Steps:

1. Peso Oculto: RMSprop
2. Pesos Salida: P-inversa
3. Retornar los pesos del Decoder para diseñar las capa del DL

Pesos Ocultos de AE : miniBatch RMSprop

- **M: Tamaño del mini Batch (bloque)**

Costo:

$$E = \frac{1}{2M}\sum_{n=1}^{M} C_n = \frac{1}{2M}\sum_{n=1}^{M}\sum_{k=1}^{n_0}\left(a_{k,n}^{(2)} - a_{k,n}^{(0)}\right)^2$$

Gradiente: Pesos Ocultos

$$gW^{(1)} = \left\{\left(w^{(2)}\right)^T \times \delta^{(2)}\right\} \otimes f'\left(z^{(1)}\right) \times \left(a^{(0)}\right)^T$$

$$\delta^{(2)} = e \otimes g'\left(z^{(2)}\right)$$

Pesos Ocultos de AE : miniBatch RMSprop

Actualización: Pesos Ocultos

$$w^{(1)}(t_k) = w^{(1)}(t_{k-1}) - gRMS \quad , \quad t_k = 1,2,\ldots,T$$

$$v^{(1)} = \beta \times v^{(1)} + (1-\beta) \times \left(gW^{(1)}\right)^2$$

$$gRMS = \frac{\mu}{\sqrt{v^{(1)} + \varepsilon}} \otimes gW^{(1)}$$

$$\mu \in (0,1)$$

$$\varepsilon = 10^{-8}$$

$$v^{(1)}(0) = 0$$

$$\beta = 0.9$$

Pesos de Salida de AE :Pseudo-Inversa

Actualización: Pesos Salida

$$w^{(2)} = a^{(0)} \times \left(a^{(1)}\right)^{T} \left(a^{(1)} \times \left(a^{(1)}\right)^{T} + \frac{I}{C} \right)^{-1}$$

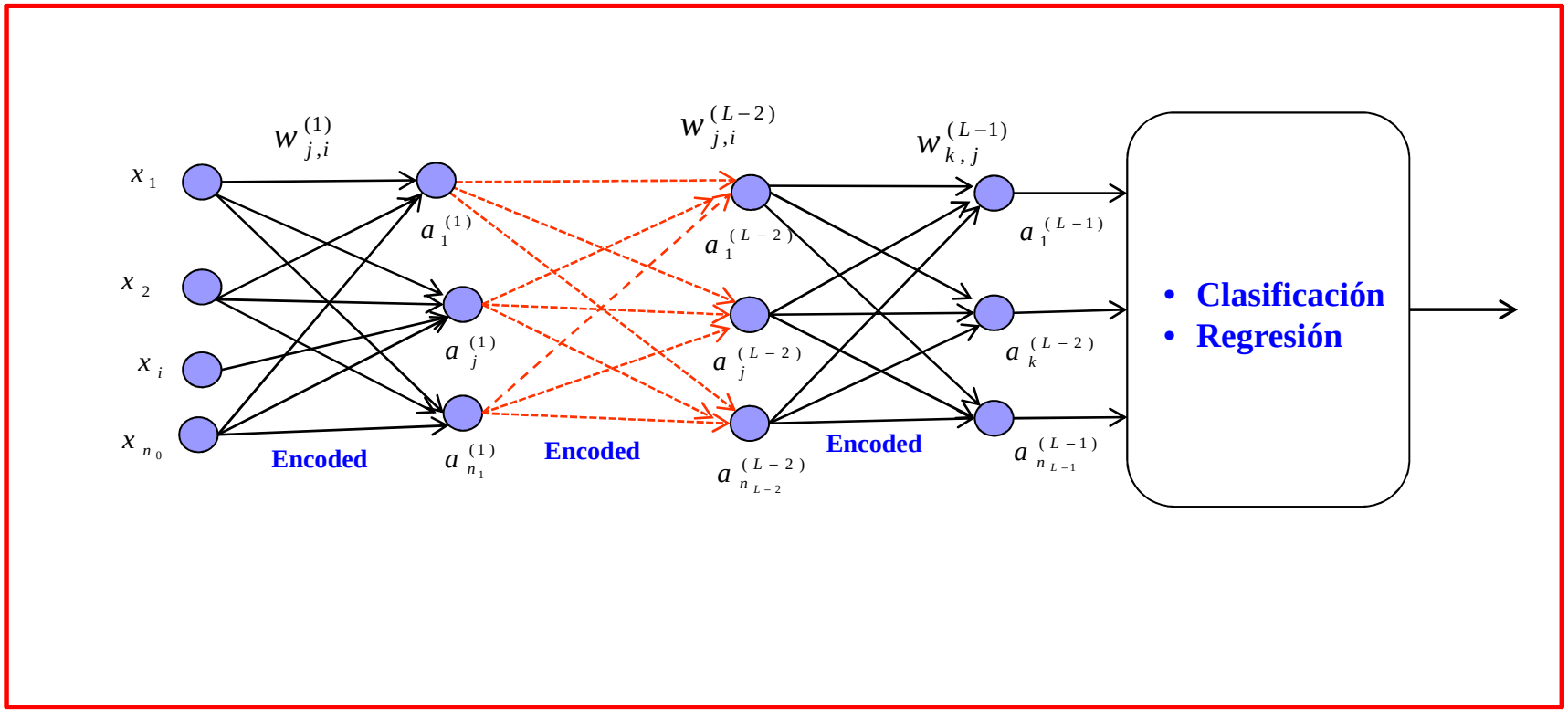
$$C \in \left(10, 10^{10}\right)$$

$$I = matriz \ \ (n_1, n_1)$$

I: Matriz Identidad

C: Penalidad de Pseudo-inversa

Deep Learning : AutoEncoder Apilados



Clasificación Softmax: Feed-forward

Activación Softmax de la *n*-ésima muestra

$$z = w \times x, \quad x \in \Re^{(d \times M)}, w \in \Re^{(m,d)}$$

$$z = \exp(z)$$

$$a_n = \frac{z(:,n)}{\sum_{k=1}^{n_L} z_{k,n}}, \quad n = 1, \ldots, M$$

Aprendizaje Softmax: miniBatch RMSprop

Entropía Cruzada:

$$Cost = -\frac{1}{M}\sum_{n=1}^{M}\sum_{k=1}^{n_L} T_{k,n}\log\left(a_{k,n}\right)$$

