Final Paper: MOL/COS 551
Jordan Boyd-Graber, Adrian de Froment, Alex Golovinskiy, Jesse Levinson
January 11, 2005

Gene Sketching

The central character of modern bioinformatics is the analysis of data to find patterns that represent relevant biological information. The success, however, of this field may one day lead to serious problems as too much data inundates the methods and resources traditionally used. A world where every individual has his or her genome mapped and stored in a central medical database might lead to more information than even future computers following the trajectory of Moore's law could handle.

Other fields have experienced similar problems with information overload and have developed interesting solutions. One such problem is the task of searching for similar images from a large repository of pictures. These images, which are essentially just arrays of color intensity, are inefficient representations of their contents. If we are only interested in the differences between two images, the only information we want to store is the "distance," however we define it. Using a compact representation of the information that attempted to only retain the distance between two images, [Lv] was able to create an image similarity search system with over a hundredfold decrease in storage size and a commensurate increase in execution time without significantly hampering the accuracy of the comparisons.

This paper outlines the creation and subsequent evaluation of a novel compression scheme for biological information motivated by the success of similar compact representations. After outlining the theory behind our approach and the tools and techniques used to implement it, we will present an analysis of the effectiveness and accuracy of the methodology for typical biological tasks such as finding similar subsets of genes and predicting gene ontology. We then present some possible extensions.

Theory

The sketching technique implemented here is part of a set of methods called metric embeddings: mappings from a complex metric space to a simpler one that preserve the distance metric. Sketching, introduced in [Lv] for an image similarity search, maps from a real-number vector space into a space of bit vectors (sketches) in such a way that the hamming distances between sketches estimate some distance function in the original vector space. [Lv] works with sketches of L1 distances, so this is the metric we chose to start with. We need to find some function f: $\Upsilon^n \rightarrow \{0,1\}^m$ such that, given vectors v and w, $d_{L_1}(v,w) \approx d_H(f(v,r), f(w,r))$, where d_H is the hamming distance and r is a random seed common to all sketches in the dataset. Largely following [Lv], we implement f as follows.

We assume for simplicity that the components of the vectors are between 0 and 1, and that the dimensionality of our target bit vector space is m and the dimensionality of the source vector v is n. We do the following m times, for each bit i of the target bit vector:

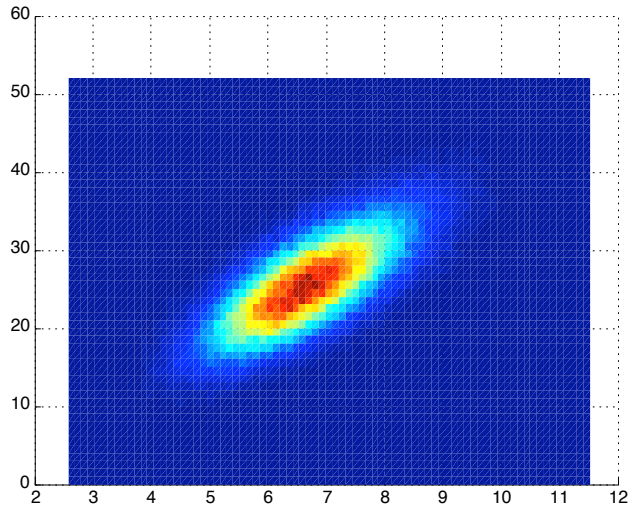
1. Choose a random dimension k from the n dimensions of the source vector space, and choose a random number x between 0 and 1.
2. Mark the ith bit of the sketch as 1 if the kth dimension of v is greater than x

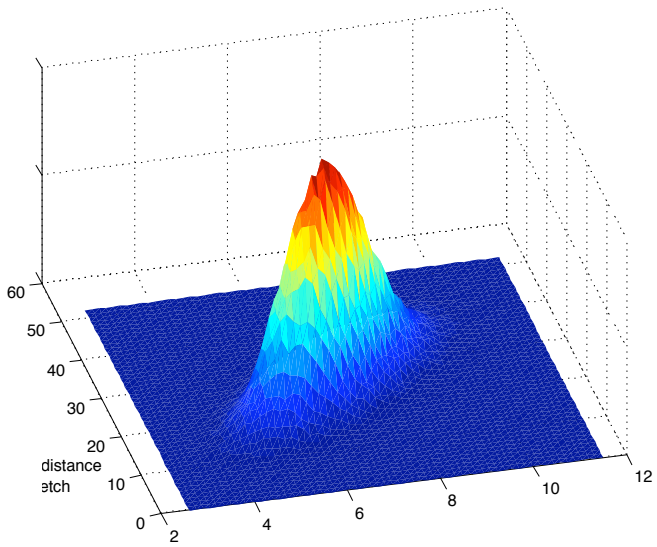
So, we defined a function f that produces a sketch of v. Now let us compare the sketches of two vectors v and w. Consider the first bit of the sketches. If that bit differs between the sketches of v and w, the random number x and dimension k we have chosen must have been between the values of the kth components of v and w. The probability of that happening is proportional to the L1 distance between v and w. Since we have chosen the number x and dimensions k independently m times for each bit of the sketch, the number of bits in which the sketches differ forms a binomial distribution whose mean is proportional to the L1 distance between v and w and the number of dimensions in the sketch m.

