

Carnegie Mellon University
15-826 – Multimedia Databases and Data Mining
Fall 2014, C. Faloutsos

Homework 3

Due Date: Nov. 18, 2014, 3:00pm, in class
By: Chenying Hou (Q1), Yuning He (Q2), Yan Zhang
(Q3) & Peixin Zheng (Q4).

IMPORTANT REMINDERS

- All homeworks are to be done **INDIVIDUALLY**.
- All written answers should be **TYPED**.
- **Code submission:** As in previous homeworks, please create an `[andrew-id].tar.gz` file, and submit it to blackboard. Please use the template at [<http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW3/template.tar.gz>] Notice that it has a `makefile` for all questions, and directories `/q1`, `/q2`, `/q3`, etc, each with an individual `makefile`, and place-holder code. As before:
 - you may use any language that runs on linux/andrew, as long as your `make` creates all the necessary output
 - Replace our place-holders with your code, `tar/ gzip` *only the absolutely necessary* files, into (`[andrew-id].tar.gz`), and submit to blackboard.

Other reminders, FYI

- Weight: 30% of total homeworks weight = 3% of course weight.
- Designer/grader and expected effort for this homework (order-of-magnitude):
 - Q1(Q1A and Q1B): Chenying HOU ($\approx$ 3-4 hours)
 - Q2: Yuning HE ($\approx$ 3-4 hours)
 - Q3: Yan ZHANG ($\approx$ 3-4 hours)
 - Q4: Peixin ZHENG ($\approx$ 3-4 hours)

Q1A – SVD [15 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: Consider the 6-dimensional “mystery” dataset *mystery* at http://www.contrib.andrew.cmu.edu/~chenyinh/Q1A/data/mystery_large.txt and it is also placed in the template under /q1/data folder. Its points lie in a lower dimensionality hyperplane of dimensionality k - you have to guess k , using SVD.

Specifically, each 6-dimensional point $\vec{x} = (x_1, \dots, x_6)$ was generated by the equations:

$$\begin{aligned}x_1 &= a_1 * y_1 + b_1 * y_2 + \dots * y_k + \epsilon \\x_2 &= a_2 * y_1 + b_2 * y_2 + \dots * y_k + \epsilon \\&\dots \\x_6 &= a_6 * y_1 + b_6 * y_2 + \dots * y_k + \epsilon\end{aligned}$$

where $\vec{y} = (y_1, \dots, y_k)$ is random point in a hyper-box of $k \leq 6$ dimensions. The numbers $a_1, a_2, a_3 \dots, b_1, b_2, b_3 \dots$ are coefficients and they are the same, for all the points in the “mystery” dataset. And ϵ is a little noise. We will also refer to k as the “degrees of freedom” of the input cloud of points.

1. [10 pt] Use SVD and report the vector $(\sigma_1, \sigma_2, \dots)$ of singular values for this dataset, sorting them in descending order.
2. [5 pt] What is the value of k (= degrees of freedom) for this dataset?
3. [0 pt] OPTIONAL: what can you say about the (correlation) fractal dimension D_2 of this cloud of points?

What to turn in:

- **Code:** As mentioned in the beginning, please put your source files in the /q1a directory in the template [<http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW3/template.tar.gz>]. The datasets are under /q1a/data directory. And, make should print out the vector of singular values for this dataset.
- **Answers:** On hard copy, please submit
 1. the code for Q1A.1
 2. the answer for Q1A.1 and Q1A.2

Q1B – SVD [10 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: Can you find fraudsters on Twitter? That is, accounts, who get money, to follow others and make them look important? Typically, they form the so-called “bipartite cores” (defined later), and SVD can spot them - we shall see how.

Instead of Twitter data, consider the “Patent Citation” dataset at <http://www.contrib.andrew.cmu.edu/~chenyinh/Q1B/data/Patents.txt>. (FYI, The patents can all be searched at USPTO website <http://patft.uspto.gov/netahtml/PT0/srchnum.htm>.)

We injected a “ $(k-k'-p)$ near bipartite-core”, that is, a source-group of k patents and a destination-group of k' , where almost all (p fraction) of the patents in the source-group cite all patents in the destination group. Thus, there is an abnormally dense block in the adjacency matrix. In real life you won't know, but in this exercise we are telling you that the density $p \approx 0.9$, $k \approx 40$, and $k' \approx 250$.

