Platonic Graph Inference

Chris Fougner, Jeff Lloyd, Scott Wong (Group 55)

December 10, 2013

1 Introduction of Problem

Networks are extremely useful structures that represent relationships among objects. In general, these objects share many types of relationships. People may be friends or coworkers, companies may share joint ventures or investors, and countries participate in various agreements. Constructing networks with edges represented by any one of these relationships can reveal interesting insight, but potentially ignores useful information from various other relationship networks. Can information about one formulation of a corporate network help with predictions in another? Is there a true, *platonic network seed* that actually captures the essence of the underlying relationships?

In this paper we seek to identify procedures that identify optimal, platonic networks that merge structure from many observed instances of networks. Specifically, our goal is to identify a network of companies from the CrunchBase data that best exemplifies multiple relationships between companies. We suggest an inference procedure that can be widely used to infer platonic seeds and test Kronecker, Bernoulli and low-rank models. We evaluate their performance on synthetic data with known platonic seeds and observe that low-rank models appear to have the lowest out-of-sample reconstruction error. Applying this procedure to the CrunchBase data, we unfortunately cannot conclude that we have identified the platonic seed and hypothesize points of failure and possible improvement.

2 Literature Review

A key component of our goal to infer a platonic network from several observed graphs is determining similarity between graphs. Nowell and Kleinberg explore the link prediction problem in [7], suggesting that network topology and node-node similarity can predict what edges are likely to appear in the future. They attempt to predict edges in the graph by defining similarity metrics between pairs of nodes such as graph distance, Jaccard similarity, preferential attachment, Katz distance, and commute time. Correlation distance, as suggested by [9], intuitively captures the notion of edges in the two graphs behaving similarly. Collectively, the authors successfully test the prediction strength of various link prediction methods, but do not seriously attempt to explain why they perform differentially. Thus, part of the challenge we faced is evaluating which metrics are best when inferring the similarity between two graphs.

With established similarity metrics, we then require a way to infer the platonic graph from our observed graphs. One method of this is the utilization of Kronecker graphs. In [6], Leskovec and Faloutsos describe that Kronecker graphs have many similar features to real-world networks: their in- and out- degree distributions both follow power laws, they exhibit small diameters, and their connections resemble "communities of communities". For a graph G, the researches calculate a Θ that could be the initiator of G, generate a new Stochastic Kronecker graph G' from Θ, and compare G and G'. They do not, however, then calculate the initiator Θ' of G', and compare it to Θ. For platonic graph inference, we wish to evaluate how G and G', analogous to our observed graphs, affect the calculated probability matrices Θ and Θ'.

Lastly, we needed to understand the state of real-world company networks, so that we can tune our random, test graphs to match those we might find in the CrunchBase data. In [2], Battiston and Catanzaro create a network of companies in the US and Italy, where two companies are connected if they share a common board member. Their findings show first that these company-company graphs have high clustering, exhibit power-law degree distribution, and contain one dominant connected component. Perhaps most importantly,

they state that the results derived from each graph are consistent; thus, they argue that directorship graphs are invariant to location and other metrics, and the same properties should appear in other data sets. With an idea of how these companies are connected, we are able to generated synthetic platonic seeds that are representative of the CrunchBase data so as to derive the most meaningful analysis.

3 Graph Distance Metrics

Platonic graph inference seeks to identify a platonic graph that is in some sense "close" or "near" the observed graphs. To formalize this notion we introduce a few graph distance functions to evaluate how close two graphs are to each other[1].

Before discussing the metrics explicitly, we observe that there are two distinct situations in which distances between graphs can be measured. First, "node-matched" graphs are those where there is a known mapping from the nodes of one graph, $G_1 = (V, E_1)$, to the nodes in another graph $G_2 = (V, E_2)$. Second, "unmatched" graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ satisfy the relationship $|V_1| = |V_2|$, but there is no known mapping from the nodes of one to the other.