Because the number of bits that differ in the sketches is the Hamming distance between the two sketches, our sketching mechanism achieves our original goal: the mean Hamming distance between two sketches (taken over random seeds r) is proportional to the L1 distance between v and w. Since this is a binomial distribution, the error is

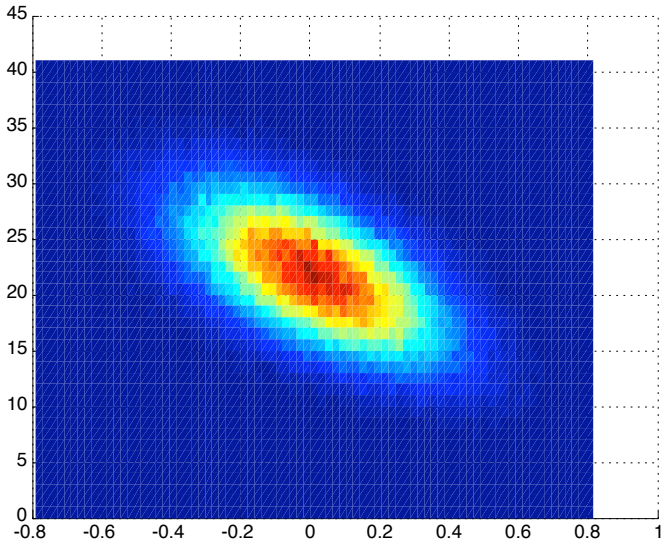
proportional to the square root of m, so the error relative to the mean decreases with the square root of the number of bits used in the sketch. By varying m, the number of bits in the sketch, we can control the tradeoff between space and computational efficiency and accuracy.

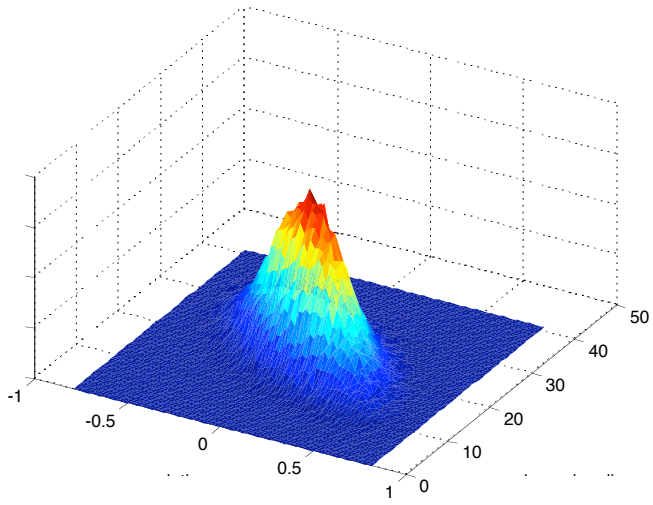
To visualize this distribution, we created a set of vectors chosen uniformly from $[0,1]^n$, created their sketches with a 1:8 compression, and for each pair, compared their real L1 distance with the estimated L1 distance:





What if we were interested in a distance other than L1? We would either have to conjure up a new sketching algorithm, or use the L1 sketcher to approximate our desired distance. As a proof of concept, and since it figures prominently in computational biology, we worked with correlation. We estimated correlation between two vectors with the L1 distance of the mean and variance normalized vectors. This turned out to be effective:





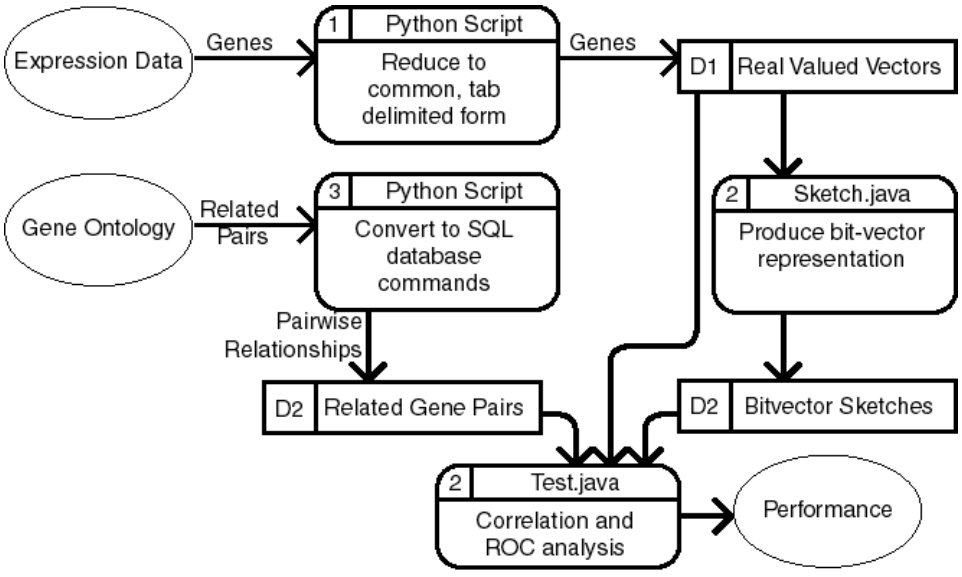
One detail that we have not mentioned is how we select the bounds of the data that are required for the reduction to a sketch. Naturally, the data we work with naturally

is not between 0 and 1, and for the above method we must choose a minimum and maximum bound for each dataset. The tighter the bound is, the more accurate the sketch is, but data outside these bounds will appear as inaccurate artifacts. The bounds ought to be chosen methodically: to include 90% of the data, for example. For our prototype system, however, we just chose the bounds manually after examining the datasets.

Implementation

Because our project focused on developing a new computational method, we could not use an off-the-shelf system and plug in our dataset. We chose to use a database system to provide a level of abstraction between the implementation and the actual information; because of our past experience, the ease of installation, and the relative speed, we chose to use MySQL to store our information.

We used Java to build the infrastructure of the program, as depicted in the data flow diagram (see Figure). We will discuss each of the components in further detail to help describe the system.



