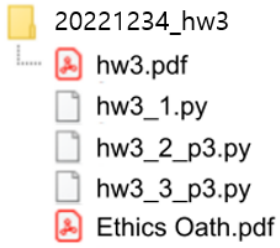
EE412 Foundation of Big Data Analytics, Fall 2022
HW3

Due date: 11/25/2022 (11:59pm)

Submission instructions: Use KAIST KLMS to submit your homework. Your submission should be a single gzipped tar file with the file name `YourStudentID_hw3.tar.gz`. For example, if your student ID is 20221234, the file name should be `20221234_hw3.tar.gz`. You can also use these extensions: tar, gz, zip, tar.zip. Do not use other options besides the ones mentioned above.

For the programming assignment except for problem 1, you are free to use either python 2.x or 3.x. Make sure to add p2 or p3 to the end of the file names. For example, if you used python 3, the file names should be `hw3_2_p3.py`, and `hw3_3_p3.py`

Your zip file should contain a total five files; one PDF file for writeup answers (`hw3.pdf`), three python files and the Ethics Oath pdf file. Before zipping your files, please make a directory named `YourStudentID_hw3` and put your files in the directory. Then, please compress the directory to make a zipped file. Do not use any Korean letters in file names or directory names.



If you do not follow this format, we will **deduct 1 point** of the total score per mistake.

Submitting writeup: Write down the solutions to the homework questions in a single PDF file. You can use the following template. If you use Python in the exercises, **make sure you attach your code to your document**(`hw3.pdf`). Please write as succinctly as possible.

Submitting code: Each problem is accompanied by a programming section. Put all the code for each question into a single file. Good coding style (including comments) will be one criterion for grading. Please make sure your code is well structured and has descriptive comments.

We will **deduct** 1 point per error type (i.e., printing extra lines, format error, and etc)

Ethics Oath: For every homework submission, please fill out and submit the **PDF** version of this document that pledges your honor that you did not violate any ethics rules required by this course and KAIST. You can either scan a printed version into a PDF file or make the Word document into a PDF file after filling it out. Please sign on the document and submit it along with your other files.

Discussions with other students are permitted and encouraged. However, for writing down the solutions, such discussions (except with course staff members) are no longer acceptable: you must write down your own solutions independently. If you received any help, you must specify on the top of your written homework any individuals from whom you received help, and the nature of the help that you received. *Do not, under any circumstances, copy another person's solution.* We check all submissions for plagiarism and take any violations seriously.

1 Link Analysis (40 points)

(a) [20 pts] Solve the following problems, which are based on the exercises in the Mining of Massive Datasets 3rd edition (MMDS) textbook.

- Exercise 5.1.2 (10 points)
  Compute the PageRank of each page in Fig 5.7, assuming $\beta = 0.8$.
  You can use programs for simple calculations. If you use any code to solve the problem, please attach your code with a brief explanation.

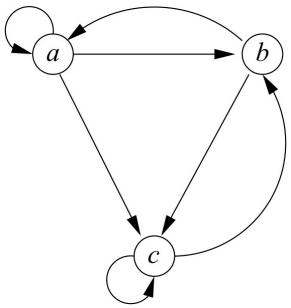


Figure 5.7: An example graph for exercises

- Exercise 5.3.1 (10 points)
  Compute the topic-sensitive PageRank for the graph of Fig 5.15, assuming $\beta = 0.8$ and the teleport set is:

  (a) A only
  (b) A and C

  You can use programs for simple calculations. If you use any code to solve the problem, please attach your code with a brief explanation.

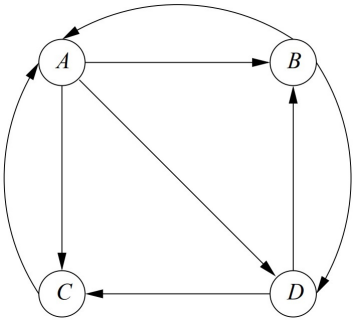


Figure 5.15: Repeat of example Web graph

(b) [20 pts] Implement the PageRank algorithm using Spark.

Implement the PageRank algorithm described in MMDS Chapter 5.2.2 using Spark.

You can download the dataset from the link below:
http://www.di.kaist.ac.kr/~swhang/ee412/graph.txt[1]

The graph is randomly generated and has 1000 nodes and about 8000 edges with no dead ends. For each row, the left page id represents the source and the right id represents the destination. If there are duplicate edges from one page to another, treat them as the same.

You may set $\beta = 0.9$ and start from the vector $\mathbf{v}$ initialized as all 1's divided by the number of pages. You do not have to break the transition matrix M into stripes. Run your algorithm for 50 iterations to produce the final vector $\mathbf{v}$ and return the ids of the top-10 pages with the highest PageRank scores.

Please use **command-line** arguments to obtain the file path of the dataset. (Do not fix the path in your code.) For example, run:

```
bin/spark-submit hw3_1.py path/to/graph.txt
```

Your code should `print` the top-10 page ids with the highest PageRank scores in $\mathbf{v}$. Print up to **5 decimal digits** for the PageRank scores. The output format is the following:

```
<PAGE_ID_0><TAB><SCORE_0>
<PAGE_ID_1><TAB><SCORE_1>
...
<PAGE_ID_9><TAB><SCORE_9>
```

The output should be 10 lines and sorted in **descending order.**

[1]This dataset is from Stanford University.