The universe of possible metrics to consider is quite large, so to narrow the space we identified a few criteria that "good" metrics should satisfy. If one graph is a slight perturbation of another (i.e., it has a few edges added, deleted, or rewired), the distance between them should be small. As the perturbation increases, the distance should increase monotonically. Additionally, small perturbations of the graph should result in a distance smaller than the distance between two independent random graphs from the same parametric distribution (i.e., two independent G_{nm} graphs). Finally, the metrics should be computationally feasible.

3.1 Matched Graphs

For node-matched graphs, we considered

- Frobenius Distance: $\frac{\|A^{(G_1)} - A^{(G_2)}\|_F}{\|A^{(G_1)}\|_F + \|A^{(G_2)}\|_F}$, for $A^{(G)}$, the adjacency matrix of G ,
- 2-Norm Distance: $\frac{\|A^{(G_1)} - A^{(G_2)}\|_2}{\|A^{(G_1)}\|_2 + \|A^{(G_2)}\|_2}$,
- 2-Norm Shortest Paths Distance: $\frac{\|S^{(G_1)} - S^{(G_2)}\|_2}{\|S^{(G_1)}\|_2 + \|S^{(G_2)}\|_2}$, for $S^{(G)}$, the shortest path matrix of G,
- Correlation Distance [9]: $\sqrt{\frac{1}{2}(1 - \rho(G_1, G_2))}$, where $\rho(G_1, G_2) = \frac{\mathrm{Cov}(G_1, G_2)}{\sqrt{\mathrm{Var}(G_1)\mathrm{Var}(G_2)}}$.

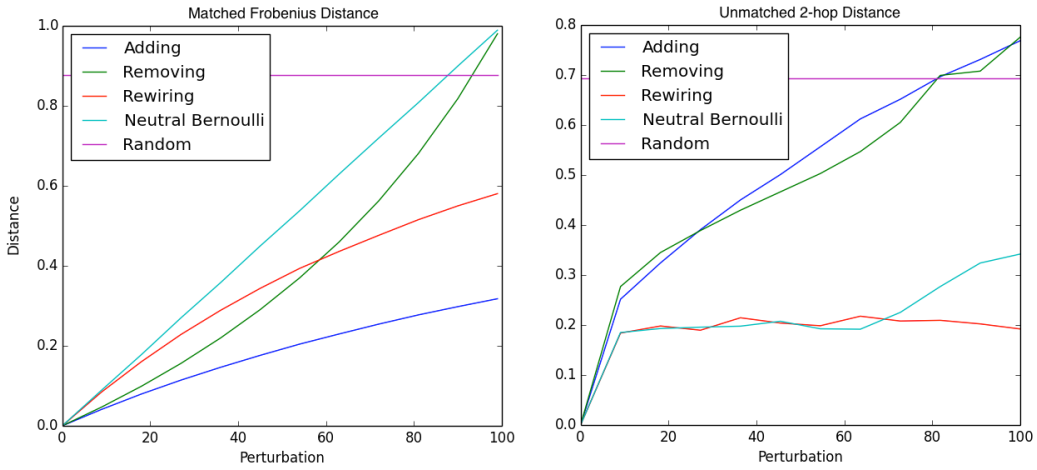
Evaluating these different metrics, the 2-norm distance is not robust to perturbations – small changes in the graph quickly make graphs farther apart than independent random graphs. This issue was further accentuated when using shortest path matrices, which also suffer from computational challenges.

Frobenius distance, being the simplest of the metrics, takes into account the number of edges that are present in one graph, but not the other. Correlation distance intuitively captures the notion of edges in the two graphs behaving similarly. It has the additional advantage of being a true metric and being more robust to different numbers of edges. Experiments showed that correlation distance had a significantly higher computational cost, while offering little improvement over the Frobenius distance.

