EE412 Foundation of Big Data Analytics, Fall 2018
HW4

Due date: 12/14/2018 (11:59pm)

Submission instructions: Use KAIST KLMS to submit your homeworks. Your submission should be one gzipped tar file whose name is `YourStudentID_hw4.tar.gz`. For example, if your student ID is 20161234, and it is for homework #4, please name the file as `20161234_hw4.tar.gz`. You can also use these extensions: tar, gz, zip, tar.zip. Do not use other options not mentioned here.

Your zip file should contain total five files; one PDF file for writeup answers (`hw4.pdf`), three python files (`hw4_1.py`, `hw4_2.py`, and `hw4_3.py`), and the Ethics Oath pdf file. Before zipping your files, please make a directory named `YourStudentID_hw4` and put your files in the directory. Then, please compress the directory to make a zipped file. Do not include Korean letters in any file name or directory name when you submit.

Submitting writeup: Prepare answers to the homework questions into a single PDF file. You can use the following template. Please write as succinctly as possible.

Submitting code: Each problem is accompanied by a programming part. 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.

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 people are permitted and encouraged. However, when the time comes to write your solution, such discussions (except with course staff members) are no longer appropriate: 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 Large-Scale Machine Learning (30 points)

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

The dataset can be downloaded from this link:
http://www.di.kaist.ac.kr/~swhang/ee412/svm.zip[1]

There are four 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, run:

```
python hw4_1.py path/to/features.txt path/to/labels.txt
```

After training the classifier on the training set, your code (`hw4_1.py`) should `print` the average accuracy of the k partitions as well as the parameters C and η and the number of iterations. The output format is the following:

```
<AVERAGE ACCURACY>
<C>
<η>
```

We will give full credit as long as the accuracy is reasonably high.

[1]This dataset is from Stanford University.

(b) [15 pts] Implement the parallel version of SVM as described in MMDS Chapter 12.3.6 using Spark. You may use the first approach where **w** and **b** are distributed to different mappers and updated in parallel before being averaged.

Use the same dataset and parameters as in 1(a).

