# IMT2112
# Tarea 1

Elwin van 't Wout

August 19, 2020

## Introducción

En muchas situaciones se puede encontrar grupos de registros similares en grandes volúmenes de datos. Encontrar estas *conglomeraciones* también está conocido como *clustering* en inglés. Hay una variedad de algoritmos para este tarea de aprendizaje automatizado no-supervisado, de lo cual $k$-means es uno de los mas comunes.
Adjunto hay una presentación con más información. Lo mas importante para esta tarea es el algoritmo mismo. El objetivo de $k$-means es encontrar $k$ conglomeraciones en la base de datos. Un algoritmo recursivo puede encontrar estas conglomeraciones en base a las distancias entre los registros.
El algoritmo se puede describir como lo siguiente.

```
1. Importar o generar n registros de m dimensiones cada uno.
2. Inicializar el algoritmo.
   a. Elegir el valor de k.
   b. Elegir k centros iniciales en el espacio.
3. Algoritmo recursivo para mejorar conglomeraciones.
   a. Asignar a cada registro la etiqueta del centro mas cercano.
   b. Calcular los nuevos centros como el centro de masa de las conglomeraciones.
   c. Iterar hasta las conglomeraciones ya no cambien.
```

## Tarea

En esta tarea hay que programar el algoritmo de $k$-means con multiprocessing.

1. Generan los datos para la conglomeración. La dimensión de los registros debe ser a menos dos y el número de registros a menos un mil. Para generar los datos aleatoriamente, se puede usar `numpy.random` o `sklearn.datasets.make_blobs`.

2. Inicilizen el algoritmo, eligiendo el valor de $k$, lo cual debe ser a menos tres.

3. Implementen el algoritmo de $k$-means en las siguientes formas. Para este, **no** se puede usar `sklearn`.

   (a) Python loops. Todos las iteraciones de $k$-means y el cálculo de distancias, sumas, vectores, etc. deben ser programados con bucles en Python. Así, Python va a usar solo un hilo.

   (b) NumPy. Vectorizen el algoritmo, es decir, usen funciones vectoriales de `NumPy`, tales como normas, distancias y sumas de arreglos.

   (c) Multiprocessing. Usen `multiprocessing` para paralelizar el código de Python. Es decir, paralelizen todos los bucles sobre los registros con `multiprocessing`.

4. Miden el tiempo de cómputo de cada implementación, para distintos tamaños de la base de datos. Observen cuantos procesos y cuantos hilos Python está usando en tu computador.

5. Finalmente, visualizen las conglomeraciones en un *scatter plot*. Si usaron mas dimensiones que dos, hay que reducir la dimensionalidad (se puede usar `sklearn` para este).

## Evaluación

Entreguen el código de Python en una mapa comprimida a través de Canvas.

Los reglamientos del curso se puede encontrar en Canvas.