2 Mining Social-Network Graphs (35 points)

(a) [20 pts] Solve the following problems, which are based on the exercises in the MMDS textbook.

- Exercise 10.3.2 (10 points)
  Suppose there is a community of $2n$ nodes. Divide the community into two groups of n members, at random, and form the bipartite graph between the two groups. Suppose that the average degree of the nodes of the bipartite graph is d. Find the set of maximal pairs (t, s), with $t \leq s$, such that an instance of $K_{s,t}$ is guaranteed to exist, for the following combinations of n and d:
  (a) $n = 20$ and $d = 5$.
  (b) $n = 200$ and $d = 150$.
  By "maximal," we mean there is no different pair (s', t') such that both $s' \geq s$ and $t' \geq t$ hold.

- Exercise 10.5.2 (modified) (10 points)
  Compute the MLE for the graph in Example 10.22 for the following guesses of the memberships of the two communities.
  (a) $C = \{w, x\}$; $D = \{y, z\}$
  (b) $C = \{w, x, y, z\}$; $D = \{x, y, z\}$

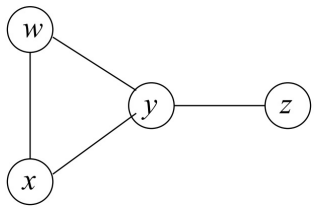


Figure 10.20: A social graph

(b) [15 pts] Implement the algorithm for finding triangles in MMDS Chapter 10.7.2. You will analyze part of the Facebook (now Meta) social network to identify communities.

You can download the dataset from the link below:
(Copy this link and paste it on the new tab)
http://socialnetworks.mpi-sws.mpg.de/data/facebook-links.txt.gz[2]

The dataset consists of three columns:

- `user_id1` : Corresponds to a user ID.
- `user_id2` : Corresponds to the user ID of the friend of `user_id1`.
- `time_stamp` : Corresponds to the UNIX timestamp with the time of link establishment. You do not need to use this column.

[2] MPI WSW dataset.

Although the data is in the form of a directed graph, consider it an undirected graph where each edge (`user_id1`, `user_id2`) means that the two users are friends with each other. There are about 63,700 users and 817,000 edges in the graph.

The algorithm is composed into two parts:

- `Step1` : Count heavy-hitter triangles.
- `Step2` : Count other triangles.

Refer to Lecture 9 and MMDS Chapter 10.7.2 to implement the algorithm in $O(m^{\frac{3}{2}})$ time where m is the number of edges. Please faithfully implement the algorithm in MMDS Chapter 10.7.2. If you use a different brute force method to count the triangles (e.g., going through each possible triples of nodes), **we will deduct 10 points out of 15 points** even if your answer is correct. In addition, you cannot use external graph libraries like `GraphX` and `Snap`.

Please use **command-line** arguments to obtain the file path of the dataset. For example, (for python 3) run:

```
python hw3_2_p3.py path/to/facebook-links.txt
```

After running, your code should `print` the total number of triangles. The output format is the following:

```
<TOTAL_NUMBER_OF_TRIANGLES>
```

The output should be a single line.

3 Large-Scale Machine Learning (25 points)

(a) [10 pts] Solve the following problems, which are based on the exercises in the MMDS textbook.

- Exercise 12.5.3 (modified) (10 points)
  An important property of a function f is *concavity*, meaning that if $x < z < y$, then

$$f(z) > \frac{y-z}{y-x}f(x) + \frac{z-x}{y-x}f(y) \tag{1}$$

  Less formally, the curve of f between x and y lies above the straight line between the points $(x, f(x))$ and $(y, f(y))$. In the following, assume there are two classes, and $f(x)$ is the impurity when x is the fraction of examples in the first class.

  (a) Prove that the GINI impurity is concave.
  (b) Prove that the Entropy measure of impurity is concave.

(b) [15 pts] Implement the gradient descent SVM algorithm described in MMDS Chapter 12.3.4 using Python.

You can download the dataset from the link below:
(Copy this link and paste it on the new tab)
http://www.di.kaist.ac.kr/~swhang/ee412/svm.zip[3]

There are two files:

- `features.txt`: Contains the feature vectors of 6K data points used for the training data. Each line is a feature vector of 122 components.
- `labels.txt`: Contains the corresponding labels of the 6K data points in `features.txt`.

Implement the *batch gradient descent* approach as described in Chapter 12.3.4 where, for each round, all the training examples are considered as a "batch." Example 12.9 may help with debugging.

For selecting the test data, use *k-fold cross validation* as described in Chapter 12.1.4. For this problem, set $k = 10$ where the 6K data points are sequentially divided into 10 chunks of size 600. In turn, let each chunk be the test data, and use the remaining 9 chunks be the training data. After training an SVM model on the training data, compute the accuracy of the model on the test data where we define accuracy as the portion of correctly predicted data points among all the data points in the test data. For example, if 300 out of 600 data points were correctly predicted, the accuracy is 0.5.