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 hw4_2.py path/to/features.txt path/to/labels.txt
```

Your code (`hw4_2.py`) should `print` similar results as 1(a) in the same output format.

2 Mining Data Streams (40 points)

(a) [25 pts] Solve the following problems, which are based on the exercises in the Mining of Massive Datasets 2nd edition (MMDS) textbook.

- Exercises 4.4.1 and 4.4.2
  Suppose our stream consists of the integers 3, 1, 4, 1, 5, 9, 2, 6, 5. Our hash functions will all be of the form $h(x) = ax + b \bmod 32$ for some a and b. You should treat the result as a 5-bit binary integer. Determine the tail length for each stream element and the resulting estimate of the number of distinct elements if the hash function is:
  (a) $h(x) = 2x + 1 \bmod 32$.
  (b) $h(x) = 3x + 7 \bmod 32$.
  (c) $h(x) = 4x \bmod 32$.
  Do you see any problems with the choice of hash functions? What advice could you give someone who was going to use a hash function of the form $h(x) = ax + b \bmod 2^k$?

- Exercise 4.5.3
  Suppose we are given the stream 3, 1, 4, 1, 3, 4, 2, 1, 2 and apply the Alon-Matias-Szegedy Algorithm to estimate the surprise number. For each possible value of i, if X_i is a variable starting position i, what is the value of $X_i.value$?

- Exercise 4.6.1
  Suppose the window is as shown in Fig. 4.2. Estimate the number of 1's the the last k positions, for $k =$ (a) 5 (b) 15. In each case, how far off the correct value is your estimate?

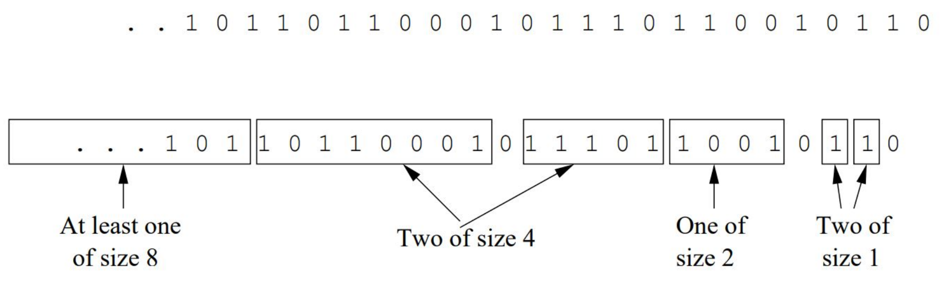

Figure 4.2: A bit-stream divided into buckets following the DGIM rules

(b) [15 pts] Implement the DGIM algorithm.

Implement the DGIM algorithm described in MMDS chapter 4.6 using Python.

The dataset can be downloaded from this link:
http://www.di.kaist.ac.kr/~swhang/ee412/stream.txt

The stream is randomly generated and contains a sequence of 10 million 0's and 1's. Each line contains a 0 or 1. Suppose that this sequence is the window (i.e., N = 10M) and that we need to summarize the window due to its large size. The goal is to estimate the number of 1's in the last k bits.

Please **use command-line** arguments to obtain the path for data file and multiple ks. (Do not fix the paths in your code.) For example, run:

`python hw4_3.py path/to/stream.txt` k_1 k_2 $\ldots$ k_M

After the run, your code (`hw4_3.py`) should `print` M estimated numbers of 1's in the last k_i bits of the window. For example, the first line is the estimated number of 1's in the last k_1 bits. You should make your algorithm as efficient as possible since M can be arbitrary.

3 Advertising on the Web (30 points)

Solve the following problems, which are based on the exercises in the MMDS textbook.

- Exercise 8.2.1
  A popular example of the design of an on-line algorithm to minimize the competitive ratio is the ski-buying problem. Suppose you can buy skis for \$100, or you can rent skis for \$10 per day. You decide to take up skiing, but you don't know if you will like it. You may try skiing for any number of days and then give it up. The merit of an algorithm is the cost per day of skis, and we must try to minimize this cost.

  One on-line algorithm for making the rent/buy decision is "buy skis immediately." If you try skiing once, fall down and give it up, then this on-line algorithm costs you \$100 per day, while the optimum off-line algorithm would have you rent skis for \$10 for the one day you used them. Thus, the competitive ratio of the algorithm "buy skis immediately" is at most 1/10th, and that is in fact the exact competitive ratio, since using the skis one day is the worst possible outcome for this algorithm. On the other hand, the on-line algorithm "always rent skis" has an arbitrarily small competitive ratio. If you turn out to really like skiing and go regularly, then after n days, you will have paid \$10n or \$10/day, while the optimum off-line algorithm would have bought skis at once, and paid only \$100, or \$100/n per day.

Your question: design an on-line algorithm for the ski-buying problem that has the best possible competitive ratio. What is that competitive ratio? Hint: Since you could, at any time, have a fall and decide to give up skiing, the only thing the on-line algorithm can use in making its decision is how many times previously you have gone skiing.

- Exercise 8.3.3
  Whether or not the greedy algorithm gives us a perfect matching for the graph of Fig. 8.1 depends on the order in which we consider the edges. Of the 6! possible orders of the six edges, how many give us a perfect matching? Give a simple test for distinguishing those orders that do give the perfect matching from those that do not.

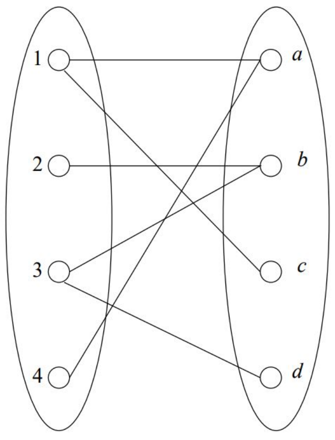

Figure 8.1: A bipartite graph

- Exercise 8.4.1 (Modified)
  Suppose that there are three advertisers, A, B, and C. There are three queries, x, y, and z. There is one ad shown for each query. Each advertiser has a budget of 2. Advertiser A bids only on x; B bids on x and y, while C bids on x, y, and z. All bids are either 0 or 1, and all click-through rates are the same. Note that on the query sequence $xxyyzz$, the optimum off-line algorithm would yield a revenue of 6, since all queries can be assigned.

  (a) Show that the greedy algorithm will assign at least 4 of these 6 queries.

  (b) Find another sequence of queries such that the greedy algorithm can assign as few as half the queries that the optimum off-line algorithm assigns on that sequence.