We obtained our test information from [Brown] and [Brazma], selecting time

sequence data for salt response and heatshock for S. cervisciae and S. pombe, respectively, and a python script was used to reduce the data to a form that could be easily stored into the database. The database class in our implementation provided methods for retrieving the data as simply an array of real numbers. To help gauge the reliability of the system, we also obtained a classification of gene ontology information. We wanted a simple way of classifying a pair of genes as related or unrelated. Genes with the same seventh-level process annotation were scored with a "1," genes with different seventh-level annotations are scored with a "-1," and if both genes don't have an annotation at that level, the pair is scored with a "0." The original information was obtained from [Myers] and stored in our database system.

Once the data is in the system, the information is converted into a sketch (as described in the previous section). The thresholds and the indices used to generate the sketches (i.e. the random seed) are stored in a data structure that is then associated with the experiment source. Likewise, we are able to store all of the new sketches we generate and retain the information necessary to generate new sketches that we might later create (e.g. to query the nearest neighbors of a new gene). The specifics of how this information is stored are described in Appendix I; the same organization is reflected in the access functions used to retrieve the data.

One might notice that we store both the original information and the sketches we generate. This is an element of redundancy that would not necessarily be in a real implementation. This redundancy enables us to compare the results of various computational methods that use not only the sketches but also the original arrays with a suite of testing software written for this problem context.

Results and Analysis

One way of assessing the viability of our sketching method for reducing the size of genomic expression data is to compare the distances between genes as reported by the original metric (e.g. true L1 distance) with those reported by the sketching metric (e.g. approximation of L1 distance based on bit vector representation). It is worth noting that even the original metric is hardly a perfect indicator of anything *useful*, such as gene

ontology; that is, two genes that are reported to have a small distance (especially if only one set of expression data is used) do not necessarily have any biological similarity, and vice versa. Thus, given the approximate nature of the relationship between distance and biological significance, it is not crucial for the sketch approximation to be identical to the original distance. However, it is reasonable to assume that an approximation of any predictor of ontology can only do worse (on average) than the original predictor, and consequently it is important to test the similarity between the two metrics.

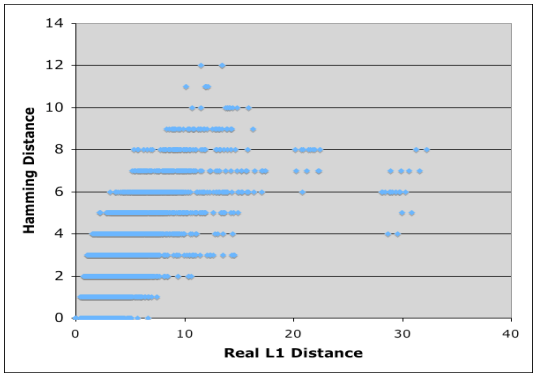
We perform three tests of our system. The first two, run on the S. pombe heatshock data set, compare L1 distances to their sketched approximations and try to find similar genes to a query gene using sketches. The next test, run on the S. cerevisiae salt response data, compares the results of gene similarity (dictated in the GO as above) prediction using sketches as well as the original data.

Distance Similarity

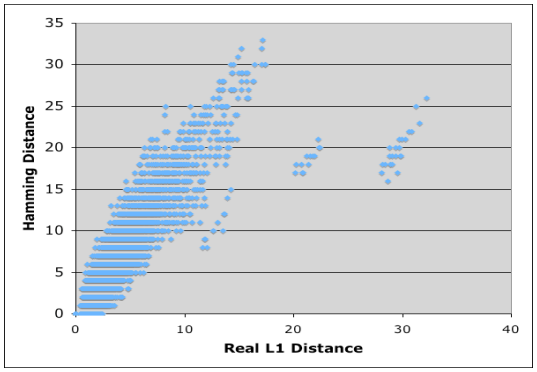
The first set of tests we performed was a comparison of true L1 distance and the sketch approximation distance for a variety of randomly chosen genes from the S. pombe heatshock. Without doing any testing it can be assumed that as the size of the bit vector used by the sketch increases, the correlation between the sketch distance and true L1 distance should increase. The intuition is simple; a higher compression ratio means fewer bits are used to represent the original expression data, and thus the less similar the sketch distance approximation will be from the true distance. On the other hand, if the bit vector is extremely large and very little compression occurs, the two distances should be close.

For this set of tests, thousands of distances from sketches of five different compression ratios were compared to their respective true L1 distances. For each compression ratio, the distance between each of four genes and ~500 other genes was determined and compared to the corresponding true L1 distance. The original, uncompressed expression data for each gene in the experiment used occupied 640 bits (20 data points * 32 bits/point). The compression ratios tested were 32:1 (20 bits), 8:1 (80 bits), 3.6:1 (180 bits), 2:1 (320 bits), and 1.3:1 (500 bits). Following is a plot of Hamming

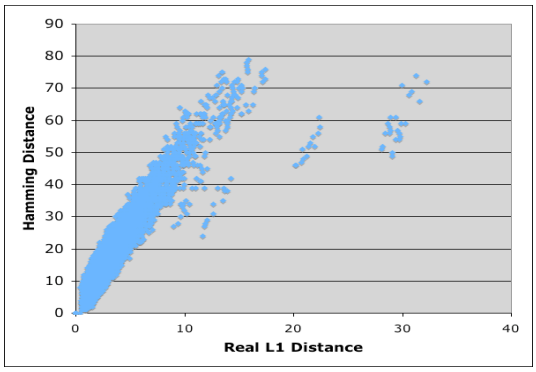
Distance (distance reported by sketch approximation) and real L1 distance for each compression ratio:



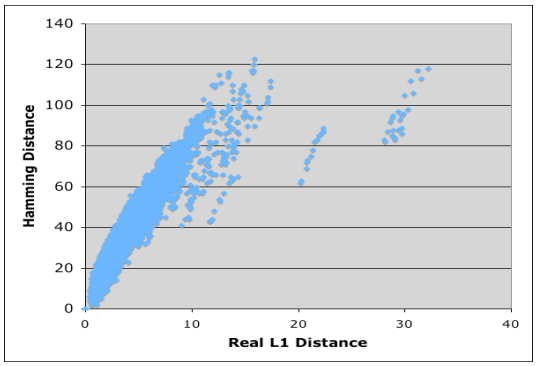
20 bits: r = .70



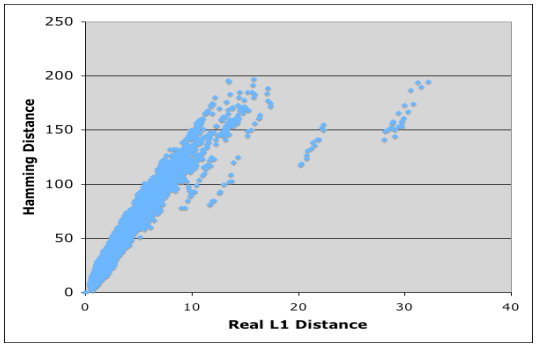
80 bits: r = .85



180 bits: r = .85

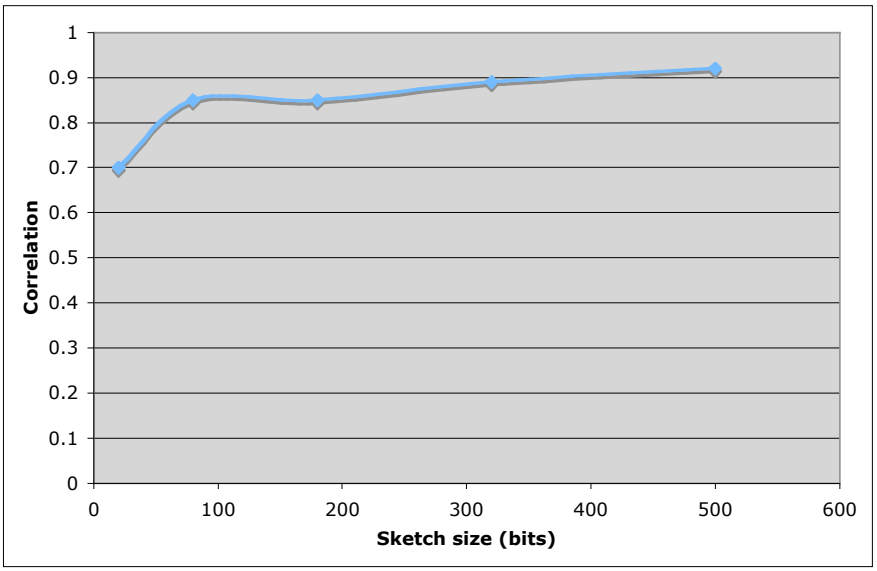


320 bits: r = .89



500 bits: r = .92

The results of this set of tests confirm the theoretical intuition and also demonstrate that sketching offers a reasonable similarity to true L1 distance at meaningful compression ratios.[1] Of course, this is merely a preliminary indication of possible utility; one should not be overly excited by such basic results. As is visually obvious and apparent from the correlation values, the similarity between hamming distance of bit vectors and real L1 distance of uncompressed vectors increases relatively steadily with decreasing compression ratios:



Correlation of Hamming and Real Distance

Observe that a sketch size of 80 bits, representing a 640:80 = 8:1 compression ratio, achieves a correlation of .85 with the true L1 distance. Given that the L1 distance itself is merely a potential predictor of useful information, achieving such a reasonably good similarity to it using 1/8 the amount of space is a notable achievement. That is not to say that other compression techniques are inferior; for example, column replacement or traditional lossless compression techniques may offer similar, or even superior,

[1] The viewer will likely notice small clusters of outliers in these graphs; these are created by the selection of data ranges by the sketching algorithm as noted in the previous section. This is a small issue given the size of the outlying clusters, but perhaps future research could transform the expression data to eliminate or reduce these outliers.

performance in some cases; however, one advantage of our sketching method is that either column replacement or traditional compression techniques could be applied *in addition* to the compressed bit vector. Of course, stacking lossy compression techniques such as sketching and column sampling will undoubtedly result in a loss of accuracy, but there is a good chance that the two methods would be at least somewhat complimentary, and we believe it would be worthwhile for future experiments to examine this possibility.

Finding Similar Genes

While it is encouraging that our sketch approximation of L1 distance offers a reasonably close estimate with significant compression ratios, L1 distance itself is not of any particular use to biologists. That is, a biologist rarely has a pressing desire to find out the L1 distance between the expression data of two genes. However, a real problem biologists are often faced with is to find a gene or group of genes most similar to some particular gene. Given that current gene ontology datasets are limited and that vast amounts of expression data are being created every day, it is certainly useful to be able to efficiently find, from a large data set, a group of genes similar to a query gene based on expression data.

