CS 493: Algorithms for Massive Data Sets	**Similarity Estimation (cont.)**
April 11, 2002	**Scribe: Jason Perry**

1 Experimental Results for Similarity Estimation

First, let's review our basic definition of similarity. Given two sets A and B, the similarity is defined as:

$$\frac{|A \bigcap B|}{|A \bigcup B|}$$

Clearly, this is a number between zero and one. (There is a heuristic mapping from documents to sets.) We are concerned with performing efficient approximate clustering or nearest neigbor searches via this statistic. In the last lecture, we looked at similarity-preserving hash functions. Now, we want to briefly look at three papers that give experimental results from implementations of similarity estimation schemes.

1.1 Content-based Web Clustering

"Syntactic Clustering of the Web" [1] describes an experiment in clustering web pages based on content.

The document-to-set mapping is done by first removing the HTML from a web page and then defining a sliding window ("shingle" in the paper) of a fixed number of consecutive words. The document is defined as the set of all such windows.

In the paper, a 10-word window size is used, and a subset of 100 windows is selected for each document. Each window is hashed to a 40-bit fingerprint. We want to compare this 100-element set for each pair of documents. For example, if we wish to find documents that are 50% similar, we check to see if the sets for two documents have more than 50 elements in common.

The authors did this by expanding the set for each into a list of $\langle hashvalue,\ documentID\rangle$ pairs. The whole list of hash value-document pairs was then sorted. From this list, they produce a second list (with repetitions) of all $\langle documentID,\ documentID\rangle$ pairs where the two documents had a common hash value. Unfortunately, this causes a quadratic blowup in storage. Finally, they sorted this second list and counted the number of appearances of each document-document pair. The output consists of all pairs that appear in the list more than 50 times.

The experiment was conducted on a dataset of 30 million HTML files (about 150 gigabytes.) The similarity information is used to build clusters by build a graph, with documents as vertices and edges between documents with at least 50% similarity. The connected components of the graph are the clusters.

In the paper, many optimizations to this method are described, including eliminating the most common sequences (producing fewer bogus similarities) and removing exactly identical documents. These also help alleviate the problem of quadratic blowup. The final process took 10.5 CPU days.

1.2 Anchor-based Web Clustering

This paper is called "Scalable Techniques for Clustering the Web" [2]. Here, the mapping from documents to sets is not based on syntactic data. Instead, the "anchor text" from links to the document is used. The set used to represent the document is the multiset of occurrences of words near a hyperlink to the page. TFIDF is used to scale the word frequencies.

The scheme used is similar to what we have already seen, and it uses locality-sensitive hashing. For each set, 80 hash signatures are produced and three are randomly chosen. The concatenation of these three is treated as the hash value.

Using the same definition of similarity, the authors looked for 20% similarity in their dataset. $(\frac{1}{5})^3 = 125$ hash values are obtained (125 different random sets of three.) So we expect one of the values to match between documents that have 20% similarity.

As in the above paper, the set of vectors of 125 hash values is sorted. The output is all pairs of urls that agree in *at least one* location. They also performed a post-filtering stage, going back and verifying that all positives had at least 20% similarity in the 80 initial hash values. To build clusters, instead of connected-component clustering they used center clustering.

The final calculation took about 278 hours on a dual-Pentium machine for a set of 12 million web pages, for which anchor information was extracted for 35 million pages. This was shrunk to a subset of 20 million.

1.3 Similarity Search with Similarity-preserving Hash Functions

The third and final paper is called "Similarity search in high dimensions via hashing" [3]. The authors designed approximate nearest-neighbor data structures using locality-sensitive hash functions. They performed experiments on two datasets. The first was a set of 20,000 histograms of color images, with each histogram represented as a point in 64-dimensional space. The second dataset contained 270,000 points in 60-dimensional space, representing texture information from arial photographs.

2 Dimension Reduction

Now we examine a more rigourous notion of the general problem we have been addressing: The problem of making large objects into small objects that retain the desired properties of the original. Dimension reduction is the problem of representing a set of n points in

d-dimensional space by a set of points in k-dimensional space with minimum distortion in the distance between any pair of points.

