**Graphs and Networks** Lecture 23

# Link Prediction and Anomoly Detection

*Daniel A. Spielman* November 19, 2013

## 23.1 Disclaimer

These notes are not necessarily an accurate representation of what happened in class. They are a combination of what I intended to say with what I think I said. They have not been carefully edited.

## 23.2 Overview

This lecture is about the problem of predicting the formation of links in networks and evaluating the reliability of existing links. It is based upon the paper [LNK07], but will also touch upon the work reported in [SEH$^+$06], [LLC10], [RJ05] and [GR03].

I consider link prediction to be the problem of determining which links are most likely to be formed in an evoloving graph. We would also like to consider the problem of detecting "noise" edges from graphs: those that are in a graph, but which don't belong. A dual problem to this is that of detecting missing edges. Some people like to talk about the problem of de-noising a graph. I do not like this problem because the edges that are most surprising will probably look like noise. So, I prefer to think about detecting anomolous edges.

## 23.3 Types of problems

The link-prediction problem shows up in many different forms. Different forms will probably require different solutions. The forms we will consider include:

1. Predicting future links. This is the problem considered in [LNK07] and [LLC10]. This problem arises when one has a network that either changes over time or in which each edge has a time associated with it. For example, one could consider a social network, the web, a collaboration network (where times are dates when papers were written), or the phone call graph. At any momment in time, a social network or the web is a graph. To turn a collaboration or call graph into a network, one can take all the edges that appear over a certain time span.

   The problem is then to predict which edges will appear in the future time spans. The advantage of this problem is that one can check which edges actually appear later.

2. Evaluating the reliability of links. This is the problem considered in [GR03]. This paper considers protein-protein interaction networks. These are supposed to include a link between two proteins that interact. However, there are many ways of measuring which proteins interact, and some are more reliable than others. As is typical, the faster tests are less reliable. So, these tests will give both false positives and false negatives.

   The goal in this situation is probably to determine the reliability of a link by using the structure of the other links.

   The authors of this paper are somewhat confused about some basic points. The algorithms they test are solely based on measuring the overlap in neighborhoods between the endpoints of an edge. So, their algorithms can work for graphs with many triangles (high clustering coefficient). However, they motivate their algorithms by talking about "small world graphs" and low diameter. Of course, low diameter has nothing to do with clustering coefficients.

3. Identifying missing links. This is essentially the problem considered in [SEH$^+$06]. These authors consider a biological data in which they know (or have evidence that) some things are related to others. For example, they know that some genes are related to certain conditions (phenotypes). These links are typically inferred from research papers.

   They would then like to determine if some gene is likely to be related to another condition. The key example in the paper is an attempt to determine which genes are related to Alzheimer's disease.

   In this problem, the existance of some links is known, and they want to determine the likelihood of certain other pairs of links. For example, they might want to consider evaluating which of some one hundred genes are related to Alzheimer's. The could run their test for each of the one hundred possible links, and then check out the most likele ones in the lab.

4. Guessing whether or not a link exists. This problem is not directly considered in any of the papers under consideration. To formulate it, we assume that partial information about the graph is known. For example, we might know that some set of edges $F$ exists in the graph, and that some set of potential edges $H$ do not exist in the graph. Then, for a potential edge $e$ that is not in $F$ or $H$, we would like to guess whether it is or is not in the graph.

   This is a variant of the collaborative filtering problem. I'll say a little about it in class.

## 23.4 Methodology of Liben-Nowell and Kleinberg

This paper deals with a collaboration network, linking scientists who have written papers together. This paper performs an empirical evaluation of many reasonable approaches to predicting links. It establishes a benchmark for future papers.

They divide time into two intervals, a training interval and a test interval. They use the data from the training interval to predict the existence of links in the test interval. The first problem they must deal with in this paper is that the nodes in the test graph can be different from the nodes in the training graph. A natural solution to this would be to just consider the set of nodes in both graphs. But, they go further. They drop all nodes that have degree less than 3. The then only study the nodes that have degree at least 3 in both graphs, which they call the Core.

The next question they address is how they should evaluate the quality of a link-prediction algorithm. They propose a natural approach. They let $n$ be the number of edges between vertices in the Core of the test graph. They then take the $n$ edges that recieve the highest score from a link-prediction algorithm, and see how many of them appear in the Core. Rather than just reporting this percentage, they compare it to the percentage obtained by the most naive algorithm–just picking $n$ random pairs of vertices in the Core. They then report the multiplicative improvement of an algorithm over the random algorithm.

They exclude from consideration all edges that are in the training graph.

One can debate the merits of this approach, and others are suggested in other papers. But, I don't think it is worthwhile to spend too much time debating how to measure these, as the discussion all hot air unless one has an application. Then, the application will determine the merit of an algorithm.