Finally, tune the C and η parameters for better accuracy.

Please **use command-line** arguments to obtain the file path of the dataset. (Do not fix the path in your code.) For example, (for python 3) run:

```
python hw3_3_p3.py path/to/features.txt path/to/labels.txt
```

After training the classifier on the training set, your code should `print` the average accuracy of the k partitions as well as the parameters C and η. The output format is the following (3 lines):

```
<AVERAGE ACCURACY>
<C>
<η>
```

We will give full credit as long as the accuracy is reasonably high (within 10 percent error).

[3]This dataset is from Stanford University.

Answers to Frequently Asked Questions

- For homework 3, we will only allow you to to import **pyspark, re, sys, math, csv, time, collections, itertools and os** in your code. Numpy will only be allowed in 2-(b) and 3-(b).
- In excercise 10.5.2, use the method used in section 10.5.3. You do not need to use gradient descent method.
- In 3-(b), you may decide the appropriate hyperparameters for gradient descent.
- An example of 5 decimal digits is 123.12345.
- Please use the print **built-in function** in Python to produce outputs
- `<TAB>` means `\t` in the Python print function.
- Time limit: 1-(a), 1-(b), 1-(c) should run within **30 minutes**.

- Q: In problem 1(b), I cannot use numpy, but vectors in pyspark.ml.linalg is okay, or it is also not allowed?
  A: We did not allow numpy because we wanted to implement matrix multiplication in parallel (i.e. each worker performs a single multiplication). If you can implement parallel multiplication with pyspark.ml.linalg you may use it.

- Q: I confused about problem 1(b). The print condition is print top 10 items and print the score just 5 decimal digits. I wonder considering 5 digits is just for the print statement or should it be considered before choosing the top10 item?
  A: Since the 5 decimal digit specification is only mentioned for the printing format, I think it should be considered only for the print statement(although it might not make much of a difference if considered before print).

- Q: For problem 1(b), do we have to round up the 5th digit when we reduce the decimals to 5 digits only? or just print the first 5 decimal digits as they are without considering the rounding. Also, what should we consider if there are ties between two pages in page rank score up to 5 digits?
  A: both rounding up and down will be considered correct.

- Q: I have a question in problem 1(b), PageRank. I am working on haedong machine. I think my algorithm is correct, but I keep receiving "java.lang.OutOfMemoryError: unable to create new native thread". Even though when I use smaller data file, I receive this error when the number of iteration is > 5. (When iteration ≤ 5, the calculated rank is the same as when I use only numpy.) For a test, I ran "bin/spark-submit examples/src/main/python/pagerank.py data/mllib/pagerank_data.txt 10" also, but it gave the same OutOfMemoryError message.
  A: Our solutions run fine on haedong machine. The only solution may not be trying to increase the number of processes. We limited the number of threads

per student so that everyone can use the server. Please try to modify the pagerank algorithm or configure the number of threads generated to avoid this issue. We tested the solution again on haedong server. It ran 50 iterations just fine. Your design choice of map and reduce function should be optimized to use smaller memory. Solving big data problems with limited resources is also an important part of the assignment. Since there are students who already solved the problem under the same conditions, we don't think it is fair to change the number of iterations and the other requirements of the assignment.
Matrix multiplication should not use more memory per iteration. You might be saving too much in memory. Also, make sure you don't use all of matrix M in each tuple generated by the map function.
We found the reason for the error. RDDs in spark lazily computes the map reduce functions. Once you use collect or top (functions that computes the results) spark allocates workers to the jobs. If you try to compute all 50 iterations in a single map reduce sequence without using collect or top, the computation graph becomes too large for the system to handle. Make sure to generate check points for some number of iterations and reconstruct the rdds rather than computing the entire 50 iterations in a single run.

- Q: I have a question in problem 1(b). If we do not implement the matrix multiplication by spark RDD methods, and if the answer is correct, will there still be some partial points?
  A: I will be testing your code via spark-submit. If you submit your code as a regular python file, you will not get any credit.

- Q: Is it allowed to use second order condition of convexity for problem 3(a)? or only using the definition of convexity is allowed?
  A: Any proof is fine as long as it is correct.

- Q: I have a question about problem 3. Is there no constraint on initial weight? I think I've not heard about how to set weight in class, could you give me advice for this? Also, by using k-fold validation, it means that iteration is fixed to k? Can I think an average accuracy as the average of the values of k tests?
  A: 1. You may use random initial random weights. Adjust the learning rate and the regularization parameter for maximum accuracy. 2. Please refer to Chapter 12.1.4 for k-fold cross validation.

- Q: I have question about problem 3(b) accuracy calculating. I'm curious that what does mean by "correctly predict". For example if the label of x1 is '-1' and the predicted $y < -1$, then it is correctly predicted? Or is it okay that I use softmax function... to exactly classify 1 or -1?
  A: Yes, if $y < -1$, it means the SVM classified it as -1. I think the decision boundary would be the $W \times X + b = 0$ hyperplane. If it is < 0, predicted -1, and if ≥ 0, predicted 1.