The left panel of Figure 1 shows how the Frobenius distance behaves as graphs are perturbed. Base preferential attachment graphs of 1,000 nodes and 9,990 edges were constructed, then these graphs were perturbed by adding, removing, rewiring, or flipping various numbers of edges. The distance from the base graph to the perturbed graph was recorded, and the experiment was repeated for a larger perturbation. This entire process was repeated four times to smooth out the curve. In most cases, the magnitude of the perturbation corresponds to the percentage of the number of starting edges that were affected. In the case of "Neutral Bernoulli", the perturbation corresponds to the probability an edge is deleted (new edges were created randomly to keep the expected number of edges constant). Finally, the "random" line shows the expected distance between the base graph and another fresh preferential attachment graph.

[1]Most functions considered do not have all properties of a true metric, and so are technically pseudo-metrics, but we will call them metrics for convenience.

Figure 1: Distance of perturbed preferential attachment graphs with n=1,000 and $m = 9{,}990$. *Left:* Frobenius distance. *Right:* Unmatched 2-hop distance.



3.2 Unmatched Graphs

Given two graphs with different node labelings, the difficulty in finding the optimal matching between nodes is factorial in the number of nodes. Rather than attempting an approximate matching metric, we decided to focus on characteristic properties of the graphs. Well known properties include degree, PageRank, clustering, and eigenvalue distributions.

Given two distributions, one must also be able to compare them. There are three issues:

1. The distributions are rough and jagged, which are difficult to directly compare.
2. Most distributions on networks exhibit superlinear decay away from the mean (direct comparison would therefore ignore any tail behavior).
3. One must define a metric by which two distributions are compared.

To address the first problem, we chose to separate the data into equally spaced bins to smooth the distributions. The second problem was dealt with by taking the logarithm of every bin height. Third, the distributional distance metric we chose is the normalized L_2 vector difference:

$$d(u, v) = \frac{||u - v||_2}{||v||_2 + ||u||_2}$$

Next, we exclude certain metrics because they fail to satisfy our criteria. Computing the full eigenvalue distribution is prohibitively expensive, so we immediately exclude it. Furthermore, the distribution of clustering coefficients was found to be too similar for two completely different graphs from the same model. This was especially a problem for the Watts-Strogatz model. [2] suggests that the Board of Directors networks have Watts-Strogatz characteristics, which makes this metric unviable for our application. This leaves degree distribution and PageRank, which is a scaled version of degree as the jump probability goes to zero.[2] The problem with degree distribution is that any two graphs from the same model will have very similar degree distributions. However, by generalizing the degree distribution, we were able to find an acceptable metric. We define the *m-hop* distribution to be the list of the cardinality of the set of nodes reachable in m hops from a specific node. Using Pythonic list comprehension:

$$\mathcal{H}_m(G) = [|\mathcal{N}_m(v)| \text{ for } v \in V]$$

[2]This follows because the stationary distribution of a Markov Chain on an undirected graph is proportional to the degree distribution.

We see that $\mathcal{H}_1$ is simply the degree distribution. The final distance metric can be summarized as:

$$d_m(G_1, G_2) = \frac{||\log(h_{\mathcal{H}_m(G_1)}) - \log(h_{\mathcal{H}_m(G_2)})||}{||\log(h_{\mathcal{H}_m(G_1)})|| + ||\log(h_{\mathcal{H}_m(G_2)})||},$$

where h_X is a vector with entries corresponding to the bin frequencies of X and $\log(Y)$ is simply the natural log of each element in the vector.

The right panel of Figure 1 shows the behavior of this metric for $m = 2$, using the same setup as with the Frobenius distance. We observe that we can add or remove up to 70% of the edges before the distance resembles that of a random graph. Furthermore we see that rewiring and neutral Bernoulli flipping preserves similarity extremely well. Other values for m were tested, but achieved unfavorable results.