- **N** : número de muestras, $\mathbf{n_L}$: numero de clases,
- $\mathbf{T_{k,n}}$: valores deseados (etiquetas), ***lambda***: penalidad de pesos.

Notación Matricial :
Gradiente Pesos Softmax

$$gW = -\frac{1}{M}\left((T - A)\times X^{T}\right)$$

Actualización de Pesos:

$$w\left(t_k\right) = w\left(t_{k-1}\right) - gRMS \quad , \quad t_k = 1,2,\ldots,T$$

$$v = \beta \times v + (1-\beta)\times\left(gW\right)^2$$

$$gRMS = \frac{\mu}{\sqrt{v+\varepsilon}} \otimes gW$$

$$\mu \in (0,1), \varepsilon = 10^{-8}, v(0) = 0, \beta = 0.9$$

TAREA 3 : Deep Learning

miniBatch RMSprop

+

Pseudo-Inversa

OBJETIVO

- Implementar y evaluar el rendimiento de un modelo Aprendizaje Profundo (DL) usando algoritmo RMSprop combinado con el método de la Pseudo-inversa para clasificación una muestra de datos en diez diferentes clases.

DATA : Train

- Formato: train**.csv** : (D,N), donde:
  - **Las primeras (D-1)-filas :**
    - Representan los atributos del problema.
  - **La última fila :**
    - Representa las etiquetas numérica de la clase.
    - Las etiquetas son valores enteros ente 1 y 10.
  - **N-columnas** :
    - Representan el número de muestras del problema.

DATA : Test

- Formato: test**.csv** : (D,N), donde:
  - **Las primeras (D-1)-filas :**
    - Representan los atributos del problema.
  - **La última fila :**
    - Representa las etiquetas numérica de la clase.
    - Las etiquetas son valores enteros ente 1 y 10.
  - **N-columnas** :
    - Representan el número de muestras del problema.

FASE 1: Pre-Tuning

- train.py:

Inicialización de pesos con valores aleatorios

$$r = \sqrt{\frac{6}{n_i + n_{i-1}}}$$

$$w^{(i)} = rand\ (n_i, n_{i-1}) \times 2 \times r - r$$

n_i : Nodos capa siguiente
n_{i-1} : Nodo capa previa

FASE 1: Pre-Tuning

- **train.py:**
  - **Convertir las etiquetas numéricas en etiquetas binaria de tamaño 10.**
  - **Archivos de Salida**:
    - costo_softmax.csv. :
      - N-filas por 1-columna
    - Pesos del Deep Learning.
      - w_dl.npz.

FASE 1: Pre-Tuning

- **test.py:**
  - **Convertir las etiquetas numéricas en etiquetas binaria de tamaño 10.**
- **Archivos de Salida:**
  - metrica_dl.csv.
    - F-scores para cada una de las 10 clases.
    - F-score promedio.

Test.py: Métrica:

$$\text{F - score}\ (j) = 2 \times \frac{\text{Pr}\ ecison\ (j) \times \text{Re}\ call\ (j)}{\text{Pr}\ ecision\ (j) + \text{Re}\ call\ (j)}$$

$$\text{Precision}\quad (i) = \frac{CM_{i,i}}{\sum_{j=1}^{n_L} CM_{i,j}}, \quad i = 1, \ldots, n_L = 10$$

$$\text{Re}\ call\ (j) = \frac{CM_{j,j}}{\sum_{i=1}^{n_L} CM_{i,j}}, \quad j = 1, \ldots, n_L = 10$$

$$avgFscore \quad = \frac{1}{10} \sum_{i=1}^{10} Fscore\ (i)$$

CM(i,j) : Matriz de confusión

Configuración: AE-Apilados

- **cnf_sae.csv:**

- Linea 1: Penalidad de P-inversa : 100
- Línea 2: Tamaño del Bloque (batch) :64
- Línea 3: Máximo Iteraciones : 60
- Línea 2: Tasa de aprendizaje : 0.1
- Línea 3: Nodos Oculto AE1 : 200
- Línea 4: Nodos Oculto AE2 : 150
- Línea 5: Nodos Oculto AE3 : 100
- …

Configuración : Softmax

- **cnf_softmax.csv**

- Línea 1: Máximo Iteraciones : 200
- Línea 2: Tasa aprendizaje (mu) : 0.01

ENTREGA

- Lunes 22/Noviembre/2021
  - Hora : 09:00 am
  - Lugar : Aula Virtual del curso

- **Lenguaje Programación:**
  - Python version: 3.7.6 window (anaconda)
    - numpy
    - panda

OBSERVACIÓN:

- Si un Grupo no Cumple con los requerimientos funcionales y no-funcionales, entonces la nota máxima será igual a 3,0 (tres coma cero).

CONTINUARÁ….