APPROXIMATING COMMUTE TIMES ON LARGE GRAPHS

Georgina Wilcox, Professor Sanjay Chawla

School of Information Technologies

georgina.wilcox@sydney.edu.au

Summary

1. We demonstrate that the commute time distance can be used for outlier detection.
2. We provide the first implementation of Spielman and Srivastava's nearly-linear time algorithm for approximating the commute time.
3. We show that the approximation is accurate enough to be useful for detecting outliers.

RANDOM WALKS

Let $G = (V, E, w)$ be a weighted graph with n vertices and m edges.

We can define a random walk on the graph as follows: given our current position at a vertex, randomly select one of the neighbouring vertices with probability proportional to the weights of the edges. [2]

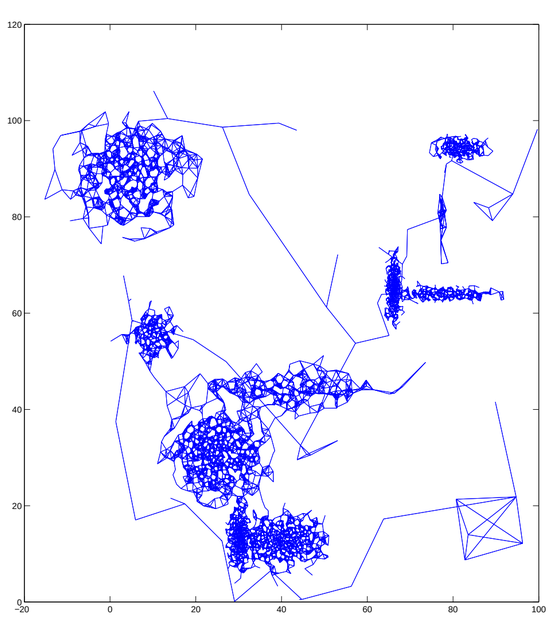
At each step of the walk, if our current vertex is called i, then the probability of selecting a specific neighbour j is

$$p_{ij} = \frac{w_{ij}}{\sum_{k:(ik \in E)} w_{ik}}.$$

Commute time distance

The *commute time* between i and j is the expected number of steps it takes a random walk from i to reach j and return to i.

Figure 1: *A synthetic data set. The commute time distance between any two vertices in the same cluster will be small, but distances to outlying points will be larger.*



Many real-world data sets can be modelled as a weighted graph. The commute time distance allows us to measure the level of similarity between data points in a way that takes into account the **connectivity** of the data set, instead of just the shortest path between two points.

CALCULATING COMMUTE TIMES

An exact computation of the commute times requires the eigenvalues of the Laplacian matrix of the graph.

Calculating the eigenvalues takes time $O(n^3)$, where n is the number of vertices in the graph. **This is infeasible for large graphs.**

A FAST APPROXIMATION

Spielman and Srivastava describe a nearly-linear time algorithm for building a data structure which allows for the commute time between any pair of nodes to be calculated in $O(\log n)$ time. [3]

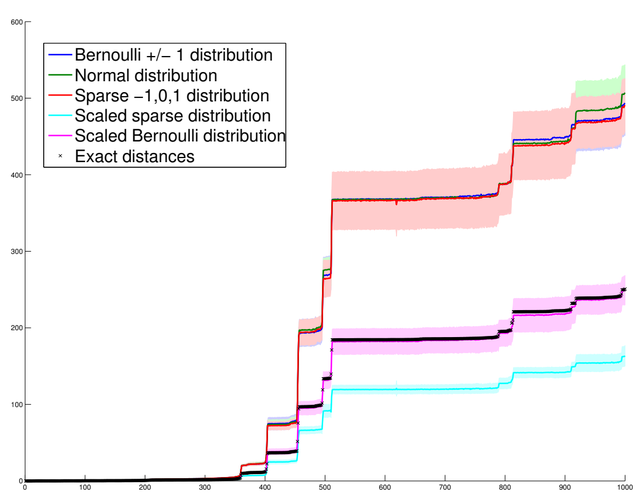
Their algorithm combines the technique of **random projection** with the **Spielman-Teng solver** for linear systems.

Random projection

The Johnson-Lindenstrauss lemma states that pairwise distances between vectors in a vector space of dimension n will be preserved when the vectors are projected onto a subspace spanned by $O(\log n)$ random vectors [3].

Spielman and Srivastava use this fact in order to reduce the dimensionality. We experimented with several different random distributions, as shown in Figure 2. Of particular interest is the sparse distribution, as this allows for greater memory efficiency.

