Lab 3 - Parallelizing k-means, Stat 215A, Fall 2024

These are the **lab3-specific** instructions. Please also see the general lab instructions in **lab-instructions.pdf**.

Contents

1 Submission

Push a folder called `lab3` to your `stat-215-a GitHub` repository by **23:59 on Friday, October 18th**. I will run a script that will pull from each of your GitHub repositories promptly at midnight so take care not to be late as late labs will not be accepted.

Follow the general lab instructions in stat-215-a-gsi/disc/week1/lab-instructions.pdf for more details. Please do not try to make your lab fit the requirements at the last minute!

I have provided a template (both `.tex` and `.ipynb`) as a guideline for the writeup. Since the template is intended to make grading easier, please do not deviate from it significantly without good reason. In your `lab3` folder, please provide everything (except the `lingBinary` data) that is needed for someone else to be able to compile your report and receive the exact same pdf.

Special coding instructions

In this lab, you will graded more carefully than usual on your code. As always, please follow a style guide like PEP-8 or the Google Python style guide. Make sure your variable names are meaningful, write comments liberally, and document your functions. Please also provide all scripts necessary to reproduce your results, includes any .py or .sh scripts used to run your code on the SCF, as well as the `S.csv` result file. Pay very close attention to the instructions below about `similarity.py`.

2 Academic honesty

Academic honesty statement

You can use your statement from lab1 or lab2.

Collaboration policy

You are welcome to discuss ideas with me or other students, but your report must be written up and completed individually. If you do consult with other students, you must acknowledge these students in your lab report.

LLM usage policy

You are allowed to use LLMs (ChatGPT, GitHub Copilot, etc.) to help with the **coding** for this lab. If you use an LLM to help with coding, you must do the following:

- At the end of the report in the appropriate section (see template), state which tool(s) you employed and for what parts of the coding.
- Describe the nature of your interactions with the LLM. e.g. did you ask ChatGPT to code things in English? Did you use Copilot by writing a comment then tab-completing individual lines? Did you get Copilot to write whole functions? Did you mostly use LLMs for debugging? For writing comments? etc.
- If you used a prompt-based LLM like ChatGPT, add a link to your ChatGPT session, e.g. `https://chatgpt.com/share/f7a9094a-9eda-47a0-9103-4aa6345284b0`.
- Describe your experience with the process. Is it a good experience? Did it save you time? Did the code end up better or worse than if you had written it yourself? Do you fully understand the code that these LLMs produce? Do you want to use it again for similar projects?

You are also allowed to use an LLM for polishing the writing of your report at the sentence or paragraph level. As an example, you **may**, for example, plug in a sentence or paragraph to ChatGPT and ask it to be checked for grammar and general sentence flow. You **must not**, for example, ask ChatGPT to expand some bullet points into a paragraph, or have it completely rewrite a paragraph, or plug in your whole report, or make something "sound better," etc. The report should still be in your "voice" and not in the (very obvious) ChatGPT voice. And you **must not** submit any writing directly by an LLM without reviewing/editing it yourself. Please ask me if you are unsure if something is allowed.

Any LLM usage for writing must be reported in the LLM usage section of your report—e.g. "I used ChatGPT to revise the grammar of section 2.1." It must be clear to me how much you used LLM tools.

3 What you need to do

We will be investigating the stability of k-means using a popular procedure outlined in Ben-Hur et al. which uses stability as a guide for picking k. The procedure is outlined in Algorithm 1. You should consult the paper for more details, particularly consulting the similarity measures you can use.

In this lab, you will be implementing this method on the binary-coded linguistic data from Lab 2. Please use the `LingBinary.Rdata` file included in this lab to ensure consistency (note: only use its binary columns!). Set the maximum number of clusters to consider: $k_{\max} = 10$; the number of repeated iterations: $B = 100$; and the sampling proportion, m as large as you can get it while also running in a reasonable time (m should be no less than 0.2, no more than 0.8). This will require a decent amount of computation, so you should try to run this in parallel on the SCF cluster

Algorithm 1 Calculation of clustering similarities in k-means

Input: X, $k_{\max}$, B
Output: S
for $k = 2$ to $k_{\max}$ **do**
  for $i = 1$ to B **do**
    $\text{sub}_1 \leftarrow \text{subsample}(\text{X}, \text{m})$, a subsample of fraction m of dataset X
    $\text{sub}_2 \leftarrow \text{subsample}(\text{X}, \text{m})$, a subsample of fraction m of dataset X
    $L_1 \leftarrow \text{cluster}(\text{sub}_1, k)$
    $L_2 \leftarrow \text{cluster}(\text{sub}_1, k)$
    $\text{intersect} \leftarrow \text{sub}_1 \cap \text{sub}_2$
    $S_{i,k} \leftarrow \text{similarity}(L_1(\text{intersect}), L_2(\text{intersect}))$
  end for
end for
return S

Similarity implementation

First, choose a similarity measure to use, and explain why you chose it.