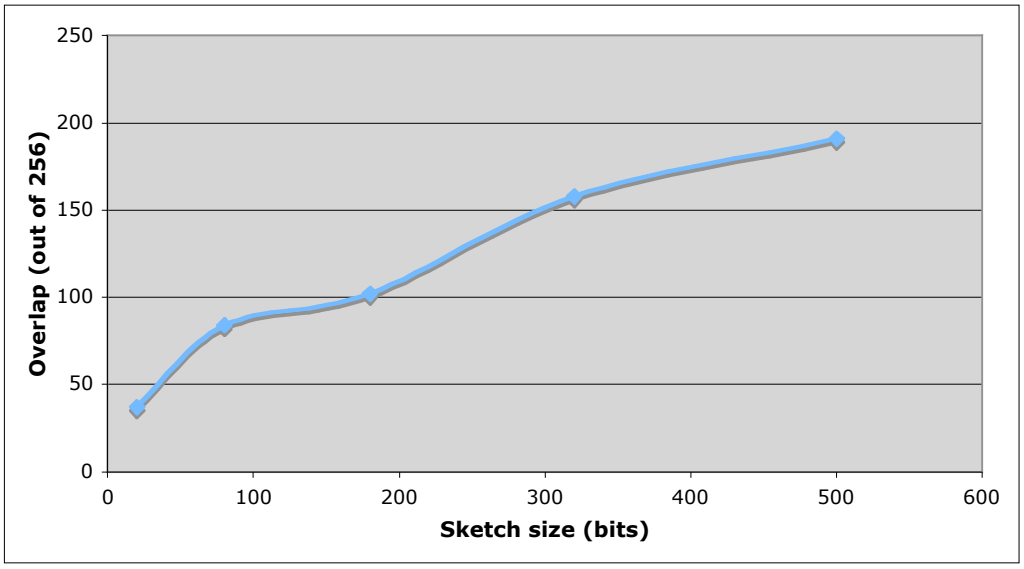
The natural question is how accurate is our sketch approximation method at retrieving genes similar (in terms of expression data) to a query gene? A logical test is to use the sketch method and the true L1 distance method to locate a certain number of similar genes to a query gene, and then to see how much the two groups overlap. The ideal result is to have a complete match between the two groups, while we would expect no better than random selection of genes if no relationship between the two methods exists. Of course, the ideal is never going to be achieved, nor is achieving it exactly particularly important, as L1 distance itself is only an estimation of biological significance. However, we would like to see whether the sketch approximation offers a similarity to the true distance group that far exceeds random probability, even for significant compression ratios.

More explicitly, the test is as follows: given a query gene, locate the k% most similar genes based on L1 expression distance. Then do the same based on the sketch

approximation, and assess the overlap between the two groups. In order to evaluate this question we wrote a program to sort and assess overlap between the two sets of results. The program works as follows:

1. Compute distance between query gene and every other gene
2. Sort genes by distance to query gene
3. Select the k% of genes with the smallest distance to the query gene
4. Perform steps 1-3 for true L1 distance and sketch approximation
5. Compare results by determining overlap between two methods

Specifically, out of the 2650 genes in our dataset, we set out to find the nearest 10% (265) for each of four query genes. Then, we determined how much overlap there was between the two methods. If our approximation were random we would expect only about 26 genes to be common to the two 265-gene groups by chance alone. Here are the results in graphical form:



Overlap between two methods

These are encouraging results. We observe that a sketch size of 80 bits (8:1 compression) generates 84 of 265 matches (over three times more than ~26 expected by chance) and a sketch size of 320 bits (2:1 compression) generates 158/265 matches.

However, we realized that we could harness even further the power of our sketch approximation by taking advantage of its speed increase.[2]

We made two observations. First, in a typical experiment, a sketch similarity search will yield a large number of hits compared to the actual hits of interest to a biologist. The biologist will then retrieve the original data to prune the results more carefully. So, it is not necessary to match identically the hits returned by the original distance function, which is itself an imperfect approximation of some biologically significant factor. Also, our search procedure compares the distances from the query gene to every gene in the database anyway, so it makes sense to return more hits than we will eventually want to work with.

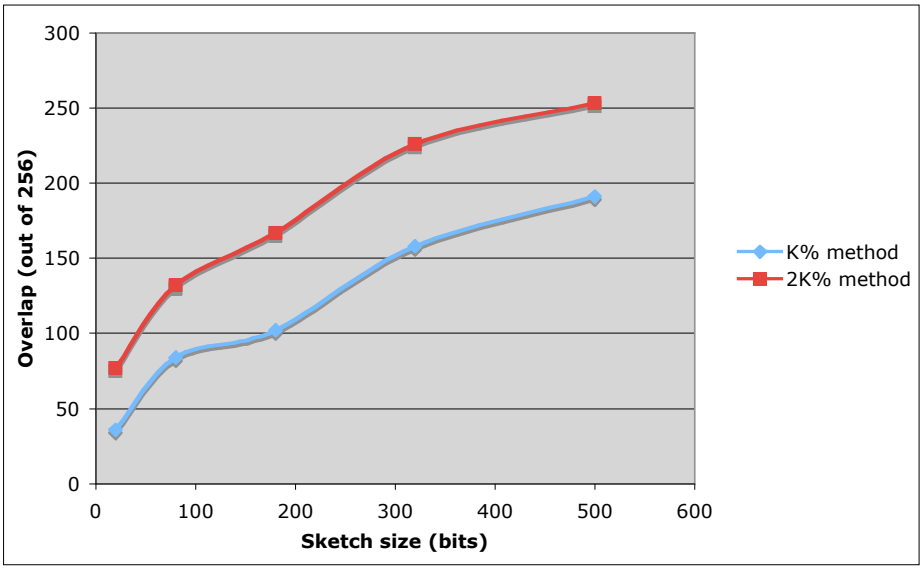
The upshot, then, is that we can utilize the reasonably close similarity of the sketch distance to "weed out" distant genes, and then use true distance to locate the closest. In fact, this can be done in scarcely more time than a purely sketch-based search for small values of k. A simple version is as follows: first, locate the nearest 2k% of genes using the sketch approximation distance, and then use the true L1 distance to narrow down to the nearest k% (that is, compute the L1 distances between the query gene and each of the 2k% reported by the sketch approximation and choose the best half of those; this is the k% we want to analyze). Although at first it might seem that doing two searches is wasteful, for small k it is highly efficient.