Observations - reminders

- A near bipartite-core of M rows, N columns and density p , has leading singular value $\sigma_1 \approx p\sqrt{MN}$
- If a graph G consists of two disconnected subgraphs $G1$ and $G2$, its singular values $(\sigma_1, \dots)$ are the union of the singular values of $G1$ and $G2$.
- As we said in class, if the input matrix consists of blocks, each block will correspond to a singular value σ_i for some i , in the sense that if you set σ_i to zero and reconstruct the matrix, the block will disappear.

Hints:

1. Use the “EigenSpoke” approach and the “EE-plots” (Eigen spoke paper <http://www.cs.cmu.edu/~badityap/papers/eigenspokes-pakdd10.pdf>).
2. If you need to re-number the patentIDs so that they start from '1', try <http://www.contrib.andrew.cmu.edu/~chenyinh/Q1B/anonymizer-all.tar>. Use the `-p -c1,2` flags; the former prints a lookup table from old patentID to new patentID.
3. Make **sure** you use **sparse** matrices - some packages turn sparse matrices to dense, by default (and you will run out of memory).

To do:

1. [2 pt] Use SVD to find the dense block we injected:
 - (a) What is your estimate for the (largest) singular value of the injected subgraph?
 - (b) How many components n do you have to ask for, in the truncated SVD of the input graph, so that the injection shows up? (*Hint:* : In matlab, `svds(A,n)` does the truncated SVD, i.e., it gives the top n singular values and corresponding vectors.

2. [2 pt] For our injection (dense block), what is the size k of the source-group, and the size k' of the destination group?
3. [2 pt] List the k ids of the source-group in increasing order.
4. [2 pt] Describe briefly how you determined the source-group. We expect something like “*I chose the entries with negative scores, in $\vec{v}_9$, the 9-th right-singular vector.*”
5. [2 pt] Similarly, describe briefly how you determined the destination-group.

What to turn in:

- Answers: On hard copy, please submit the answers to all questions: Q1B.1-5.

Q2 – Hadoop [25 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: In this question, we will use Hadoop 1.2.1, to find the in-degree and out-degree used in Patent citation network. Note, this question has been designed for and tested on the GHC Andrew machines (ghc#.ghc.andrew.cmu.edu where # ranges from 01-79), ¹. **Each student has a specific machine** assigned to them in the grade center on Blackboard that they should use (under the column “andrew-machine-id”).

Setup:

- Download the script <http://www.contrib.andrew.cmu.edu/~yuningh/build-hadoop-1.2.sh>
- Place it in the directory where you want to run Hadoop and solve this question.
- Check the working jvm version on your andrew machine in `/usr/lib/jvm`, and modify the version number in the script.
- Check `~/.cshrc` that you have set a `$JAVA_HOME`.
- Run

`bash build-hadoop-1.2.sh; source setenv.csh`

FYI (For your information): You will likely need to enter your Andrew password a few times to while the script is running. This script will install Hadoop and HDFS locally in the directory you placed the original script ². When the script is done running, Hadoop will be running on the machine, `start-hadoop.sh` and `stop-hadoop.sh` will be in your directory that will respectively start and stop Hadoop, and the necessary resources for this question will be in place, which we will go over later.

Careful: It should not happen, but if another student is currently running Hadoop on the machine you are on, you can not use that machine. In that remote case just log into another GHC machine and try starting your Hadoop instance there. (Please try the machines from 70-79 since these will not be assigned to any students.) The script will warn you if there is an issue.

IMPORTANT: Hence, **Please shut down your Hadoop instance** when you log out so that other students can use the machine!

(If it appears someone else is using the machine, you can look for long running java instances in `top` and email the user to turn off their Hadoop.) When using Hadoop on a new machine, you may get an error when running your commands that Hadoop is in “safe mode.” If you

¹If, in the future, you want to run it on some other machine, please reset the `$JAVA_HOME` term in the script. See the comments in the script.

²Note that the script assumes that your default shell on the Andrew machines is `csh`, which it appears to be by default; if you have changed this you (to `bash` for example) you will need to add the correct `$JAVA_HOME` and `$HADOOP_PREFIX` to your `.bashrc` file.

get this error, go to the `hadoop-1.2.1` directory and run the following command `bin/hadoop dfsadmin -safemode leave`

Once you have done the set-up, please do the following:

1. **[2 pt]** First, upload the data file `cit-Patents-small.txt` to HDFS. The data file ³ can be found in your directory `hadoop-1.2.1/Patents/`. An edge $A \rightarrow B$ means A cites B. Upload the entire Patents directory to HDFS and then verify it uploaded correctly.