4 Inference Procedure

As with most statistical inference procedures, our proposed procedure is based on the hypothesis that there exists a true underlying distribution from which the observed data is generated. In the case of platonic inference we call this the *platonic seed*. When examining real graphs, the platonic seed is unknown and the goal of the procedure is to approximate it from the *observed graphs*. When testing our procedures, however, we can access the seed directly to generate more observed graphs or even measure the error between fitted seed and the platonic seed. The proposed inference procedure is as follows:

1. Separate the set of observed graphs into a training and a test set. The training set will be used to infer the platonic seed; the test set will be used to evaluate the success.
2. Calculate an *inferred seed* from the training set of observed graphs, using a particular platonic model, as discussed in Section 5.
3. For both the training and test set, evaluate the error in the fit as follows:
   (a) Calculate the *observed graph distance distribution* of pairwise distances between all of the observed graphs, using a predefined distance metric.
   (b) Sample k graphs from the distribution defined by the inferred seed (these are called *fit-sampled graphs*).[3]
   (c) Calculate the *fitted-observed graph distance distribution* of pairwise distances between fitted and observed graphs, using the same predefined distance metric.
   (d) Calculate statistics comparing the observed distance distribution and the fitted-observed distance distribution. We used the Kolmogorov-Smirnov statistic (KS-statistic) and the Z-score of the distributions having the same mean.

The training error is then reported as a function of the statistics found using the training set of observed graphs, and the test error is that function of the statistics found using the test set of observed graphs. This function can be used to combine statistics found using various distance metrics.

To evaluate our inference procedure, we used both the matched Frobenius norm metric and the 2-hop unmatched metric, as well as one minus the KS statistic ("1-KS") statistic and the Z-score. To combine information from the two metrics we report the error of the inference procedure to be the maximum of the statistic found on either metrics (i.e., the reported Z-score is max of the Z-scores over the two metrics, and the reported 1-KS statistic is likewise calculated)[4]. The Z-score represents how many standard deviations the mean of the fitted-observed distance distribution is from the mean of the observed distance distribution (negative implies the mean of the fitted-observed distribution is lower than that of the observed distance distribution). Separately, the 1-KS statistic is the confidence level with which we can reject the hypothesis that the distributions are the same. For example, a 1-KS statistic of 0.90 implies that with 90% confidence we can reject that the distributions are the same. We conclude that the inference procedure is successful if

[3] Larger k will provide more data for step (d); we have defaulted to set k equal to the number of training observed graphs.
[4] Reporting the average could also be a reasonable error function.

there is a low 1-KS statistic (e.g., < 0.70). Even if this is not the case, if the Z-score is very negative, this means that the inferred seed generates graphs that are in some sense "close" to the observed graphs, with high probability. This latter test is not ideal because simply choosing one of the observed graphs *as* the platonic seed will result in a low Z-score, but nonetheless provides helpful information.

5 Platonic Models

5.1 Bernoulli MLE

As a first model, we hypothesize that the platonic seed is a complete, undirected graph on nodes V, with edge weights between zero and one, denoted p_{uv} for all (u, v), $u \neq v \in V$. These weights represent the strength of the relationship between the relevant nodes. Observed graphs are drawn from the platonic graph with edges appearing as a result of an independent Bernoulli trial for every node-node pair – the edge weights of the platonic graph are exactly the parameters of the edges' Bernoulli distributions.

Given k observed graphs, denote each graph $G_1 = (V, E_1)$, $G_2 = (V, E_2)$, $\ldots$, $G_k = (V, E_k)$. Consider a fixed node-node pair, (u, v), and define

$$(u, v)_i = \begin{cases} 1 & \text{if } (u, v) \in E_i \\ 0 & \text{if } (u, v) \notin E_i \end{cases}$$

Then $\{(u, v)_i\}_{i \in \mathbb{Z}_k}$ are k independently and identically distributed (iid) samples from the Bernoulli process $\phi(p_{uv})$. Under this model, the maximum likelihood estimator (MLE) for p_{uv} is

$$p_{uv} = \frac{1}{k} \sum_{i=1}^{k} (u, v)_i,$$

or simply the average occurrence of the edge. Therefore, under the Bernoulli model, the MLE of the platonic seed is easily identified as the average of observed graphs.