Figure 2: *This plot shows the distribution of pairwise distances for the points in Figure 1. Notice that the approximations are roughly monotone increasing, and also the step-like nature of the distances.*



Fast solvers for linear systems

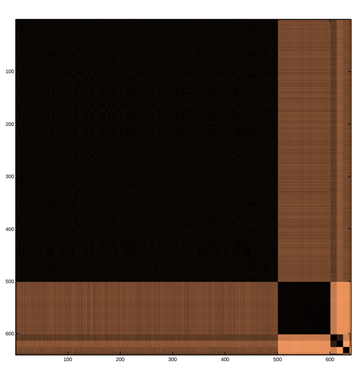
The algorithm also takes into account the special matrix of the Laplacian matrix. The Laplacian matrix for a sparse graph is diagonally dominant and sparse, and we can take advantage of this to efficiently solve equations of the form $y = Lx$, where L is a Laplacian matrix.

COMMUTE TIME DISTANCE FOR OUTLIER DETECTION

Given as input a set of data points, we can use the commute time in conjunction with a known distance-based method for outlier detection in order to find outlying points in the graph. [1]

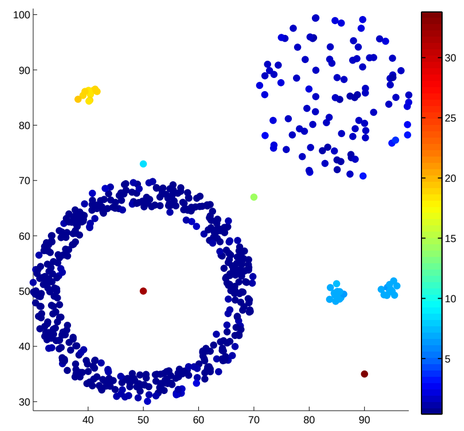
After constructing a weighted (connected) graph as in Figure 5, we use the Spielman and Srivastava approximation to calculate the commute times. The pairwise commute times for all vertices in the graph are visualised in Figure 3.

Figure 3: *Commute times for the data set in Figure*



We use the Top N pruning method to calculate (for each vertex) an *outlier factor* based on its average distance to its k-nearest neighbours. The original data points are shown in Figure 4, highlighted according to outlier factor. "Normal" points are blue, and outlying points are coloured red.

Figure 4: *The outlier detection algorithm on a small synthetic data set in two dimensions. A visual inspection of the results shows the commute time distance has good potential as for identifying both local outliers and outlying clusters.*



INITIAL RESULTS

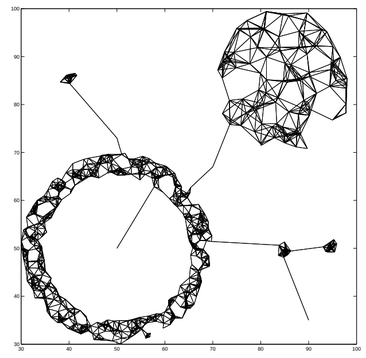
Initial results show that the random projection mostly preserves the ranking of the points. This would be sufficient for an application where the goal was to identify (for example) the top N outliers, and we didn't care about order.

The Spearman rank correlation coefficient is a measure for comparing the correlation of two variables when we are only interested in the ranks.

Table 1: *The table shows the Spearman coefficient of the exact commute times compared to the approximate commute times for various values of k. In this case, k is the dimension of the subspace from the random projection.*

	Spearman correlation coefficient		
k	Top 20	Top 50	Top 100
20	0.58 (0.11)	0.8 (0.063)	0.75 (0.027)
50	0.63 (0.21)	0.84 (0.043)	0.77 (0.017)
100	0.63 (0.18)	0.85 (0.031)	0.77 (0.047)
200	0.81 (0.049)	0.86 (0.031)	0.79 (0.031)

Figure 5: *A weighted graph on 640 vertices with 5300 edges. (Weights are simply Euclidean distances between points in $\mathbb{R}^2$).*



References

[1] Nguyen Khoa and Sanjay Chawla. Robust Outlier Detection Using Commute Time and Eigenspace Embedding. In Mohammed Zaki, Jeffrey Yu, B Ravindran, and Vikram Pudi, editors, *Advances in Knowledge Discovery and Data Mining*, volume 6119 of *Lecture Notes in Computer Science*, pages 422–434. Springer Berlin / Heidelberg, 2010.

[2] László Lovász. Random Walks on Graphs: A Survey, 1993.

[3] Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 563–568, New York, NY, USA, 2008. ACM.