To do this, you will want to go into the `hadoop-1.2.1/` directory and use commands of the form `bin/hadoop fs`. Entering that command will give you a list of possible commands to run such as `-ls` and `-mkdir`, which of course correspond to their Unix equivalent `ls` and `mkdir`. (For more information you may want to see the HDFS User Guide http://hadoop.apache.org/docs/r0.18.3/hdfs_shell.html) Your home path is just `/` When this step is complete your HDFS should have a directory `/Patents` with a file `/Patents/cit-Patents-small.txt`.

2. **[2 pt]** We will now test that Hadoop is working with a classic example, WordCount. In your directory to `hadoop-1.2.1/Degree/` you will find a script `compile.sh` and a Java program `WordCount.java`⁴. Running `bash compile.sh` will compile the WordCount example and produce a jar `WordCount.jar`.

Run the WordCount example on the `cit-Patents-small.txt` data and report the number of unique NodeId (Please consider FromNodeId and ToNodeId as a whole) in the data file. (The Unix command `wc -l` may be handy.) General instructions on how to run the code are at http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html#Usage, although in our example there are some slight deviations.

3. **[4 pt]** Copy the WordCount example into a new java file, e.g. `InDegree.java`. Modify the WordCount example there, so that to count the in-degree for each NodeId.
4. **[2 pt]** Which are the top 10 NodeIds with the highest in-degree? Report them, along with their in-degree (you may use Unix's `sort` command).
5. **[4 pt]** Copy the WordCount example into a new java file, e.g. `OutDegree.java`. Modify it, to count the out-degree for each NodeId.
6. **[2 pt]** For the top 10 NodeIds with the highest out-degree, report the NodeId and the out-degree.

³The data file can also be downloaded independently from <http://www.contrib.andrew.cmu.edu/~yuningh/cit-Patents-small.txt>.

⁴The two can also be downloaded from <http://www.contrib.andrew.cmu.edu/~yuningh/compile.sh> and <http://www.contrib.andrew.cmu.edu/~yuningh/WordCount.java>. This is modified from the Apache Tutorial http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

7. [7 pt] Now we want to see the *importance* of each node and we will use a simplified version of PageRank. Then, the importance of a node is defined as the sum of importance of the nodes that point to it, as well as its own importance. The initial importance of each node is 1. Modify your program and run it for 5 iterations.

Hints:

- NO need to normalize the vectors for this exercise (although in a production system, we should definitely normalize to unit vectors, after each iteration).
 - If you meet your AFS account quota, delete the unused data in HDFS.
 - Also, some pre-processing of the dataset may be needed, to form the data structure for your algorithm.
8. [2 pt] Give the 5 most important patents, and specifically the *PatentId*, *patent name* and its *importance value*. For the patent name, use the website <http://patft.uspto.gov/netahtml/PTO/srchnum.htm>.

Hints:

- For example, PatentId 3824442 has patent name “INVERTER CIRCUITS”.
- Some patents might be too old and only have an image. Click the image button to check the patent name.

What to turn in:

- **Answers:** Turn in a print out of all of the code necessary to complete the tasks above as well as their results. This should include the necessary Unix commands, etc.
- **Code:** In the `[andrew-id].tar.gz` file, turn in all code that was necessary to complete this question. Also include a text file with the answers to the questions.

Q3 – Tensors [25 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: In this question, you will learn about the CP/PARAFAC tensor decomposition, and how to use to spot interesting structures in data. Suppose that you have a traffic trace of Internet, consisting of a list of triplets

src-IP dst-IP port-number

This can be envisioned as a 3-mode tensor. For example the triplet

128.1.1.1 128.2.1.115 5432

means that the machine in IP address 128.1.1.1 got connected to the second machine, over port 5432 (which is the default port for `postgres`).

You will use the CP/PARAFAC decomposition of such a tensor to discover (3-mode) communities, where a set of source addresses connects to a set of destination addresses, using a set of destination-ports. Such a “community” may indicate an attack: for example, some members of a botnet, may be probing, say, CMU servers, looking for open database ports (`postgres`, `mysql`, etc).

For your convenience, we re-numbered the IP addresses and ports, to make them integers. Thus, the dataset is in *tab separated* format, with one quadruplet per line, of the form

i j k value

where the indices start from 1, and `value` is always 1, meaning there was 1 packet from source i to destination j over port k .