## 23.5 Results of Liben-Nowell and Kleinberg

Liben-Nowell and Kleinberg explore many naive algorithms for estimating the likelihood of links. For each pair of nodes $x$ and $y$, they produce a measure of the likelihood that an edge appears between them.

The following is a rough list. I will say more about them in class. In the following, I use $N(x)$ to denote the neighbors of a node $x$.

1. Length of shortest path between $x$ and $y$. We view the short paths as most likely. In their experiments, the number of pairs $x$ and $y$ at distance 2 exceeded $n$. So, they evaluated what happens if one breaks ties at random.

2. Common neighbors: $|N(x) \cap N(y)|$.

3. Jaccard's coefficient : $|N(x) \cap N(y)| \,/\, |N(x) \cup N(y)|$.

4. Adamic/Adar: $\sum_{z \in N(x) \cap N(y)} 1/\log d(z)$. This weights each common neighbor by the log of its frequency.

5. Preferential attachment: $|N(x)|\,|N(y)|$.

6. Katz$_\beta$: $\sum_{t=1}^{\infty} \beta^t \left|\text{paths}_{x,y}^{t}\right|$. This is the sum over $t$ of the number of paths between $x$ and $y$ of length $t$, multiplied by $\beta^t$. It was proposed by Katz in [**?**]. The parameter $\beta$ must be set. There are two variations of this, one in which we just use the adjacency matrix, and one in which we use a weighted adjacency matrix. In the weighted case, we count the number of collaborations between a pair of authors.

   Note how similar this is to Personal PageRank.

   The idea behind this measure is that long paths should be discounted more. However, this measure only really makes sense if the long paths do not dominate the computation. But, they can if $\beta$ is too big. Let $A$ be the adjacency matrix of the graph, and let $e_x$ be the elementary unit vector in direction $x$. Then, the number of paths of length $t$ between $x$ and $y$ is

$$A^t(x,y) = e_x^T A^t e_y.$$

The terms $A^t$ will tend to grow unless $\beta$ is smaller than the largest eigenvalue of $A$. So, we should choose $\beta < 1/\lambda_{max}(A)$. A choice like $\beta = 1/2\lambda_{max}(A)$ would be reasonable. Unfortunately, people often forget this. They don't notice that their sum is going to infinity because they just truncate after a couple of terms. We, of course, will also just truncate after a couple of terms.

7. Personal Pagerank, called rooted PageRank in this paper. Again, we must set the parameter $\alpha$.

8. Hitting Time. The hitting time $H_{x,y}$ is the expected time it will take for a random walk that starts at $x$ to hit $y$. As nodes that have high probability under the stationary distribution are more likely to be hit, they also consider normalizing this by multiplying by this probability: $H_{x,y}\pi(y)$.

9. Commute Time. Recall that the commute time is $H_{x,y} + H_{y,x}$, and that it is proportional to the effective resistance between $x$ and $y$. They also consider summing the normalized terms.

10. SimRank. This was defined by Jeh and Widom. It can be thought of as a symmetrized version of personal PageRank. In terms of random walks, one starts a random walk at both $x$ and $y$, and then considers the expectation of $\gamma$ raised to the power of the time these walks first intersect. The parameter $\gamma$ must be set.

11. Low-rank approximations. This is a technique that has proved very useful in collaborative filtering, at least when one regularizes to prevent over-fitting. They consider many reasonable things to do with the matrix obtained, including using the entries of the matrix (the most obvious), using the inner products of rows, and running Katz's algorithm on it.

In class, we will stare at the results a little.

Let me point out one big problem with using algorithms that have parameters. It is not clear how to set them. You could make some reasonable guesses at reasonable values. Or, you could report the results obtained for the best setting of the parameters. But, this favors algorithms with more parameters.

## 23.6 Supervised Learning

The algorithms I just mentioned fall into the class of unsupervised learning algorithms. Lichtenwalter *et. al.* suggest treating this as a supervised learning problem. In this case, one can use some of the training data to train any parameters. One can also use the training data to find the best way to combine different predictors.

## 23.7 A methodology for anomoly detection

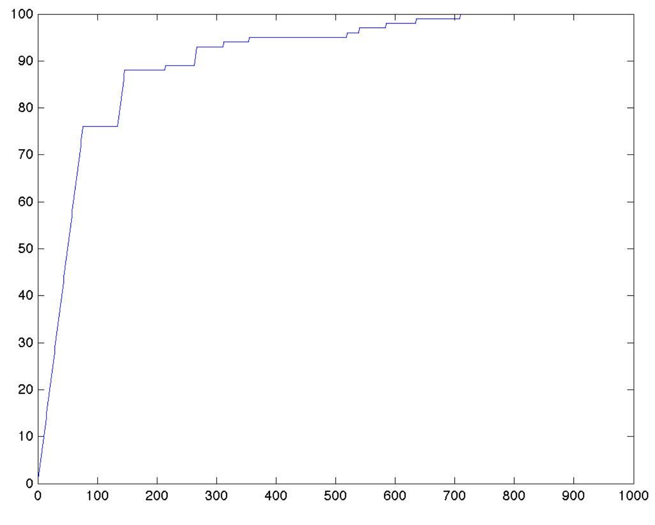
The problem with detecting anomolous edges is that we don't have a good way to decide if our graph has any. I will get around this by adding random edges to a graph, as these should certainly be anomolous. I will chose random edges by selecting their endpoints at random. But, one could also consider making one endpoint the destination of a short random walk from another.