The initial first-round search takes the same amount of time regardless of k. Thus, returning the best 2k% matches in the first round takes the *same* amount of time as returning the best k%. Now, if k is small, doing a second-round search, even using true L1 distance, takes almost no time compared to the original search. But in doing so, we hopefully achieve much better accuracy than a simple one-pass k% search.

---

[2] While we did not have the time or resources to optimize the speed of our sketching code at a low level, it is simple to predict, and has been confirmed in the original sketching paper, that sketching not only offers a storage advantage but also a speed advantage over the uncompressed comparison, because the bitwise XOR operation is extremely fast when implemented at a low level. In addition, the smaller sizes of the compressed bit vectors enable more data to fit into CPU cache. Thus, although we do not have empirical evidence for our particular implementation, it is not much of a stretch to assert that sketching offers a significant speed advantage.

To examine whether this idea works in practice we repeated our initial task of locating the best 10% of genes in our 2650 sample space, but this time took the best 20% of matches from the sketch approximation and ran *those* through the true L1 metric. The results were encouraging:



Overlap between two methods using 2k% sketch trick

Observe the significant improvement in overlap as a result of using the 2k% method. Indeed, for a sketch size of 80 bits (8:1 compression) we achieve 132 of 265 matches, compared to just 84 of 265 matches in the original method (which itself isn't bad). And for a sketch size of 320 bits (2:1 compression) we generate 226 of 265 matches instead of 158. Thus we obtain a significant improvement in accuracy with a minimal performance loss for small k.

The benefits of this improvement could likely be extended even further by using more than two rounds, or values greater than 2k for small k. For example, one could use a very small sketch size, such as 16:1 compression, to weed out the worst 75% of hits, leaving the best one-fourth. Now, although the best 25% would likely differ significantly from the "true" best 25%, if we are only ultimately looking for the best 1%, those best 1% are probably almost all in the approximate 25%. We could then use a modest sketch size of 4:1 compression to leave us with just the best 5% of genes, followed by a true L1 comparison to find the best 1%. Those three rounds combined would take far less time

and memory than a complete L1 comparison, and even less time than a one-pass 4:1 compression, while offering nearly the same accuracy as the first and significantly improved accuracy over the second. Indeed, to look at this method another way, if a biologist has a fixed amount of time to conduct a search, using our sketching method will actually *improve* the accuracy of the results because the researcher will not have to settle for an incomplete search. In short, the user of the system has several parameters to vary the tradeoff between accuracy and computational overhead.

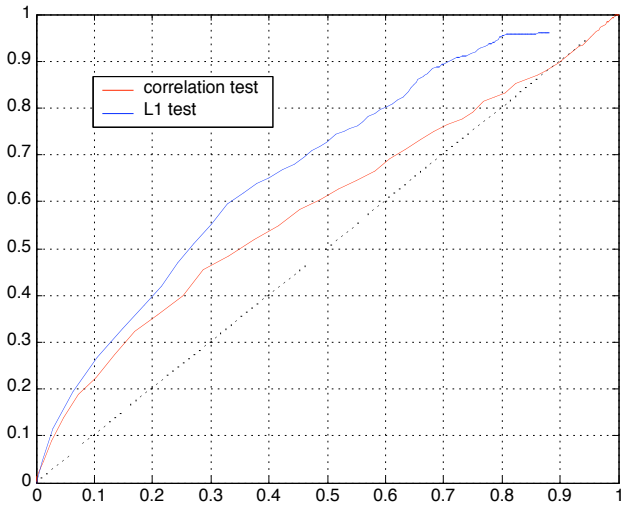
Given the strong correlation of sketch approximation distance and true L1 distance even for significant compression ratios and, in particular, the impressive ability of the sketch approximation when combined with the iterative searching trick to return similar hits to a query gene, we believe that the use of sketches for genomic expression data searches is a strong candidate for materially improving biological searches. Based on the demonstrable success in these preliminary experiments, we recommend further research into low-level optimizations of bit vector comparisons, experiments with very large datasets, and comparison and possible combination with other compression and column sampling techniques.

Gene Ontology Classification

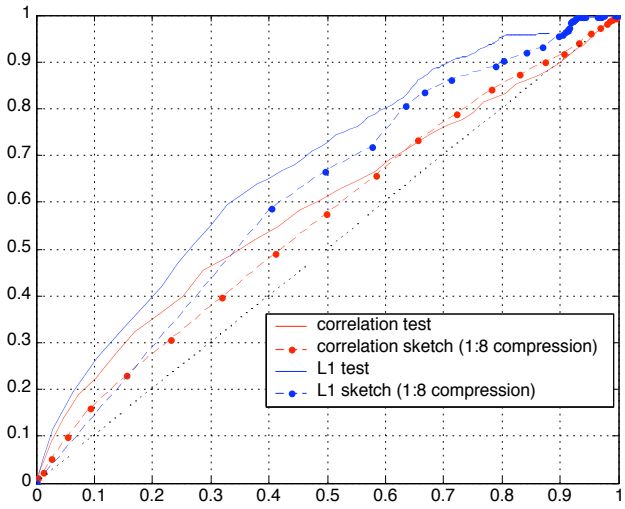
Gene expression experiments are often used to infer biological functionality and classification. One might attempt to predict whether two genes are related in the GO (in the sense described above) by considering their distance. A straightforward approach would be to predict genes to be related if their distance is below some threshold. As we vary the threshold, we vary the sensitivity and specificity of the predictor, and can build a ROC curve to evaluate how accurate our predictions are. This provides a way to check how well sketches preserve this biological information. We compare the accuracy of a predictor that uses sketches with that of a predictor that uses the original information.