Implementation Details:

1. [0 pt] Download and install the Tensor Toolbox for Matlab, from <http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.5.html>
2. [0 pt] Read the tab separated file <http://www.contrib.andrew.cmu.edu/~yanzhan2/box.dat> into a sparse tensor in Tensor Toolbox. [*Hint*: : type `help dlmread` and `help sptensor`].
3. [5 pt] Run the CP/PARAFAC decomposition on the tensor you created, for rank $R = 2$. Provide 6 plots, one for each of the components ($\vec{a}_1, \vec{a}_2, \vec{b}_1, \vec{b}_2, \vec{c}_1, \vec{c}_2$) That is, for say, component $\vec{a}_1$, plot the score $a_{1,i}$, versus the index i . A high value for $a_{1,i}$ means that source-IP i participates in the first 'concept'. [*Hint*: : Check the documentation of Tensor Toolbox by typing `help tensor_toolbox`. The function you will need is `cp_als`.]

4. [5 pt] Consider the first component (with the three vectors $\vec{a}_1, \vec{b}_1, \vec{c}_1$). Entries with non-zero values of the rank-one component, “belong” to that 3-mode community. Give
 - (a) [1 pt] the count of src-IDs in this community
 - (b) [1 pt] the src-IDs of the community
 - (c) [1 pt] the count of dst-IDs in this community
 - (d) [1 pt] the dst-IDs of those users
 - (e) [1 pt] the ports used in this community
5. [5 pt] *SQL, for tensor analysis:* It turns out that SQL is enough, to help us find the rank-1 decomposition of a tensor. Implement the method in <http://www.cs.cmu.edu/~maraujo/papers/pakdd14.pdf> (Equation 1, page 6), to find the rank-one decomposition. Attach your SQL code.
Hint: : Use **SQLite** or **PostgreSQL** for this question. *Hint:* : After every iteration, normalize the vectors, by dividing each entry with the (absolute) max value for that vector Ie., if the original vector is (3, 2.2, -5), we divide everything by | - 5|, and we get (3/5, 2.2/5, -1).
6. Apply your SQL code from the previous question, to another dataset: <http://www.contrib.andrew.cmu.edu/~yanzhan2/LBNL-sdp.dat>, which is in the same format. Use SQL, and do 20 iterations Answer the following questions:
 - (a) [4 pt] plot the 3 vectors of the rank-one decomposition, ie., $\vec{a}_1$ (for source-IPs), $\vec{b}_1$ (for destination-IPs), and $\vec{c}_1$ (for destination-ports).
 - (b) [1 pt] Report the count of *significant* src-IDS in this first component. We define a_i be *significant* if it is within 10% of the maximum, that is if it has value larger than $0.9 * \max(a_i)$ for i in $1, 2, \dots$
 - (c) [1 pt] List the *significant* src-IDs in this component.
 - (d) [1 pt] Report the count of *significant* dst-IDs in this component.
 - (e) [1 pt] List the *significant* dst-IDs in this component.
 - (f) [1 pt] Given the count of *significant* port numbers in this component.
 - (g) [1 pt] List the *significant* port numbers in this component.
 - (h) [0 pt] OPTIONAL: Find the functionality of the port numbers above (say, 8080 is the http port). Could they indicate an attack?

What to turn in:

- **Code:** In the [andrew-id.tar.gz] file, put all the code you used to generate these results
- **Answers:** On hard copy, please submit your answers, plots and code for Q3.3 - Q3.6.

Q4 – Fourier, wavelets [25 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: The goal is to become familiar with Digital Signal Processing (DSP) and the data mining capabilities of the Fourier and wavelet transforms.

Implementation Details:

1. [1 pt] Consider the signal in file `noiseWithSinusoid.mat` inside the template file [<http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW3/template.tar.gz>], under `q4` directory. Each row corresponds to the value of a single timetick (indices start from 1). Somewhere inside this signal, we have injected a sinusoid, starting from time t_1 and ending at time t_2 , with frequency f . What is your best guess for t_1 , t_2 , and f ?

Hint: Use the Haar Wavelet Transformation; plot the Wavelet scaleogram.

Hand in:

- Your code to generate the Wavelet scaleogram
- The plot of the Wavelet scaleogram
- Your estimate of t_1 , t_2 , and f . (in place of f , we will also accept the period T , in number of time-ticks).

2. [12 pt] Similar to the above question, but with a real sound, as opposed to a synthetic, sinusoid.

Consider the signal in the file `noiseWithMosquito.mat` inside the template file [<http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW3/template.tar.gz>], under `q4` directory. Again, each row corresponds to the value of the signal at this timetick (indices start from 1). Somewhere inside this (almost-noise) signal, we have injected another signal, which is the sound of a mosquito flapping its wings. The mosquito sound starts from time t_1 and ends at time t_2 . Use the Haar Wavelet Transformation to estimate t_1 , t_2 , and the approximate frequency of the mosquito sound f . Plot the Wavelet scaleogram.