2.1 The Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss Lemma is a fundamental result in the problem of dimension reduction that is. This lemma was proven in [4], and a much simpler proof was given in [5]. We will see an elementary probabilistic proof given by Dasgupta and Gupta in [6].

Theorem 2.1 *(The Johnson-Lindenstrauss Lemma)*

We are given n points in $\mathbf{R^d}$. Distances between points are measured by the Euclidean distance, $d(u,v) = \sqrt{\sum_{i=1}^{d}(u_i - v_i)^2}$. There is a mapping of these points $f : \mathbf{R^d} \to \mathbf{R^k}$ such that

$$(1-\epsilon)\|u-v\|^2 \leq \|u'-v'\|^2 \leq (1+\epsilon)\|u-v\|^2$$

when $k = O(\frac{\log n}{\epsilon^2})$.

In other words, distances can be preserved within $(1 \pm \epsilon)$ using only a logarithmic number of dimensions. For our proof, we have $k \geq \frac{4}{(\epsilon^2/2 - \epsilon^3/3)} \ln n$. This is better than the bound in the proof by Frankl and Maehara.

The method of the dimension reduction is simple and practical enough to use in real applications. Given the n points, just take a projection onto a random k-dimensional subspace of $\mathbf{R^d}$. This can be done by randomly rotating the d-dimensional space and randomly choosing k of the basis vectors, or by randomly generating k orthogonal vectors (which can be done by generating k vectors sequentially and taking the orthogonal component of each one.)

Proof. We consider the vector induced by two points u, v. The length of the vector is the distance between the points. Since projection is a linear operation and the $(1+\epsilon)$ factor scales, we only have to consider unit vectors. So our proof will essentially go as follows: Pick a random n-dimensional unit vector, and project it onto the first k coordinates. Show that the squared length of this vector is sharply concentrated about its mean. Then, use the union bound to get the result for n points.

First, what is the expected squared length of our projection? Choose the random vector Y by picking n normal random values and normalizing the vector

$$Y = \frac{1}{\|x\|}(x_1, x_2, \cdots, x_n)$$

Let Z be the projection of Y on the first k coordinates, and $L = \|Z\|^2$. Then clearly $E[L] = \frac{k}{d}$.

Now we have to show that L is tightly concentrated around $\frac{k}{d} = \mu$. We have the following two tail-bound lemmas:

Claim 2.2 *If $\beta < 1$ then*

$$\Pr[L \le \beta k/d] \le \beta^{k/2}\left(1 + \frac{(1+\beta)k}{(d-k)}\right)^{(d-k)/2}$$

which is approximately $\exp(\frac{k}{2}(1-\beta+\ln\beta))$.

Claim 2.3 *If $\beta > 1$ then*

$$\Pr[L \ge \beta k/d] \le \beta^{k/2}\left(1 + \frac{(1+\beta)k}{(d-k)}\right)^{(d-k)/2}$$

which clearly is also approximately $\exp(\frac{k}{2}(1-\beta+\ln\beta))$.

For now, assume these claims. Let v_i' and v_j' be the projections of some v_i and v_j into the k-dimensional subspace. Then $L = \|v_i' - v_j'\|^2$ and $\mu = (k/d)\|v_i - v_j\|^2$. By the first claim we get

$$\Pr[L \le (1-\epsilon)\mu] \le \exp\left(\frac{k}{2}(1-(1-\epsilon)+\ln(1-\epsilon))\right)$$

Then, using the approximation $\ln(1-x) \le -x - x^2/2$,

$$\begin{aligned}&\le \exp\left(\frac{k}{2}(\epsilon - (\epsilon + \epsilon^2/2))\right) = \exp\left(-\frac{k\epsilon^2}{4}\right)\\ &\le \exp(-2\ln n) = 1/n^2.\end{aligned}$$

Now for the upper bound:

$$\Pr[L \ge (1+\epsilon)\mu] \le \exp\left(\frac{k}{2}(1-(1+\epsilon)+\ln(1+\epsilon))\right)$$

