CS4786 Sp2015 - competition 1 instructions (an evolving document)

Prefatory note It's on! The competition has officially begun.

Due date The due date is April 22nd, 11:59PM on CMS. Submit what you have at least once by an hour before that deadline, even if you haven't quite added all the finishing touches — CMS allows resubmissions up to, but not after, the deadline. If there is an emergency such that you need an extension, contact the professors.

How to form groups for this competition You may work in groups of one up to four[1], where your group was formed by you (ASAP, and certainly by April 15th) both on the "First competition preliminary submission" CMS assignment **and** via our google-doc form for setting up the Kaggle-team invites. (We will automatically transfer the groups from the CMS preliminary assignment to the actual CMS competition assignment.) Please ensure that each member of the group can individually defend or explain your group's submission equally well.

> 1. Footnote: *The choice of the number "four" is intended to reflect the idea of allowing collaboration, but requiring that all group members be able to fit "all together at the whiteboard", and thus all be participating equally at all times. (Admittedly, it will be a tight squeeze around a laptop, but please try.)*

Collaboration and academic integrity policy

Students may discuss and exchange ideas with students not in their group, but only at the conceptual level.

We distinguish between "merely" violating the rules for a given assignment and violating academic integrity. To violate the latter is to commit fraud by claiming credit for someone else's work. For this assignment, an example of the former would be getting detailed feedback on your approach from person X who is not in your group but stating in your homework that X was the source of that particular answer. You would cross the line into fraud if you did not mention X. The worst-case outcome for the former is a grade penalty; the worst-case scenario in the latter is academic-integrity hearing procedures.

The way to avoid violating academic integrity is to always document any portions of work you submit that are due to or influenced by other sources, even if those sources weren't permitted by the rules.[2]

> *2. Footnote: We make an exception for sources that can be taken for granted in the instructional setting, namely, the course materials. To minimize documentation effort, we also do not expect you to credit the course staff for ideas you get from them, although it's nice to do so anyway.*

Data

First, download the data zipfile, unzip it, and read the included README file. Note that there is an *err* error in the documentation for the speech graph: here is a correction - additions in italics.

A link to the page where you can view diffs of precisely what text changed between different versions

Version	Published	Changed By	Comment
CURRENT (v. 8)	**Apr 22, 2015 11:28**	**Lillian Lee**	requirement adde
v. 7	Apr 21, 2015 01:42	**Lillian Lee**	inserted implemer
v. 6	Apr 19, 2015 15:27	**Lillian Lee**	change stray "1-3
v. 5	Apr 19, 2015 15:21	**Lillian Lee**	fix typo in challeng
v. 4	Apr 19, 2015 15:20	**Lillian Lee**	note time period w
v. 3	Apr 19, 2015 09:33	**Karthik Sridharan**	
v. 2	Apr 19, 2015 09:32	**Karthik Sridharan**	
v. 1	Apr 18, 2015 15:25	**Lillian Lee**	added hint for cha

> *Two speeches are connected in the graph (with some probability) if they ~~both are in favor or against of a certain legislation~~.* are both speeches in favor (although they may be in favor of two different bills) or both speeches against (although they may be against two different bills.) *Two speeches are also connected (with some probability) if they occur in the debate for same legislation. In general, with some low probability noisy links might occur.*

Q1 (Challenge 1: Two clusters, "For" and "Against").

The first challenge is to cluster the 2740 speeches into two groups, the first group consisting of "Against" speeches and the second group consisting of "For" speeches. To start you off, you are told that points (rows, where the first row is number 0) **2**, **13**, **18**, **24** are examples of speeches that belong to category "Against" (=label 0) and speeches **1**, **3, 27**, **177** are examples of speeches that belong to the "For"(=label 1) category. Your goal is to produce a file consisting of 2741 comma-separated lines, where, to conform to what Kaggle is expecting:

1. The first line should be "Id,Prediction".
2. Each subsequent line consists of two comma-separated numbers: the first number is which speech is being referred to, *starting with speech **0***, and the second number is either a "0" or "1". So, the second line in your csv file (the one after "Id, Prediction" will be either "0,0" or "0,1". You will name this file "votes.csv". We will evaluate your grouping against the ground truth (which is of course hidden from you — but from the information above, you know that some of its lines are "1,1", "2,0", "3,1", "177,1", etc.). Your goal is to make as few mistakes as possible.

Q2 (Challenge 2: 38 clusters, one for each debate). Here your goal is to cluster the points into 38 groups where each group is meant to represent speeches made in a single debate. Your goal is to produce a file consisting of 2741 lines, where, to conform to what Kaggle is expecting:

1. The first line should be "Id,Prediction".
2. Each subsequent line consists of two comma-separated numbers: the first number is which speech is being referred to, *starting with speech **0***, and the second number ranges between 0 and 37 inclusive, indicating which debate the speech belongs to. So, the second line in the file (the one after ("Id,Prediction") could look like "0,3" or "0,14". You will name this file "debates.csv".

Deliverables and instructions: Part of the competition is on Kaggle so do sign up and compete with each other. Some of the instructions on this will (or have already been) given in lecture.

At the end of the competition you will be required to submit two data files described in the two challenges above *and* (as a zip file) the code for your best-performing submission, such that we could reproduce your results if necessary for grading. (In order to do so, we need to know the values of any parameters you set; please include these in your README and writeup.) But more importantly you need to also submit a writeup/report ("writeup.pdf") of the things you tried and why you tried them (irrespective of whether they worked or failed). The writeup will count for at least as much of the grade as the empirical results of the final cluster assignments you submit.