To compute the similarity of clusterings, you will (in theory) be dealing with a $q \times q$ matrix, where q is the number of data points that are common to each subsample (if $m = 0.8$, q will be approximately 29000, meaning that the similarity matrix, C, will be a 6GB matrix). This could be prohibitively large. For the following questions, you can use any similarity measure mentioned in the paper: correlation, Jaccard, matching.

Write a Python function called `similarity` to calculate the similarity between two membership vectors. At **bare minimum**, the function should complete in a reasonable time for inputs of size 5000. Write this function in a file called `similarity.py` in the `code` folder. This is how your function will be evaluated:

1. Your function should take two membership vectors as input and return a single number (a scalar) as output.
2. Of course, your function must return the correct similarity value for the given membership vectors.
3. Can you avoid storing the entire $q \times q$ matrix in memory?
4. Is your algorithm efficient? The most obvious way to calculate similarity is $O(q^2)$, but you can do better!
5. Did you try to speed-up the function by compiling it with Numba or Cython?
6. I will benchmark your function on my own machine on randomly generated membership vectors with $k = 10$, first with $q = 5000$ and then with $q = 29000$, and compare its runtime with a reference implementation. I have written an $O(q^2)$ implementation of Jaccard similarity which (on my machine) runs in 0.32s for $q = 29000$. I have also written a smarter implementation which (on my machine) runs in 58μs (5.8×10^{-5} seconds) for $q = 29000$. Note that any speed differences between my machine and yours are probably less than an order of magnitude, so on your laptop (or SCF or whatever) you should be able to get it down to 1ms or so. Also note that speeds roughly this good can be achieved for any of the similarity measures, not just Jaccard. I'll be looking at the order of magnitude of your implementation's runtime. I have provided a `benchmark.py` script in the `code` folder which you can use to benchmark your function in the same (or similar) way as I will when grading.

Discuss your algorithm for the similarity function. Did you use any clever algorithmic tricks to speed up the computation? If you were able to avoid storing the $q \times q$ matrix in memory, how did you do it? Were you able to make the algorithm faster than $O(q^2)$? If so, what is an O-notation expression for its runtime?

Discuss your Python implementation—how did you get the runtime down?

Discuss the timing of your implementation of the similarity function when applied to the `lingBinary.Rdata` for various choices of m. A figure is very good for this! If you kept some different versions of your similarity function, it would be especially cool to see a comparison of different implementations (e.g. Numba vs Cython vs pure Python, or different algorithms) for different values of m.

PS: Don't worry if you aren't able to get the runtime down super low! Making it better than $O(q^2)$ in particular is pretty tricky. But if you get it super low you will get a very good grade in this portion of the lab.

Parallel implementation of Algorithm 1

Implement Algorithm 1, using your fastest implementation of the similarity function to compute the similarity. You can use the k-means implementation of some Python library.

To speed up computation, parallelize using `joblib` (or something else if you would like). You can choose whether to parallelize the outer or inner for loop—will one work better? Run your job with the `lingBinary.Rdata` on the SCF cluster using `sbatch`. You must save the resulting S matrix (or dataframe) as a .csv file in a folder named `results/`.

See the following reference for detailed instruction on using the cluster: `http://statistics.berkeley.edu/computing/servers/`. If you do not have an SCF account, you will need to request one by going to `https://scf.berkeley.edu/account`

Here is what you should document in this section:

1. How did you choose the parameters for k-means (other than k)? (the library you used probably has a bunch of parameters in its implementation of k-means).
2. Did you parallelize the inner or outer for loop? Why?
3. How long did it take for your code to run on the SCF cluster?
4. How many cores did you use? Why?

Note: You should **not** run Algorithm 1 in your `lab1.ipynb` (if you are using a notebook for your report). Good lab submissions will have a separate Python script for Algorithm 1.

Result

Make a plot similar to Figure 3 in Ben-Hur et al. Discuss the plot. What would you pick as k for this dataset and why? Do you trust this method? Why or why not? Do you have any ideas for a different method to choose k?

You should discuss these points in detail—certainly more than a few sentences.

Note: You can and should make a much more attractive figure than their Figure 3.