Here, we can use the inequality $\ln(1+x) \le x - x^2/2 + x^3/3$:

$$\begin{aligned}&\le \exp\left(\frac{k}{2}(-\epsilon + (\epsilon - \epsilon^2/2 + \epsilon^3/3))\right) = \exp\left(-\frac{k(\epsilon^2/2 - \epsilon^3/3)}{2}\right)\\ &\le \exp(-2\ln n) = 1/n^2.\end{aligned}$$

So a given pair of points is 'bad' with probability $1/n^2 + 1/n^2 = 2/n^2$. There are $\binom{n}{2}$ pairs, so $\Pr[\text{one pair is bad}] \le n(n-1)/2 \cdot 2/n^2 = (1-1/n)$. We can repeat the projection $O(n)$ times to get any constant probability of success. ■

2.2 Proofs of Tail Bounds

Now we will begin to prove the tail bounds we used in the proof. We won't finish in this lecture.

Recall that the coordinates selected for our random vector Y were normal random variables:

$$Y = \frac{1}{\|x\|}(x_1, \ldots, x_d),$$

and Z is the projection of Y on the first k coordinates. $L = \|Z\|^2$.

Now we'll try to find $\Pr[L \le \beta \frac{k}{d}]$. This is equivalent to

$$\begin{aligned} &\Pr[(x_1^2 + \cdots + x_k^2) \le \frac{\beta k}{d}(x_1^2 + \cdots + x_d^2)] \\ =\, &\Pr[k\beta(x_1^2 + \cdots + x_d^2) - d(x_1^2 + \cdots + x_k^2) \ge 0]. \end{aligned}$$

Now we use the "Chernoff trick": to bound $\Pr[x \ge 0]$, look at e^{tx}. By Markov's inequality, $Pr[e^{tx} > 1] \le \mathrm{E}[e^{tx}]$. Continuing on:

$$\begin{aligned} &= \Pr[\exp(t(k\beta(x_1^2 + \cdots + x_d^2) - d(x_1^2 + \cdots + x_k^2))) > 1] \\ &\le \mathrm{E}[\exp(t(k\beta(x_1^2 + \cdots + x_d^2) - d(x_1^2 + \cdots + x_k^2)))] \end{aligned}$$

The random variables are all independent, so we can multiply expectations:

$$= \mathrm{E}[e^{t(k\beta - d)x^2}]^k \mathrm{E}[e^{tk\beta x^2}]^{d-k}$$

The x_i are normal, i.e. their distribution is $(1/2\pi)e^{-x^2/2}$. This gives the nice situation where $\mathrm{E}[e^{tx^2}] = 1/\sqrt{1-2t}$. So, with a change of variables, we can get

$$= (1 - 2tk\beta)^{-(d-k)/2}(1 - 2t(k\beta - d))^{-k/2},$$

as long as $-\infty < t < \frac{1}{2}$. We can choose t to minimize this expression and get the right bound.

References

[1] A.Z. Broder, S.C. Glassman, M.S. Manasse, and G. Zweig, "Syntactic Clustering of the Web," *www6*, available at http://www.scope.gmd.de/info/www6/technical/paper205/paper205.html.

[2] T.H. Haveliwala, A. Gionis, and P. Indyk, "Scalable Techniques for Clustering the Web," *WebDB*, available at http://dbpubs.stanford.edu/pub/2000-23.

[3] A. Gionis, P. Indyk, and R. Motwani, "Similarity Search in High Dimensions via Hashing", *The VLDB Journal*, 518-529, 1999.

[4] Johnson, W. B. and Lindenstrauss, J. "Extensions of Lipschitz mappings into a Hilbert space". *Contemporary Mathematics* **26**, 189-206, 1984.

[5] P. Frankl and H. Maehara, "The Johnson–Lindenstrauss lemma and the sphericity of some graphs. *Journal of Combinatorial Theory Series B*, 44:355–362, 1988.

[6] S. Dasgupta and A. Gupta, "An elementary proof of the Johnson-Lindenstrauss lemma", Tech. Rep. TR-99-06, Intl. Comput. Sci. Inst., Berkeley, CA, 1999.