5.2 Low-rank Estimation

Bernoulli estimation tries to fit n^2 parameters from kn^2 observed potential edges. In general, however, because graphs are sparse, fewer than n^2 edges are observed in k graphs, so Bernoulli tries to estimate more parameters than data points. To prevent against overfitting, we can require the inferred seed to be of a specified low rank [1] or regularize the nuclear norm of the solution [8]. Here we focus on restricting the inferred seed to be low-rank. That is, let P be the probability matrix found by the Bernoulli MLE. We can write its singular value decomposition as $P = U \Sigma U^T$ (note P is symmetric). Let U_r and Σ_r be the truncated versions of these matrices corresponding to the first r singular vectors and values, respectively. Then by the Eckart-Young theorem, we know that $\tilde{P} = U_r \Sigma_r U_r^T$ minimizes the Frobenius norm of the difference between any rank r matrix and P. From this, our inferred low-rank seed is $L = U_r \Sigma_r^{1/2}$, so $\tilde{P} = LL^T$.

Note, we examined the performance of low-rank seeds that were calculated as the average of the low-rank factors of each observed graph (rather than the low-rank factor of the Bernoulli graph), and it had nearly identical results to the proposed method. We exclude it in future analyses because it is computationally much more burdensome. Additionally, we considered further convex optimization of the low-rank factors like Alternating Least Squares[4] to more strongly emphasise observed edges over non-edges, but ultimately concluded that this might lead to overfitting.

5.3 Kronecker Graph Fitting

Stochastic Kronecker Graphs are generated from a probabilistic seed matrix Θ. Performing κ Kronecker products of this seed matrix induces an adjacency matrix of probabilities, with dimension $2^\kappa \times 2^\kappa$. Each entry corresponds to an edge probability p_{uv}. Kronecker Graph Fitting attempts to find the most likely seed Θ which could have generated a given graph G. One method to infer such a seed is the KronFit[6] algorithm, which takes advantage of the Metropolis Hastings algorithm. Given that we observe k graphs, we

use KronFit to generate k seeds, $\Theta_1 \cdots \Theta_k$, and then view each of these seeds as an independent sample of a platonic Θ. The MLE of Θ is simply the average $\hat{\Theta} = \frac{1}{k}\sum_{i=1}^{k}\Theta_k$.

Kronecker Graphs only depend on four parameters, which make them robust to overfitting. In addition they have many similar properties to real world graphs, making them a good candidate for a platonic seed. One thing to note, however, is that Kronecker models do not assume a known node ordering, so matched-node metrics may not be useful when comparing graphs generated from a common Kronecker seed.

6 Evaluation of Procedure and Model

Before running the full inference procedure that compares the distribution of the observed distances to the fitted-observed distances, we wanted to see how well the fitting procedures reconstructed the platonic seed itself. In Matlab we randomly generated graphs from known Bernoulli and low-rank platonic seeds, ran both types of fits, and evaluated how far the inferred seed was from the platonic seed. As expected, the Bernoulli model overfit to the observed graphs and inferred seed was not very similar to the platonic seed. Luckily, regardless of the rank of the platonic seed (1, 10, or full rank), the low-rank model did reconstruct an accurate inferred seed. In fact, for the size of graphs in question, the rank-1 model performed the best. Because of the low number of observed graphs, sparsity of the graphs, and large number of parameters being fit, it should not be surprising that a highly regularized solution performed best.

To evaluate the inference procedure we constructed Kronecker, Bernoulli, and low-rank (rank-1 and rank-10) platonic seeds and ran the procedure with each type of fitting model. For computational efficiency we used 10 observed graphs for training and testing, with 128 nodes and average degree of 3. We used both Frobenius norm matched distance and 2-hop unmatched distance as the metrics, and evaluated the procedures by looking at the max of the 1-KS statistics and Z-scores across metrics. By examining the max across the metrics, if the model fails on either the matched or unmatched basis, the procedure will fail. Table 1 shows a summary of the data.

