



Evaluación Tarea 2

■ Fase 1: Pre-Tuning (40%)

- SAEs: mini-Batch SGD + P-inversa.
- Clasificador Softmax : GD.

■ Fase 2: Fine-Tuning (60%)

- Training: Back-Propagation.



Pre-Proceso: `pre_proc.py`

- Crear Nuevos Datos desde `data.csv`:
 - 1.- `data_input`
 - Tamaño: N-filas por (D-1)-columnas.
 - Contiene las primeras (D-1)-columnas desde archivo original `data.csv`.
 - 2.- `data_label`
 - Tamaño: N-filas por 1-coumna.
 - Contiene la última columna, la cual denota la etiqueta numérica de la clase.



Pre-Proceso: pre_proc.py

- Transformar las etiquetas numérica a etiqueta Binaria.
- Actualizar **data_label** con Etiquetas Binarias.



Parámetros AEs

- Crea archivo: **param_sae.csv**
- Porcentaje de training :
- Penalidad Pseudo-inversa :
- Número Máximo de Épocas :
- Tamaño mini-Batch :
- Tasa Aprendizaje :
- Nodos Ocultos AE1 :
- Nodos Ocultos AE2 :
- Nodos Ocultos AE3 :



Parámetros Softmax

- Crear archivo: **param_softmax.csv**
- Máximo Número Iteraciones :
- Tasa Aprendizaje :
- Penalidad de Pesos (lambda) :



Pre-Tuning: train_main.py

■ Crear Data train y Data Test:

- Número de muestras de training:

- $L = \text{round}(p \times N)$,

- p : porcentaje de training dado en el archivo **param_sae**

- Número de muestras de testing

- $M = L + 1$ hasta N .



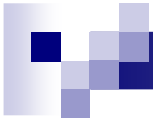
Pre-Tuning: train_main.py

- Re-ordenar aleatoriamente la **data_input** y **data_label**.
- Crear data training con L-muestras.
- Crear data Testing con M-muestras.
- **Generar archivos de Resultados:**
 - **pre_costos.csv (Clasificador Softmax)**



Pre-Tuning: test_main.py

- Generar Archivo Resultado:
 - *pre_fscore.csv*



FINE TUNING : Back-Propagation

Fine Tuning: Pesos de Salida con BP

Capa de Salida del Deep Learning:

$$z_k^{(L)} = w_{k,j}^{(L)} a_j^{(L-1)}$$
$$a_k^{(L)} = \frac{\exp(z_k^{(L)})}{\sum_{i=1}^{nC} \exp(z_i)}$$

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

Función de Costo:

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

Update Pesos de Salida:

$$w^{(L)}(m) = w^{(L)}(m-1) - \mu \frac{\partial E}{\partial w^{(L)}}, \quad m = 1, \dots, MaxIter$$
$$\delta_k^{(L)} = (a_k^{(L)} - T_k) = \delta^{(L)}$$
$$\frac{\partial E}{\partial w^{(L)}} = \frac{1}{N} \left\{ \delta^{(L)} \times (a^{(L-1)})^T \right\}$$



Fine Tuning: Pesos Ocultos con BP

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

**Update Pesos
Ocultos:**

$$w^{(l)}(m) = w^{(l)}(m-1) - \mu \frac{\partial E}{\partial w^{(l)}}, \quad m = 1, \dots, MaxIter$$
$$\delta^{(l)} = \left\{ \left(w^{(l+1)} \right)^T \times \delta^{(l+1)} \right\} \otimes f' \left(z^{(l)} \right)$$
$$\frac{\partial E}{\partial w^{(l)}} = \delta^{(l)} \times \left(a^{(l-1)} \right)^T$$



Parámetros Back-Propagation

- Crear archivo: **fine_bp.csv**
- Máximo Número Iteraciones :
- Tasa Aprendizaje :



Fine-Tuning: `train_fine.py`

- Leer Data de Training
- Leer pesos pre-training
- Training: **Back-propagation**
- Generar Resultados:
 - **`fine_costo.csv` (Clasificador Softmax)**



Fine-Tuning: test_fine.py

- Leer Data de Testing
- Leer pesos obtenidos con Back-propagation
- **Generar Resultados:**
 - **`fine_fscore.csv`**



ENTREGA

- Miércoles 17/Junio/2020, Hora: 10:30 am
- Lugar: Aula Virtual del curso

- **Programas fuentes:**
 - ☐ pre_proc.py, train_main.py, test_main.py
 - ☐ train_fine.py, test_fine.py
- **Archivos Resultados:**
 - ☐ Pre-Tuning: Costos y F-scores
 - ☐ Fine-tuning : Costos y F-scores
- **Manual de Usuario:**



Lenguaje Implementación: PYTHON v. 3.6.7 window