IMT2112 Semestre 2024-2
Tarea 1

Elwin van 't Wout

10 de agosto de 2024

Introducción

Un desafío común de aprendizaje automatizado no supervisado es buscar grupos de registros similares en grandes volúmenes de datos, llamado *conglomeraciones* o *clústers* (no hay que confundir con un 'clúster' de computadores). En esta tarea van a programar el algoritmo de k-medias. Este algoritmo contempla los pasos siguientes.

```
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 encontrar 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.
4. Resultado: cada registro tiene una etiqueta que corresponde a
   la conglomeración final.
```

Acá, el centro de masa de un grupo de registros $\mathbf{x}_i \in \mathbb{R}^d$ con $i = 1, 2, 3, \ldots, m$ está dado por $\mathbf{c} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$.

Tarea

Esta tarea contempla una implementación paralela del algoritmo k-medias.

1. En Canvas pueden encontrar una implementación de k-medias en base a funcionalidad estándar de Python.

   a) Corren el código disponible en Canvas.

   b) Se puede cambiar la cantidad de registros, la cantidad de grupos generados y el valor de k si quieren. En esta tarea, deben usar al menos un mil de registros y un valor de k de al menos cinco.

2. Vectorizen el algoritmo, es decir, usen arreglos en lugar de listas y funciones vectoriales de `NumPy` o `SciPy`, tales como normas, distancias y sumas de vectores. Al final, no deben tener ciclos (tampoco *list comprehension*) sobre los registros de la base de datos. Se puede usar ciclos sobre los centros. Revisen bien si el resultado es consistente con la implementación anterior. Deben usar funcionalidad de `NumPy` o `SciPy`, no pueden usar `sklearn`, `pandas` u otras librerías.
   Sugerencia: la función `scipy.spatial.distance.cdist` podría ser util.

3. Miden el tiempo de cómputo de cada implementación. Asegúrense que la base de datos es suficientemente grande para ver diferencias en el tiempo de cómputo. Observen cuantos procesos y cuantos hilos Python está usando en tu computador con el *administrador de tareas*, *activity monitor*, *top* u otro programa para este propósito.

4. En general, es un desafío encontrar el mejor valor de k para una base de datos. Lo más común es correr el algoritmo para varios valores de k y buscar el punto de inflexión donde la dispersión ya no reduce mucho.

   a) Dados los resultados de k-medias, calculen la dispersión

   $$D_k = \sum_{i=1}^{k} \sum_{j=1}^{N_i} ||\mathbf{x}_{ij} - \mathbf{c}_i||$$

   donde $\mathbf{c}_i$ y $\mathbf{x}_{ij}$ son el centro y registro j de conglomeración i. La dispersión es una medida de la precisión o ajuste de las conglomeraciones a los datos. Pueden usar `NumPy` para este cálculo.

   b) Corren el algoritmo de k-medias para $k = 1, 2, 3, \ldots, 25$ y visualizen D_k en un plot.

c) Utilizen la librería `joblib` para paralelizar estas tareas bajo el paradigma *multiprocessing*. Cabe señalar que cada tarea consta de correr todo el algoritmo de k-medias para un único valor de k. Revisen bien si `joblib` efectivamente usa distintos procesos.

d) Miden el tiempo de cómputo y cambien el número de procesos que se asigna a `joblib`.

Escriben un informe corto en lo cual discuten los resultados. Puede ser en formato PDF o en el mismo Jupyter Notebook. Responden al menos las preguntas siguientes.

- Expliquen si el código disponible en Canvas paraleliza alguna parte del algoritmo.
- ¿Cuál es la diferencia en tiempo de cómputo entre las dos variantes de k-medias en preguntas 1 y 2? Expliquen por qué las implementaciones con ciclos y NumPy son diferentes.
- ¿Cuál es la diferencia en tiempo de cómputo con distintos números de procesos en `joblib`? Expliquen por qué son diferentes. Responde este usando
  - k-medias implementada con ciclos,
  - k-medias implementada con NumPy

  como tarea dentro de `joblib`.
- ¿Cuantos núcleos físicos tiene tu computador?

Observaciones

No se puede compartir código entre compañeros, tampoco usar código de fuentes externos salvo la página del curso en Canvas.

Evaluación

Entreguen todo el código de Python en un Jupyter notebook y el informe (pdf separado o en el Jupyter notebook) a través de Canvas. Si son múltiples archivos, entreguen una mapa comprimida.

Deben hacer un `Restart Kernel and Run all Cells` antes de subir, es decir, deben entregar el Jupyter notebook con el output y las celdas deben ser ejecutadas en orden.

Los reglamientos del curso se puede encontrar en Canvas. Se destaca que las tareas deben ser hechas de forma individual.

Sugerencias

La tarea se puede preparar en un computador personal o en Google Colab (https://colab.research.google.com). Para esta tarea se requiere una versión reciente de Python y algunas librerías estándares. La librería `joblib` se puede instalar con `conda install joblib` o `pip install joblib`.

Se recomienda crear un nuevo *environment* de Anaconda para esta tarea con el objetivo de evitar problemas de distintas versiones the Python y librerías. Ver https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html o https://docs.anaconda.com/anaconda/navigator/tutorials/manage-environments/.

Tengan cuidado copiando variables, especialmente dentro de una iteración. Por ejemplo, si usen `old_centers = new_centers`, la variable `old_centers` es una referencia a la variable `new_centers`. Por lo tanto, cambiar un elemento de `old_centers` también cambia `new_centers` y vice versa. Para crear una copia, uno debe usar `old_centers = copy.deepcopy(new_centers)` en Python.