IMT2112 2022
Tarea 1

Elwin van 't Wout

18 de agosto de 2022

Introducción

Una tarea 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 programamos el algoritmo de k-medias. Este algoritmo se puede describir con 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 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

Este tarea contempla distintas implementaciones del algoritmo k-medias.

1. Generen los datos para la conglomeración. La dimensión de los puntos debe ser a menos dos (2D) y el número de registros a menos un mil. Para generar los datos aleatoriamente, se puede usar, por ejemplo, `numpy.random` o `sklearn.datasets.make_blobs`.

2. Inicializen el algoritmo, eligiendo el valor de k (lo cual debe ser a menos tres) y los centros iniciales.

3. Implementen el algoritmo de k-medias en las siguientes variantes. Para esta parte, **no** se puede usar `sklearn`.

   a) Python loops. Todos las iteraciones de k-medias y el cálculo de distancias, sumas, vectores, etc. deben ser programados con bucles en Python. En específico, la asignación (paso 3a del algoritmo) debe ser dos bucles sobre los registros y los centros; y el update (paso 3b) un bucle sobre los registros. No pueden usar funciones de `Numpy` sobre vectores, solo sobre un único número, por ejemplo, `np.sqrt(x)` para `x` un número y no un arreglo.

   b) NumPy. Vectorizen el algoritmo, es decir, usen funciones vectoriales de `NumPy` o `SciPy`, tales como normas, distancias y sumas de vectores. Al final, no deben tener bucles (tampoco *list comprehension*) sobre los registros de la base de datos. Se puede usar bucles sobre los centros.
   Sugerencia: la función `scipy.spatial.distance.cdist` podría ser util.

   c) Multiprocessing. Usen la librería `joblib` para paralelizar el código de Python en distintos procesos (*multiprocessing*, no *multithreading*). En esta tarea, es suficiente paralelizar la asignación (paso 3a) sobre los registros de la base de datos. Para el update (paso 3b) se puede usar NumPy.

4. 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. Escriben un informe corto en lo cual explican los resultados. Puede ser en formato PDF o en un Jupyter Notebook. En específico, responden las preguntas siguientes.

   - ¿Cuál es la diferencia en tiempo de cómputo entre las tres variantes? Expliquen por qué son diferentes.
   - ¿Cuál es la diferencia en tiempo de cómputo con distintos números de procesos en `joblib`? Expliquen por qué son diferentes.

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).
   Sugerencia: se puede hacer algo como
   `plt.scatter(dataset[:,0], dataset[:,1], c=cluster)` para dibujar conglomeraciones.

Extensiones

Para los estudiantes motivados, pueden implementar el update (paso 3b) con multiprocessing y comparar el rendimiento con `sklearn`. (Estas sugerencias son opcionales y no se evaluarán.)

Evaluación

Entreguen todo el código de Python (script y/o Jupyter notebook) y el informe (pdf o Jupyter notebook) en una mapa comprimida a través de Canvas.

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 este tarea se requiere una versión reciente de Python y las librerías `numpy`, `scipy` y opcionalmente `sklearn`. La librería `joblib` se puede instalar con `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, por ejemplo,
https://towardsdatascience.com/a-guide-to-conda-environments-bc6180fc533,
https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html,
o https://docs.anaconda.com/anaconda/navigator/tutorials/manage-environments/.