CS168, Spring 2015

Mini-Project #2

Due by 11:59 PM on Tuesday, April 14th.

Instructions

Recall that you can work individually or with one partner. If you work in a pair, both partners will receive the same grade.

Name your files so that the course staff can easily figure out the contents of each file.

Detailed submission instruction can be found on the course website (`http://cs168.stanford.edu`) under "Coursework - Assignment" section. If you work in pairs, **only one member** should submit all of the relevant files.

Reminder: No late assignments will be accepted, but we will drop your lowest assignment grade when calculating your final grade.

Part 1: Similarity Metrics

Goal: The goal of this part of the assignment is to understand better the differences between distance metrics, and to think about which metric makes the most sense for a particular application.

Description: In this part you will look at the similarity between the posts on various newsgroups. We'll use the well-known 20 newsgroups dataset.[1] You will use a version of the dataset where every article is represented by a bag-of-words — a vector indexed by words, with each component indicating the number of occurrences of that word. You will need 3 files: `data50.csv`, `label.csv`, and `group.csv`. In `data50.csv` there is a sparse representation of the bags-of-words, with each line containing 3 fields: `articleId`, `wordId`, and `count`. To find out which group an article belongs to, use the file `label.csv`, where for `articleId` i, line i in `label.csv` contains the `groupId`. Finally the group name is in `group.csv`, with line i containing the name of group i.

We'll use the following similarity metrics, where $\mathbf{x}$ and $\mathbf{y}$ are two bags of words:

- Jaccard Similarity: $J(\mathbf{x}, \mathbf{y}) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$.
- L_2 Similarity[2]: $L_2(\mathbf{x}, \mathbf{y}) = -||\mathbf{x} - \mathbf{y}||_2 = -\sqrt{\sum_i (x_i - y_i)^2}$.
- Cosine Similarity: $S_C(\mathbf{x}, \mathbf{y}) = \frac{\sum_i x_i \cdot y_i}{||\mathbf{x}||_2 \cdot ||\mathbf{y}||_2}$.

Note that Jaccard and cosine similarity are numbers between 0 and 1, while L_2 similarity is between $-\infty$ and 0 (with higher numbers indicating more similarity).

(a) Make sure you can import the given datasets into whatever language you're using. For example, if you're using python, read the `data50.csv` file and store the information in an appropriate way. Remember that the total number of words in the corpus is huge, so you probably want to work with a sparse representation of your data (e.g., you don't want to waste space on words that don't occur in a document). If you're using MATLAB, you can simply import the data using the GUI.

[1] `http://qwone.com/~jason/20Newsgroups/`

[2] While we typically talk about L_2 *distance*, to make sure that a higher number means a higher similarity we negate the distances.

(b) Implement the three similarity metrics described above. For each metric, prepare the following plot. The plot will look like a 20×20 matrix. Rows and columns are indexed by newsgroups (in the same order). For each entry (A, B) of the matrix (including the diagonal), compute the average similarity over all ways of pairing up one article from A with one article from B. After you've computed these 400 numbers, plot your results in a heatmap. Make sure that you label your axes with the group names and pick an appropriate colormap to represent the data: the rainbow colormap may look fancy, but a simple color map from white to blue may be a lot more insightful. Make sure to include a legend.

(c) Based on your three heatmaps, which of the similarity metrics seems the most reasonable? Are there any pairs of different newsgroups that are very similar? Would you have expected these to be similar?

(d) Next, let's look at an alternative way to compute each entry (A, B) of the matrix. First, for each article a_1, find the article a_2 from a different newsgroup that has the largest Jaccard similarity with a_1. (Just use the straightforward brute-force search algorithm to do this.)

Now, for each pair of different groups (A, B), count how many articles in group A have their nearest neighbor in group B. (Diagonal entries (A, A) are defined to be 0.) Plot these results in a heatmap.

(e) Your plot for part (b) was symmetric, but for part (d) it was not. Why is this the case?

(f) In your plot for (d), which groups seem the most similar? How does the answer compare to that for the plots in part (b)?

Deliverables: All of your code. Three heat maps for (b), one heat map for (d). Your explanations for (c), (e) and (f).

Part 2: Dimension Reduction

Goal: The goal of this part is to get a feel for the trade-off in dimension reduction between the quality of approximation and the number of dimensions used.

Description: It probably took a non-trivial amount of time to run the code in the previous part (hopefully at most a minute or two). In this part we will implement a dimension reduction technique to reduce the running time.

In the following, k refers to the original dimension of your data (i.e., the number of components per data point), and d refers to the target dimension.

- *Random Projection:* This algorithm is given a set of k-dimensional vectors $\{v_1, v_2, \ldots, v_n\}$. It proceeds by picking a $d \times k$ matrix M, drawing each entry randomly (and independently) from a standard Gaussian distribution (i.e., normally distributed with mean 0 and variance 1). The d-dimensional reduced vector corresponding to v_i is given by the product Mv_i. Equivalently, the matrix M can be thought of as a set of d random k-dimensional vectors $\{w_1, \ldots, w_d\}$ (the rows of M), with the jth component of the reduced version of a k-dimensional vector v the inner product between v and w_j.

(a) Implement the random projection algorithm above. For each of the four choices $d = 10, 25, 50, 100$ for the target dimension, redo the computation and plot from part 1(d) on the corresponding reduced vectors. *Use the cosine similarity measure.*[3] Also record the wall-clock time for each of these runs. Is there are a "sweet spot" value of d, where the results appear accurate and the computations are fast?

(b) We will now look at how much the distances are distorted. Take article 3 (which is in the group `alt.atheism`), and for each dimension $d = 10, 25, 50, 100$, create a scatter plot where each point represents an article, with the x coordinate representing the cosine similarity to article 3 in the original full-dimensional space, and the y coordinate representing the cosine similarity to article 3 in the reduced d-dimensional space. Discuss the plots. (For example, does the error from the dimension reduction seem to be an additive term or a multiplicative term?)

Deliverables: Code, plots, times, and discussion for part (a). Plots and discussion for part (b).

[3] We're switching the metric because cosine similarity tends to behave better than Jaccard similarity when you use random projection.