ELEN 4903: Machine Learning

Columbia University, Spring 2016

Homework 4: Due May 2, 2016 by 11:59pm

Please read these instructions to ensure you receive full credit on your homework. Submit the written portion of your homework as a *single* PDF file through Courseworks (less than 5MB). In addition to your PDF write-up, submit all code written by you in their original extensions through Courseworks (e.g., .m, .r, .py, etc.). Any coding language is acceptable. Do not wrap your files in .rar, .zip, .tar and do not submit your write-up in .doc or other file type. Your grade will be based on the contents of *one* PDF file and the original source code. Additional files will be ignored. We will not run your code, so everything you are asked to show should be put in the PDF file. Show all work for full credit.

Late submission policy: Late homeworks will have 0.1% deducted from the final grade for each minute late. *Your homework submission time will be based on the time of your* ***last*** *submission to Courseworks. I will not revert to an earlier submission!* Therefore, do not re-submit after midnight on the due date unless you are confident the new submission is significantly better to overcompensate for the points lost. Submission time is non-negotiable and will be based on the time you submitted your last file to Courseworks. The number of points deducted will be rounded to the nearest integer.

Problem 1 (Markov chains) – 50 points

In this problem, you will rank 759 college football teams based only on the scores of every game in the 2015 season. The data provided contains the result of one game on each line. For the ith line, the information contained in "scores" is

```
scores(i,1) = Team1 index,  scores(i,2) = Team1 points,
scores(i,3) = Team2 index,  scores(i,4) = Team2 points.
```

If `scores(i,2) > scores(i,4)` then Team1 wins and Team2 loses, and vice versa. The index of a team refers to the row of "legend" where that team's name can be found.

Construct a 759×759 random walk matrix M on the college football teams. First construct the unnormalized matrix $\widehat{M}$, initially set to all zeros. For a particular game i, let j_1 be the index of Team1 and j_2 the index of Team2. Then update

$$\widehat{M}_{j_1j_1} \leftarrow \widehat{M}_{j_1j_1} + \mathbb{1}\{\texttt{Team1 wins}\} + \frac{\texttt{points}_{j_1}}{\texttt{points}_{j_1}+\texttt{points}_{j_2}},$$

$$\widehat{M}_{j_2j_2} \leftarrow \widehat{M}_{j_2j_2} + \mathbb{1}\{\texttt{Team2 wins}\} + \frac{\texttt{points}_{j_2}}{\texttt{points}_{j_1}+\texttt{points}_{j_2}},$$

$$\widehat{M}_{j_1j_2} \leftarrow \widehat{M}_{j_1j_2} + \mathbb{1}\{\texttt{Team2 wins}\} + \frac{\texttt{points}_{j_2}}{\texttt{points}_{j_1}+\texttt{points}_{j_2}},$$

$$\widehat{M}_{j_2j_1} \leftarrow \widehat{M}_{j_2j_1} + \mathbb{1}\{\texttt{Team1 wins}\} + \frac{\texttt{points}_{j_1}}{\texttt{points}_{j_1}+\texttt{points}_{j_2}}.$$

After processing all games, let M be the matrix formed by normalizing the rows of $\widehat{M}$ so they sum to 1.

- Let w_t be the 1×759 state vector at step t. Set w_0 to the uniform distribution. Therefore, w_t is the distribution on the state after t steps given that the starting state at time 0 is uniformly distributed.
- Use w_t to rank the teams by sorting in decreasing value according to this vector. List the top 25 teams and their corresponding values in w_t for $t = 10, 100, 1000, 2500$.
- We saw that w_∞ is related to the first eigenvector of M^T. That is, we can find w_∞ by getting the first eigenvector and eigenvalue of M^T and post-processing:

  $$M^T u_1 = \lambda_1 u_1, \quad w_\infty = u_1^T / \left[\sum_j u_1(j)\right]$$

  This is because $u_1^T u_1 = 1$ by convention. Also, we observe that $\lambda_1 = 1$ for this specific matrix. Plot $\|w_t - w_\infty\|_1$ as a function of t for $t = 1, \ldots, 2500$. What is the value of $\|w_{2500} - w_\infty\|_1$?

Problem 2 (Nonnegative matrix factorization) – 70 points

In this problem you will factorize a $n \times m$ matrix X into a rank-K approximation WH, where W is $n \times K$, H is $K \times m$ and all values in the matrices are nonnegative. Each value in W and H can be initialized randomly, e.g., from a Uniform(0,1) distribution. (See a hint about the implementation below.)

<u>*Part 1*</u>: The data to be used for Part 1 consists of 1000 images of faces, each originally 32×32, but vectorized to length 1024. The data matrix is therefore 1024×1000.

- Implement and run the NMF algorithm on this data using the *Euclidean penalty*. Set the rank of the factorization to 25 and run for 200 iterations.
- Plot the objective as a function of iteration.
- Pick 10 columns from W and show them as 32×32 images. For each vector you select from W, find the column of H that places the highest weight on this vector and show the corresponding column of X as a 32×32 image.

<u>*Part 2*</u>: The data to be used for Part 2 consists of 8447 documents from *The New York Times*. (See below for how to process the data.) The vocabulary size is 3012 words. You will need to use this data to constitute the matrix X, where X_{ij} is the number of times word i appears in document j. Therefore, X is 3012×8447 and most values in X will equal zero.

- Implement and run the NMF algorithm on this data using the *divergence penalty*. Set the rank to 25 and run for 200 iterations. This corresponds to learning 25 topics.
- Plot the objective as a function of iteration.
- After running the algorithm, normalize the columns of W so they sum to one. Pick 10 columns of W. For each column you select show the 10 words having the largest weight according to that vector and show the weight. The ith row of W corresponds to the ith word in the "dictionary" provided with the data.

Hint for Problem 2: When dividing, you may get 0/0 = NaN. This will cause the algorithm to return all NaN values. In the division, add a very small number (e.g., 10^{-16}) to the denominator to avoid this.

```
About the text data used in Part 2:

MATLAB
- Xid is a cell array. Xid{d} is a vector giving the indexes of words
      appearing in document d. Any word not listed doesn't appear.
- Xcnt is a cell array. Xcnt{d} is a vector indicating that word
       Xid{d}(i) appears Xcnt{d}(i) times in document d.

CSV
- Each row in nyt_data.txt corresponds to a single document. It
  gives the index of words appearing in that document and the number
  of times they appear. It uses the format "idx:cnt" with commas
  separating each unique word in the document.
```