# Aprendizaje Automático: Trabajo práctico 0

PAttern Recongition and MAchine Learning Group (PARMA-Group)

22 de febrero de 2022

**Fecha de entrega:** Domingo 13 de Marzo del 2021.

**Entrega**: Un archivo .zip con el código fuente LaTeX o Lyx, el pdf, y un jupyter en Pytorch, debidamente documentado, con una función definida por ejercicio. A través del TEC-digital.

## 1. Implementación del algoritmo K-vecinos mas cercanos

El algoritmo de K-vecinos mas cercanos es un algoritmo de aprendizaje automatico supervisado muy popular por su simplicidad. Dado un conjunto de datos representado matricialmente en la matrix $X_{\text{train}} \in \mathbb{R}^{N \times D}$ y un arreglo de etiquetas $\overrightarrow{t} \in \mathbb{R}^{N}$:

$$X_{\text{train}} = \begin{bmatrix} - & \overrightarrow{x}_1 & - \\ & \vdots & \\ - & \overrightarrow{x}_{N_{\text{train}}} & - \end{bmatrix} \qquad \overrightarrow{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_{N_{\text{train}}} \end{bmatrix}$$

Para cada dato $\overrightarrow{x}_i^{(\text{test})} \in X_{\text{test}}$ en un conjunto de datos de prueba o evaluacion $X_{\text{test}} \in \mathbb{R}^{N_{\text{test}} \times D}$:

$$X_{\text{test}} = \begin{bmatrix} - & \overrightarrow{x}_1 & - \\ & \vdots & \\ - & \overrightarrow{x}_N & - \end{bmatrix}$$

se crea un conjunto de datos $X_{\text{KNN}}$ con los $K$ vecinos mas cercanos de la observacion $\overrightarrow{x}_j$ en el conjunto de datos $X_{\text{train}}$, donde cada observacion $\overrightarrow{x}_i \in X_{\text{KNN}}$ cumple que:

$$X_{\text{KNN}} = \arg \min_{K \text{ min } j} \left( d\left( \overrightarrow{x}_i^{(\text{test})} - \overrightarrow{x}_j \right) \right)$$

Luego de tomar los K vecinos mas cercanos de la observacion $\overrightarrow{x}_i^{(\text{test})}$ se realiza una votacion segun las etiquetas correspondientes $t_i^{(\text{test})}$, y se toma como estimacion de la etiqueta $\widetilde{t}_j$ la etiqueta mas votada.

1. **(50 puntos)** Implemente el algoritmo de K-vecinos mas cercanos con la posibilidad de usar la distancia euclidiana y la de Manhattan en la funcion $d\left(\overrightarrow{x}_i - \overrightarrow{x}_j\right)$.

   *a*) Realice la implementacion de forma completamente matricial, para cada observacion $\overrightarrow{x}_i^{(\text{test})}$ *evaluate_k_nearest_neighbors_observation(data_training, labels_training, test_observation, K = 7, is_euclidian = True)* (**No use ciclos *for***).

      1) Para ello use funcionalidades de *pytorch* como repeat, mode, sort, etc.
      2) is_euclidian indica si se usara la distancia Euclidiana o la de Manhattan de lo contrario. $K$ corresponde a la cantidad de vecinos a evaluar

   *b*) **(10 puntos)** Para todo el conjunto de datos $X_{\text{test}}$ implemente la funcion *evaluate_k_nearest_neighbors_test_dataset(data_training, labels_training, test_dataset, K = 3, is_euclidian = True)*, la cual utilice la funcion previamente construida *evaluate_k_nearest_neighbors_observation* para calcular el arreglo de estimaciones $\overrightarrow{\widetilde{t}}$ para todos los datos en $X_{\text{test}}$.

   *c*) **(10 puntos)** Implemente la funcion *calcular_tasa_aciertos* la cual tome un arreglo de estimaciones $\overrightarrow{\widetilde{t}}$ y un arreglo de etiquetas $\overrightarrow{x}_i^{(\text{test})}$ y calcule la tasa de aciertos definida como $\frac{c}{N}$ donde $c$ es la cantidad de estimaciones correctas. **(No use ciclos *for*)**.

2. **(5 puntos)** Para un conjunto de datos de $N = 4000$ (2000 observaciones por clase) genere un conjunto de datos con medias $\mu_1 = [12, 12]^T$, $\mu_2 = [15, 15]^T$, y desviaciones estandar $\sigma_1 = [3, 3]^T$, $\sigma_2 = [2, 2]^T$. Grafique los datos y muestre las figuras.

3. Compruebe y compare para las dos distancias implementadas, usando el dataset anterior, y $K = 7$:

   *a*) **(5 puntos)** La tasa de aciertos, definida como $\frac{c}{N}$ donde $c$ es la cantidad de estimaciones correctas, usando el mismo conjunto de datos $X_{\text{train}}$ como conjunto de prueba $X_{\text{test}}$. Documente los resultados y

comentelos. Puede probar otros valores de medias que faciliten la separabilidad de los datos para facilitar la explicacion.

*b*) **(20 puntos)** Usando las funciones de particion de datos del paquete *sklearn* necesarias, implemente la particion de datos del conjunto de datos original $X$ para crear las particiones $X_{\text{train}}$ y $X_{\text{test}}$. Cree 10 particiones distintas, para ejecutar 10 veces el algoritmo. Use $N = 4000$ (2000 observaciones por clase).

1) Utilice 70 % de los datos para entrenamiento y el resto para prueba.
2) Calcule la tasa de aciertos para las 2 configuraciones (distancia $\ell_1$ y $\ell_2$ ) probadas, usando $X_{\text{train}}$ para entrenamiento y $X_{\text{test}}$ para prueba. Reporte los resultados en una tabla, junto con su media y desviacion estandar y comentelos.
3) Calcule ademas el tiempo de ejecucion por corrida para cada configuracion. Reporte los resultados en una tabla, junto con su media y desviacion estandar y comentelos. Cual distancia resulto mas eficiente? Hay algun costo en cuanto a la tasa de aciertos? Explique el porque de la diferencia en la tasa de aciertos, si la hay.