Again, hand in:

- The code to generate the Wavelet scaleogram
- The plot of the Wavelet scaleogram
- Your estimate of t_1 , t_2 , and f . Since there may be several harmonics, report the strongest one (or the top few strongest ones). (in place of f , we will also accept the period T , in number of time-ticks).

3. Here we show how DSP can help us visualize a large collection of sounds.

Consider the dataset at [<http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW3/q4/Train.zip>]. containing time series in form of `.wav` files, each representing the sound of one insect (usually, a mosquito). We also have class label for each time series

in `classLabel.mat` (and, identically, `classLabel.txt`).

- (a) [10 pt] Each of the time series is approximately $15k$ time ticks in length. We want to extract some features from each time series, so that we can better visualize them, and eventually classify them. We try here the Fourier Transformation. Figure 1 shows a typical Fourier spectrum of a mosquito sound. There is a high spike at around $f=400\text{Hz}$, with additional spikes at multiples of that frequency (the so-called “harmonics”). The goal is to find the first 2 frequencies f_1 and f_2 ,

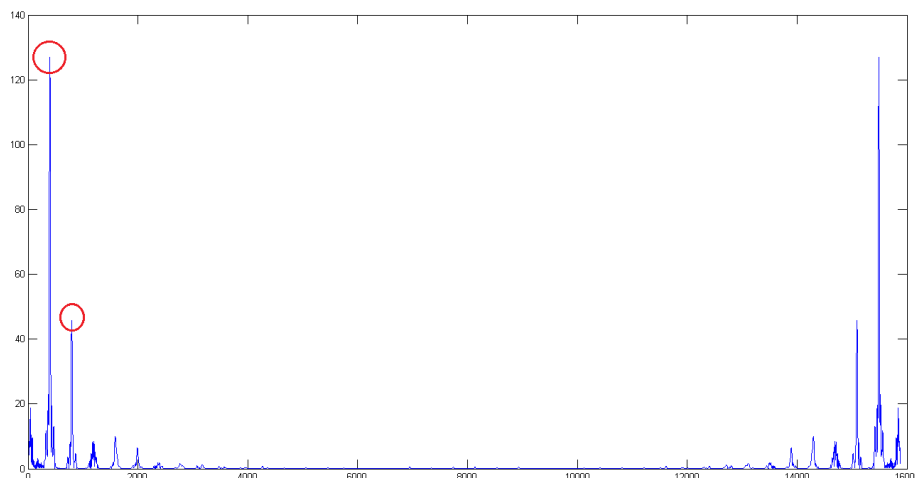


Figure 1: An example of the Fourier spectrum of a mosquito sound. We want the frequencies for the spikes marked by red circle.

for each sequence, and treat the sequence as a point in 2-d space, with coordinates (f_1, f_2) .

Finding the two spikes is tricky, for real, noisy signals. In the sample spectrum in Figure 1, the red circles mark the frequencies we want. You may use any method you want, but we recommend the hint below:

Hint: : We recommend the following steps and heuristics: For each time series, do the Fourier Transformation. Ignore the mirror part (see lecture notes); find out all the spikes (= “local maxima”) and pick the **3** frequencies f_1, f_2, f_3 , that that satisfy the following heuristics:

- Each frequency f_i should be $> 100\text{Hz}$. (This drops local maxima that occur in unnaturally low frequencies)
- Any two of these 3 frequencies should be more than 100Hz apart. (This is because, close to a single spike, there may be little ups and downs, as in the example spectrum of Figure 1.)
- Then, select those frequencies with the highest amplitude.

Finally, among these 3 frequencies, sort them ($f1 < f2 < f3$) and use the first two as the (x,y) coordinates of that insect sound.

Please write code to generate the two coordinates for each insect sound.

- (b) [2 pt] Using the two frequencies as coordinates, plot all time series. Color-code the class label, that is, sounds of the same class, should correspond to points with the same color.
- (c) [0 pt] OPTIONAL: Do you find anything interesting on the plot? Try to explain it.
- (d) [0 pt] OPTIONAL: Do you see obvious grouping of one or more classes of insects/mosquitoes?
- (e) [0 pt] OPTIONAL: Do you find any obvious outliers?

What to turn in:

- **Code:** : All the code you use to generate these results.
- **Answers:** : On hard copy, your answers to all the questions above, including code, answers, and plots.