We used the heat shock experiment [Brazma] and sampled 600 genes. Of the 600*599 gene pairs, the GO method above yielded 167038 unknowns, 2582 matches, and 10080 non-matches. We only consider the matches and non-matches, and use the L1 distance and correlation. We first test the predictor using the original data; that is, our

predictor calls genes a and b a match if: $d(a,b) < d_{\text{thresh}}$ (greater than for correlation). The resulting ROC curve is:



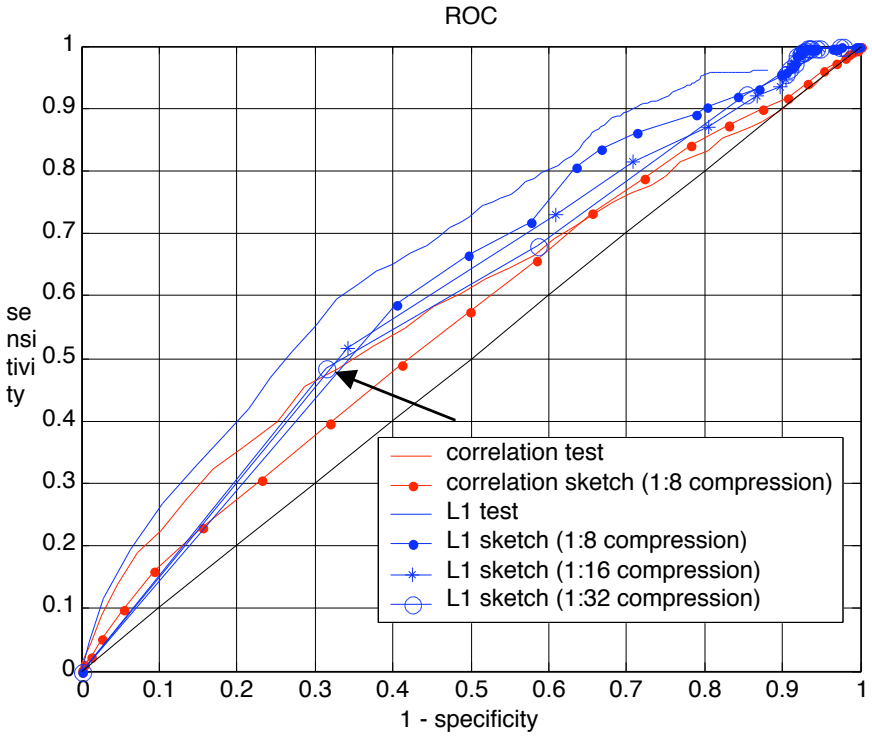
We see that this is not a great but a statistically significant predictor. How does performance degrade as we use sketches? That is, our predictor declares a match when $d_H(sketch(a), sketch(b)) < d_{\text{thresh}}$. We use 1:8 compression as a baseline:



As expected, the sketch predictor is not as good as the original, but it is still statistically significant. Note that we have less control over the sensitivity of the predictor. When we compared original data, we could set this parameter continuously, whereas with sketches, we compare Hamming distances, so we only have a discrete set to

choose from. In the graph above, these discrete points are interpolated linearly for easier visualization. The dashed lines hold no other significance. Note that most threshold values are overly sensitive: that is, the test is meaningful if we set the threshold Hamming distance to a relatively small value.

Focusing on the L1 test, how does accuracy degrade as we increase compression?



We see that accuracy decreases somewhat, but the test still yields meaningful results. One interesting point is the first non-zero point of the 1:32 compression test (the hollow circle pointed to by the arrow). With a specificity of roughly .7, at test at that point produces a sensitivity of roughly .5. Comparing this with the original test, which produces a sensitivity of roughly .6 for that specificity, we see that the decrease in accuracy is not large. This particular predictor simply tests if the two sketches have a Hamming distance of less than 1; that is, it just tests whether the two sketches are identical. To underline the point, the slight decrease in accuracy bought us a change from computing the L1 distance between two vectors to comparing whether two vectors that are 32 times smaller are identical. This trade off, with faster computation and smaller space requirements at a slight cost to accuracy, may be useful for applications such as

estimating biclustering, in which many estimates of distances between genes must be made.

A surprising observation is that the L1 distance seems to be better than correlation at predicting GO similarity. In particular, the point of the L1 sketch test with 1:32 compression described above is a better predictor than the correlation test. Our impression is that correlation is generally held as a reliable metric for such tests and that the L1 distance is rarely mentioned in computational biology. This is something we would like to investigate further.

Extensions and Applications

We think the sketching method offers enough promise to warrant further examination in several directions. An exhaustive analysis of performance and memory usage of sketching applied to a concrete problem on a large database should be performed. Better organization and indexing of the database of sketches would further decrease the computational overhead. Techniques that are more efficient than comparing a query to every sketch in the database can be examined; one particular line of investigation could use techniques from computational geometry to recursively partition the search space to reduce computation time. Also, other distance metrics should be looked into—it is not clear how difficult it is to develop a new sketching technique that preserves some particular distance.

One of the more promising applications of sketching is in aiding heuristic algorithms that rely on distance comparisons.

Biclustering

Biclustering will be an interesting application of the sketching method outlined here. Traditionally, gene expression data are clustered hierarchically according to some measure of correlation between their expression levels across the various different conditions in an experiment (i.e. by rows of the expression matrix only) [Eisen]. If genes cluster together in an experiment (for instance various time-points in the cell cycle), this

can give us information about their possible co-regulation, or allow us to annotate genes of unknown function.