Table 1: Maximum 1-KS and Z statistics among both matched and unmatched similarity metrics.

Fit Type	**Training (1-KS, Z)**				**Testing (1-KS, Z)**			
	Graph Type				**Graph Type**			
	Kronecker	Bernoulli	LowRank10	LowRank1	Kronecker	Bernoulli	LowRank10	LowRank1
KronFit	(1.00, 1.71)	(1.00, 4.87)	(1.00, 2.16)	(1.00, 3.04)	(1.00, 1.87)	(1.00, 4.22)	(1.00, 2.57)	(1.00, 3.12)
Bernoulli	(1.00, 1.51)	(1.00, 4.20)	(1.00, 2.03)	(1.00, 2.24)	(1.00, 1.62)	(1.00, 4.24)	(1.00, 2.46)	(1.00, 1.73)
LowRank10	(1.00, 2.78)	(0.46, 0.12)	(0.38, 0.08)	(0.78, -0.12)	(1.00, 2.85)	(0.35, 0.04)	(0.67, 0.08)	(0.30, 0.02)
LowRank1	(1.00, 1.38)	(0.63, 0.11)	(0.95, 0.22)	(0.74, -0.12)	(1.00, 1.70)	(0.99, 0.37)	(0.57, 0.02)	(0.38, 0.13)

The Kronecker model produced matched fits with negative Z-scores ($\sim -2\sigma$) in training and testing when the platonic seed was Kronecker. This indicates that the graphs sampled from the inferred seed are closer to the observed graphs than they are to each other. In principle this sounds good, but it means that the fitted seed does not have enough variance to replicate the platonic seed, and we did not find the platonic seed. Using the unmatched metric, the Kronecker model produced high Z-scores when the true platonic seed was Kronecker, indicating that the fitted seed did not generate graphs with comparable 2-hop degree distributions. Collectively, this implies that the initiator matrix found from Kronecker fitting was not representative of the platonic initiator, even when the true platonic seed was Kronecker. Additionally, when the platonic seed was not Kronecker, the Kronecker fit model could not replicate the observed graphs.

The Bernoulli model generates a sparse inferred seed because there are few observed graphs and each has many fewer edges than nodes. Because the univariate Bernoulli distribution has zero variance when ϕ equals zero, the sampled graphs from the Bernoulli inferred seed are all very similar; in fact, they can only contain edges found in the observed graphs. The averaging effect of the Bernoulli model finds a seed in the "middle" of the observed graphs, but does not have enough variance to replicate the platonic seed. Knowing this, it is not surprising that the training Z-scores using the matched metric and Bernoulli fitting were very negative (-4σ to -9σ). The test Z-scores were in the range -1σ to -2σ. However, using the unmatched metric, the Z-scores of the Bernoulli fit were very positive, hence the combined Z-scores are all greater than 1. Thus, the Bernoulli fit model could not replicate the platonic seed.

Table 2: Graph statistics for different CrunchBase networks. Note that N/A values are used where the result is not applicable (for example, many nodes share the same PageRank by virtue of the edge generation criterion).

Graph Type	Clust. Coef.	Avg. Diameter	Avg. Degree	Top Page Rank
Acquisition	0.000	0.393	0.276	Oracle Google Ebay
Common Category	0.956	0.988	71.055	N/A N/A N/A
Common Cofounder	0.000	0.048	0.029	N/A N/A N/A
Common Investor	0.170	3.263	6.773	Twitter Dropbox Path
Common Tag	0.025	4.986	0.258	Gendai Games Celframe Socialcam
Competitor	0.106	7.042	1.171	Google IBM Microsoft

The low-rank model produced low magnitude Z-scores for training and test sets, using both matched and unmatched metrics. Additionally, the 1-KS statistics were quite small for some of the models. The combined test 1-KS statistics for the 10-rank fit, seen in the right panel of Figure 1, are generally less than 0.70, and in two cases less than 0.40. This suggests that the low-rank model can produce graphs that appear to be from the same distribution as the observed graphs if the platonic seed is not Kronecker.