Here are a few other remarks:

0. We are declining to pre-specify too many regulations on your writeup because we want this competition to be somewhat open-ended, so with respect to format, our intent is just that you submit "something we find reasonable". To that end, we do require that

1. this report is at least 5 pages long and not more than 15 pages, but with font sizes, spacing, etc., we just expect you to do something reasonable. (See item 6 below.)
2. (added after some grief with providing the initial feedback) include all of your names, netids, and the name of your Kaggle team at the beginning of your document

1. Include visualizations of both successful and unsuccessful trials.

2. Make a note of all successful and unsuccessful methods you tried. Explain why you made the choices you made and why you expected them to work both for successful and not-so-successful choices and take a shot at explaining why the less successful ones were in fact not so successful.

3. Organize your writeup into sections where each section (and its title) corresponds to a particular method.

4. You are certainly encouraged to try methods you might have picked up outside those covered in class and maybe even extensions you develop on your own for the problem!. If you use methods other than ones covered in class, **do compare the performance both empirically and conceptually with (a reasonable choice of) methods covered in class.**

5. We will definitely award karma points for work that goes above and beyond, and have made this competition somewhat open-ended to that end. Our definition of "work" emphasizes novelty and well-foundedness of methods tried, but also includes presentation aspects of your writeup, and we are hoping to see some striking visualizations (for instance, that demonstrate a clever analysis of an algorithm or its failings/successes.)

6. **If you turn in a draft in the first week** on the "preliminary assignment" for this competition, **we can provide feedback** on whether your current version of your writeup seems reasonable to us.

7. **The first challenge is also set up on Kaggle as a competition,** and your ranking on Kaggle will form a small part of your grade for this assignment. Much more importantly, you can (and, really, ought to) get feedback on how good your approaches are by getting them judged on Kaggle. The earlier you start, the more tries you get, since your group is limited to two Kaggle submissions a day!

At the same time, though, because the number of submission you can make per day on the Kaggle competition is limited, you should also be additionally evaluating your approaches by visualizing or analyzing the structure of their results.

8. **Please check for new/edited drafts of this instruction document, as it will keep getting updated. (a)** We recommend you set things up so that you get emailed an update when this page pages: to do this, log in to confluence using your netid and netid password and then s et a "watch" on this page by clicking the "Watch" option near the top right (it has an "eye" icon next to it). **(b)** Regardless of whether you set a watch or not, the right-hand side of this page shows when the page has been changed and, hopefully, a comment documentation what the changes were. Even better, you can see a diff of what changes have been made between different versions ---- a handy feature that is the main reason we switched to a wiki format for these instructions. To do so, click on the link indicated at the top of the right-hand side of this page. This will bring you to the "Page History" page. There, you can select checkboxes to select any two current or past versions of this page; then, click on "Compare selected versions".

9. The zipfile of code you submit on the final (not the preliminary) CMS "Assignment" needs to include a README.txt file that explains how we can run your code. Include in the README the exact values of any parameters you set to achieve your best result. The code should be "standalone" in the sense you should include any extra modules, libraries, etc. that you wouldn't expect our standard installs of R, Matlab, python, numpy, etc. to necessarily have.

10. A*dditional revealing hints:* (we had said we would reveal more information a bit at a time as the competition wore on.):

1. The first 25,000 features roughly correspond to sentiment features (relative frequencies of some words having to do with being "For" or "Against" something), and the second 25,000 features roughly correspond to topic features (relative frequencies of some words that are related to the subject matter of the conversation, or at least not known to be sentiment related).
2. Our solutions involve projection onto a small number of eigenvectors, but not just two.
3. Here is a .csv file listing, for (hidden) cluster label, 0-37, the true size of that cluster. You can use this is a guide for deciding how to label your clusters.
   (Be sure to note that this file was *incorrect* between April 18th 15:25 and April 19th, 09:33 - Prof Lee interpreted an email conversation incorrectly!)
4. Here is a .csv file where each line indicates line number (data point) followed by the cluster the data point belongs to. (Eg. "21, 0" means line/data point 21 belongs to debate 0)

11. Include in your writeup the values of any parameters you set for your best results. These values should also be in your README, as stated in point 9.

Python implementation notes

1. The scikit-learn package can be quite slow. Some staff recommendations are:
   a. Do the requisite matrix multiplications yourself, and you may fall back on doing multiplication "piecewise" (e.g., compute one row of the product at a time).
   b. Compile numpy with OpenBLAS.
   c. Consider the scipy.sparse utilities, including scipy.sparse.linalg.eigs; student-suggested usage: w,v = sparse.linalg.eigs(M, k=k, which='SM')
2. Student-suggested k-means code that allows setting the initial seeds: kmeans2 in scipy, KMeans in sklearn (*kmeans=KMeans (n_clusters=2,init=centroids,n_init=1, max_iter=300)*, put your own centroids in the init argument)
3. See Piazza @198 regarding numpy.savez, numpy.load, and pickling.

Matlab implementation notes

1. Student-suggested k-means code that allows setting the initial seeds:
   idx = kmeans(X, k, 'Start', mu);
   where each row of mu is an initial seed location.
2. Student suggest remark about using 'SM' or 'SA' for eigs: (Smal lest magnitude = finds eigenvectors with lowest absolute value, Smallest algebraic = finds eigenvectors with lowest numerical value): use SM, but to avoid problems with invalid shift, specify sigma to be something very small.

General implementation notes

1. If you only need a few eigenvectors, don't compute all of them.