Biclustering improves on a fundamental weakness of the clustering approach as it has been used to date. Genes are clustered by row and column simultaneously, with the output of the algorithm being a list of the biclusters (ie submatrices) that involve each gene. A bicluster corresponds to a subset of the experimental conditions under which a subset of genes' expression correlates significantly. This allows us to reduce the effect of the inherent noise present in microarray data, as among the subset of columns ignored in each bicluster will be noisy and uninformative experimental conditions. It also allows us to integrate data from many different sources. For instance, conditions can be included from several different experiments. Lastly the conditions that may be informative in a given cluster are chosen by algorithm, rather than by the intuition of the experimenter deciding which to include in the experiment [Madeira].

These advantages lead to a greater degree of biological realism. We can use co-membership of biclusters to infer functional annotation of genes of previously unknown function. Membership of multiple clusters is more reflective of systems in which pleiotropy may be important; individual genes may have multiple functions, and participate in multiple processes. The fact that genes are clustered across subsets of conditions represents the context-dependence of gene function. Gene regulation is dynamic, adapted to respond to both endogenous and exogenous changes in the cell's environment. This is achieved in part by the differential binding of transcription factors to regulatory regions under different physiological conditions [Harbison]. Thus some genes may be coexpressed under some conditions, but independent under others, and this is well represented by bicluster analysis.

Our sketching methods may provide an important step towards making biclustering methods more useful. Biclustering is an NP-complete problem, and most algorithms use heuristic methods to identify biclusters [Madeira]. Within this framework, the application of sketching methods to microarray data will allow faster searching, increased accuracy for a given time investment, and will give us the ability to work with larger datasets. This last point is crucial, as it allows us to combine ever more conditions

into single analyses with the object of finding the contexts within which interesting and informative patterns are expressed.

Conclusion

Gene sketching presents a promising means of dealing with the voluminous amount of data bioinformatics researchers will face in the future in applications like biclustering. Our method offers a means to conduct searches comparable to traditional methods with a less onerous investment of time and allows the classification of gene ontology annotations at a rate competitive with the L1 distance and correlation. With optimizations for search and more clever storage of sketches, this algorithm presents an exciting opportunity to conduct previously impossible investigations using bioinformatics data.

Acknowledgements

Chad Myers
Kai Li
Christine Lv
Olga Troyanskaya

References

Brazma, A., et al. ArrayExpress—a public repository for microarray gene expression data at the EBI. *Nucleic Acids Research* (2003) 31: 68-71. [http://www.ebi.ac.uk/arrayexpress]

Brown, A., et al. *S. cerevisiae - EUROFAN II*. MIPS Comprehensive Yeast Genome Database. [http://mips.gsf.de/proj/eurofan/eurofan_2/b2/]

Eisen M., P. Spellman, P. Brown P, D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci U S A* (1998): 95: 14863-8.

Madeira, S. and A. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE Transactions on Computational Biology and Bioinformatics* (2004) 1.1.

Myers, C. [clmyers@princeton.edu]. " Re: Gene Ontology Information." Private e-mail message to Jordan Boyd-Graber, [jbg@princeton.edu]. 7 December 2004.

Harbison, C. et al Transcriptional regulatory code of a eukaryotic genome. *Nature* (2004) 431:99-104.

Lv, Q., M. Charikar, and K . Li. Image Similarity Search with Compact Data Structures. To appear in *ACM 13th Conference on Information and Knowledge Management* (2004) Washington D.C., 2004.

Appendix I: Database Organization

```
mysql> show columns from entry_info;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int(11)      |      | PRI | NULL    | auto_increment |
| exp      | int(11)      | YES  |     | NULL    |                |
| short    | varchar(30)  | YES  |     | NULL    |                |
| ext_info | varchar(200) | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Stores an individual gene's name and ID.

```
mysql> show columns from exp_info;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| exp     | int(11)     |      | PRI | NULL    | auto_increment |
| short   | varchar(30) | YES  |     | NULL    |                |
| numcols | int(11)     | YES  |     | NULL    |                |
| bits    | int(11)     | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Stores the extended information of an experiment that is linked to a collection of genes.

```
mysql> show columns from orig_dat;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| type  | int(11)|      | PRI | 0       |       |
| id    | int(11)|      | PRI | 0       |       |
| value | double | YES  |     | NULL    |       |
+-------+--------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Stores the actual information (i.e. uncompressed) from the experiment linked to a gene.

```
mysql> show columns from sketch;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| id    | int(11)| YES  |     | NULL    |       |
| value | int(11)| YES  |     | NULL    |       |
| idx   | int(11)| YES  |     | NULL    |       |
+-------+--------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Stores the sketch information linked to a gene.

```
mysql> show columns from sketch_info;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| exp   | int(11)|      | PRI | 0       |       |
| value | double | YES  |     | NULL    |       |
| idx   | int(11)|      | PRI | 0       |       |
+-------+--------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Stores the profile needed to recreate a sketch.

```
mysql> show columns from types;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| type  | int(11)     |      | PRI | 0       |       |
| short | varchar(30) | YES  |     | NULL    |       |
```

exp	int(11)		PRI	0	

3 rows in set (0.00 sec)

Stores the column information of the original expression data.

```
mysql> show columns from yeast_go;
```

Field	Type	Null	Key	Default	Extra
gene1	char(20)		PRI		
gene2	char(20)		PRI		
value	int(11)	YES		NULL	

3 rows in set (0.00 sec)

Stores the gene ontology information consisting of pairs of genes mapped to (1,-1); missing information is assumed to be zero.