CS 276A Practical Exercise #2

Assigned: Tuesday, November 16, 2004

Due: Tuesday, November 30, 2004 by 11:59 p.m.

Review session: Friday, November 19, 2004, 3:15-4:05 p.m. in Gates B01

Delivery: All students should submit their work electronically. See below for details.

Late policy: Refer to the course webpage.

Group work: We encourage you to work in pairs for this assignment. Teams of two should only submit one copy of their work. *Please note that the requirements for teams of two are slightly more demanding.*

Overview

In this assignment, you will conduct an exercise in unsupervised classification by producing clusters of Usenet newsgroup messages. We provide you with a data set and starter code that reads in the messages and converts them into normalized tf.idf vectors. Your task is to cluster the messages by first employing a dimensionality reduction technique, then running a clustering algorithm such as k-means. The provided code will also evaluate the success of your clustering by computing the purity of your clusters.

This assignment might be a bit more challenging than practical exercise #1, but it should also be more conceptually interesting. It also offers greater opportunity for creativity and extra credit.

Before you start

This assignment requires competence in both Java and linear algebra. If you're not comfortable with either of those, we strongly recommend that you find a partner who is.

You should probably complete the assignment using one of the Leland Unix machines such as the elaines or trees. You're free to use any platform you choose – but the course staff can't promise any help with non-Unix platform-specific difficulties you may encounter.

Supplied files

The starter code is located in /usr/class/cs276a/pe2. Make a new directory and copy everything into it.

Data set: We're using the 20 Newsgroups collection, an archive of 1000 messages from each of 20 different Usenet newsgroups. The version we selected for this assignment excludes cross-posts (messages that were posted to more than one of the 20 newsgroups), so the actual number of messages is slightly below 1000 per group. For more information, please visit http://people.csail.mit.edu/people/jrennie/20Newsgroups/. The data is located in /afs/ir/data/linguistic-data/TextCat/20Newsgroups/20news-18828/, with each newsgroup in a subdirectory containing one file per message. Don't bother making your own copy of it – your program can just read it from its current location. To avoid the possibility of running into memory and CPU limitations, we're going to limit ourselves to 100 messages per group.

Files:

- english.stop: list of stop words
- Stemmer.java: Porter stemmer implementation
- messages2vectors.java: Reads in the messages (100 per group) to produce vectors. Creates separate features for the subject and body fields, employs the stop words and the Porter stemmer and calculates tf.idf values, then normalizes each vector and outputs a file containg the vectors, plus a separate file with the newsgroup labels for those vectors. You shouldn't need to modify this code.
- clusterer.java: Reads in a file of vectors and its corresponding label file, then performs clustering and calculates the purity values. Contains a partial implementation of k-means that you can use if you like.

What you have to do

You have two main tasks: dimensionality reduction and clustering.

Clustering tends to be more successful (not to mention faster) in lower-dimensional spaces. Even with stop words and stemming, our document vectors exist in a space with tens of thousands of features. Thus some type of dimensionality reduction is essential. Two popular approaches to this problem are random projection and singular value decomposition (SVD). You may choose either of these strategies (or any other legitimate dimensionality reduction approach). Whichever one you choose, we request that you project your vectors into a 200-dimensional space. Please do not expend effort trying to determine the optimal number of dimensions, as this is beyond the scope of the assignment.

To perform random projection, you can implement a standard orthogonalization algorithm to generate your projection matrix. For SVD, you can use the JAMA package (http://math.nist.gov/javanumerics/jama/) or Matlab. We will distribute a separate handout on dimensionality reduction and we will cover both random projection and SVD in more detail at the review session on Friday.

Once you've flattened your documents into 200 dimensions, you need to cluster them. You can employ any clustering algorithm you like; we recommend k-means (lecture 13) or agglomerative clustering (lecture 14). Though we would like you to be generally aware of efficiency concerns in writing your code, we will not penalize you for choosing an inherently less efficient algorithm (read: you don't have to do k-means).

Since the dataset can also be partitioned into broader topic groups (see the 20 Newsgroups link above), you should try clustering with both k=6 and k=20. And to test whether dimensionality reduction is actually beneficial, you should also trying clustering the original, non-reduced vectors.

Some considerations for your clustering algorithms:

- The initial seeding step of k-means, in which you choose the centroids for the first iteration, is essential to producing a good clustering.
- Though it is supposed to be guaranteed to converge, k-means occasionally seems to get stuck oscillating between two nearly identical points. You may want to detect this in your code or set a maximum number of iterations (100 is plenty).
- Clusters that are very large or very small are not generally desirable. After your clustering algorithm completes, you may want to "massage" the clusters by somehow eliminating these outliers. For example, if among your k clusters you have one very large cluster, you could split it into two, then take the members of your smallest cluster and redistribute them among the other clusters – thus preserving the total number of clusters at k.
- Agglomerative clustering presents various options for measuring intercluster distances, including centroid-centroid, single-link, complete-link and average-link. Refer to slides 11-22 of lecture 14.

Write-up and deliverables

In addition to your code, please submit a *brief* report (1-2 pages) summarizing your work. Describe what you implemented and report the success of your clustering effort for both k=6 and k=20, with and without dimensionality reduction. If you tried multiple approaches, briefly compare/contrast them. Be sure to mention anything you did that merits extra credit.

Your submission should include:

- any code necessary to compile your program
- your write-up in Word or PDF format (please call it results.pdf or results.doc)
- if necessary, a README file that lists any special information about your code that we should know

Make sure your code compiles cleanly on Leland computers! When you're ready to submit, change to the directory where all of your files are located and run the submission script: /usr/class/cs276a/bin/submit-pe2.

Grading criteria

To earn full credit on this assignment, a group of two must:

1. implement two reasonable dimensionality reduction algorithms
2. implement a reasonable clustering algorithm (if you choose k-means, this must include an initial seeding step that is more sophisticated than random selection)
3. produce clusters with k=6 and k=20, with each of the dimensionality reduction techniques and without any dimensionality reduction
4. achieve decent purity measures

Students working alone only have to implement one type of dimensionality reduction.

Extra credit will be awarded for exceeding the above requirements by:

- trying additional dimensionality reduction techniques such as principal component analysis or another random projection algorithm
- trying multiple clustering algorithms
- enhancing your clustering algorithm with features such as a particularly clever initialization or a "massaging" step
- achieving especially good purity

If you have other ideas that you think might be worthy of extra credit, please ask the course staff before investing your time and effort.

This assignment designed by Louis Eisenberg with assistance from Dan Gindikin, Chris Manning and Prabhakar Raghavan.