7 Data Collection

With an accurate procedure to generate platonic graphs from sets of observed graphs, we elected to infer and analyze the platonic graph for the most popular companies on CrunchBase, a large tech-company database. To do this, we created numerous observed graphs of CrunchBase companies, using various criteria to decide if an edge should be added between two companies. For example, in one observed graph, two companies are connected if they share a common investor, where in another, they are instead connected if they are listed as competitors of each other. Thus, the observed graphs each capture part of the underlying platonic corporate network.

We extracted the JSON data of over 186,000 companies listed in CrunchBase, but quickly realized that most companies had little data, making for very sparse graphs with many disconnected nodes. As companies with larger JSON files should have more interesting data, we used the file size of the JSON to determine which companies to include in our analysis. We sampled various sets of companies using different file size thresholds, and in the interest of space and time complexity, we elected to use the top 1,024 largest companies on CrunchBase for our analysis. (1,024 was the number chosen due to the Kronecker procedure favoring graphs with 2^n nodes for some n).

Ultimately, we generated six observed CrunchBase graphs, feeding these into our previously mentioned inference procedures to generate a platonic CrunchBase graph. The statistics for these six graphs are listed in Table 2. In addition to the statistics, we calculated the distributions for each of clustering, diameter, and degree. Figures 2 and 3 depict plots for degree and diameter distributions.

8 Inference on CrunchBase Data

To evaluate whether a platonic seed could be inferred from the CrunchBase graphs, we applied the inference procedures and computed the Z- and 1-KS-statistic, as outlined in Section 6. The results are shown in Table 3. We see that the general trend for the fitting procedure was to perform better when evaluated

Figure 2: Degree distribution of Competitor Graph.

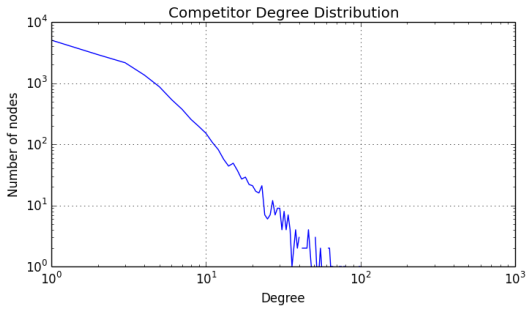
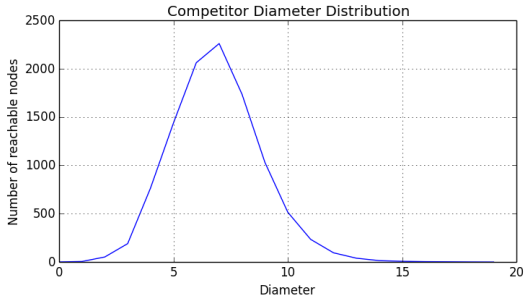


Figure 3: Approximate diameter distribution of Competitor Graph (sampling 20 nodes).



with the unmatched metric, suggesting that the inference procedures were better able to capture the 2-hop neighborhood size than to re-create the actual edges in the graphs. Focusing on unmatched metrics, KronFit appears to fit the CrunchBase graphs most accurately, due to both the low Z- and 1-KS-statistic. This indicates that the distributions of observed unmatched distances were reasonably similar in mean and shape to the fitted-observed distances. KronFit does not preserve node ordering, so it makes little sense to evaluate this fitting procedure using a matched metric and accordingly it is difficult to conclude to what extent KronFit succeeded. The low-rank model provided reasonable error using the unmatched metric, but could not reproduce the matched metric distribution.

Because we do not have an independent test set, we cannot with confidence conclude that any of the fitting procedures were able to accurately fit the CrunchBase graphs. We recommend that observed graphs be split into training and test sets; however, in this case, six graphs were not enough to meaningfully fit *and* test the fit, so we only use the graphs as training data.

Table 3: CrunchBase 1-KS and Z statistics for both matched and unmatched similarity metrics.

Fit Type	**CrunchBase (1-KS, Z)**	
	Graph Type	
	Matched Metric	Unmatched Metric
KronFit	(0.95, 0.41)	(0.26, 0.04)
Bernoulli	(0.97, -0.59)	(0.91, 0.17)
LowRank10	(1.00, -0.63)	(0.69, 0.01)
LowRank1	(1.00, -0.53)	(0.51, 0.80)

9 Difficulties and Future Work

As seen in the data, the CrunchBase platonic graph inference did not go as well as hoped. In the matched nodes case, every one of the fitting techniques performed poorly, in that the fit-sampled graphs were not likely from the same distribution as the observed graphs and the means were also fairly disparate. In the unmatched case, the results were slightly more promising, but still appear lackluster. While this is unfortunate, we understand a few shortcomings of our procedures that may have caused this to occur.

First, we neglected to account for variation in the number of edges in the observed graphs. For our random graph generation, each was created with approximately the same number of edges, but the CrunchBase data does not follow this rule. For example, the graph that represents the common cofounder relationship between companies contains just 15 edges, as it is fairly unlikely that a person would create and leave multiple successful companies, but the common category graph has over 36,000, due to the limited number of categories a company can have. While edges are scaled down when computing averages over multiple graphs such that one graph doesn't dominate over the rest, further testing is needed to find models that can accommodate this variation.

Second, the CrunchBase data was fairly limited in its scope and its usability. The CrunchBase JSON files provided only some information for which we could construct graphs that actually represented something

meaningful. For example, we could link companies together if they appeared in the same locale, but this would have resulted in far too many edges that only demonstrated a very weak, almost incidental, connection between two companies. Other options that we explored, such as tf-idf comparisons of descriptions, proved to be too computationally expensive and too easily gamed to be helpful. In the end, we only used six CrunchBase graphs, but we would have liked more. In addition, because CrunchBase can be edited by anyone, some of the data was skewed in ways we didn't anticipate. Some companies had tags that were only tangentially related to their business (such as the tag "usa" simply because they operated in the US), which created edges that held the same weight as one between two companies which shared a specific tag like "ipad-game". Further, missing data from CrunchBase does not preclude the possibility of there being a real-world connection between two companies. As an example, one company that acquired another recently may not have that change reflected in CrunchBase, and thus our graphs were not necessarily indicative of the full state of the corporate network.

Third, we were limited to working with CrunchBase graphs of size 1024 nodes due to the time complexity of our various metrics on graphs with many nodes. KronFit in particular takes several minutes for graphs of size 1024, which implies long computation time when many graphs must be fit. Graphs of larger size did not converge in time for this report.

Fourth, none of our platonic models include covariance structure of the edges despite that we expect triangles to occur with higher probability, as suggested by [5]. This could be done by directly calculating the covariance of the edges of the observed graphs and regularizing toward the identity matrix, as suggested by [3]. In the regularization limit, this will result in all of the edges being independent. Adding covariance may make the fit-sampled graphs more realistic, which may reduce their distance from the observed graphs.

Lastly, other distance metrics may have fared better than Frobenius norm and 2-hop distance. If we had more time, perhaps using correlation distance would have proved to work better in the matched case, but its time complexity made it unattractive. Graph similarity in general is a complicated problem with limited literature, thus complicating our results.

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] Stefano Battiston and Michele Catanzaro. Statistical properties of corporate board and director networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):345–352, 2004.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[4] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[5] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 462–470. ACM, 2008.

[6] Jure Leskovec and Christos Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*, pages 497–504. ACM, 2007.

[7] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[8] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *J. Mach. Learn. Res.*, 11:2287–2322, August 2010.

[9] Bernadette CM van Wijk, Cornelis J Stam, and Andreas Daffertshofer. Comparing brain networks of different size and connectivity density using graph theory. *PLoS One*, 5(10):e13701, 2010.