To test a measure of anomolousness (better word?), I will compute my measure for each of the added edges along with a large number of the original edges. We will then check how many of the highest ranked edges were actually noise. For example, I will now take a graph, add 100 noise edges, and select 900 additional original edges. I will then compute the overlap in the neighborhoods between the endpoints of each edge.

```
load snap_wikivote
[amod,noise,real] = ps6gen(a);
edges = [noise;real];
for i = 1:1000, e = edges(i,:);
  sc(i) = amod(:,e(1))'*amod(:,e(2));
end
```

I will then sort the edges by score, and plot for every $x$ the number of noise edges that are ranked in the top $x$. Here, a lower score results in a higher rank.

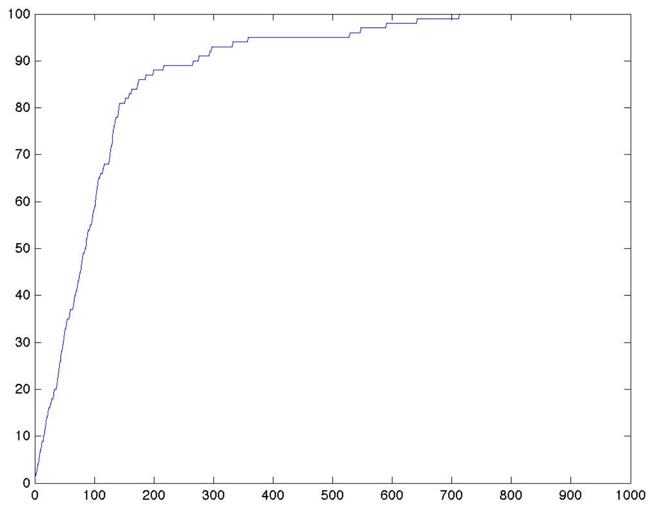
```
x = [ones(100,1); zeros(900,1)];
[val,ord] = sort(sc);
plot(cumsum(x(ord)));
```



This plot is amazing! The first turn in the plot is at $(76, 76)$, which means that the 76 edges of lower rank were our noise edges. But, this result is too good. The reason it came out this way is that there are 134 edges of score 0, and these include of our random edges. The reason they appeared first is because the way that we broke ties between edges of equal score put them first. It is more reasonable to break ties at random. The following code does this by permuting the edges to begin.

```
pe = randperm(1000);
edges = edges(pe,:);
for i = 1:1000, e = edges(i,:);
sc(i) = amod(:,e(1))'*amod(:,e(2));
end
x = [ones(100,1); zeros(900,1)];
x = x(pe);
```

```
[val,ord] = sort(sc);
plot(cumsum(x(ord)));
```



We still did pretty well. The point at $(142, 81)$ tells us that of the first 142 edgses, 81 were noise, and the point at $(358, 95)$ tells us that 95 of the noise edges were in the first 358.

People spend a lot of time deciding how to measure which algorithm is best. For example, some people choose an arbitrary point to measure, like "how many of the highest ranked 100 (or 500 or 1000) edges are noise edges"? Some think that picking an arbitrary point is just too arbitrary. So, they look at the integral under the curve and call it either AUC (Area Under Curve) or ROC (Receiver Operating Characteristic). These look fancier, but are equally arbitrary.

## References

[GR03] DS Goldberg and FP Roth. Assessing experimentally derived interactions in a small world. *Proceedings of the National Academy of Sciences of the United States of America*, 100(8):4372, 2003.

[LLC10] Ryan Lichtenwalter, Jake Lussier, and Nitesh Chawla. New perspectives and methods in link prediction. *KDD '10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, Jul 2010.

[LNK07] D Liben-Nowell and J Kleinberg. The linkprediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.

[RJ05] Matthew J Rattigan and David Jensen. The case for anomalous link discovery. *ACM SIGKDD Explorations Newsletter*, 7(2):41–47, 2005.

[SEH$^+$06] P Sevon, L Eronen, P Hintsanen, K Kulovesi, and H Toivonen. Link discovery in graphs derived from biological databases. *Data Integration in the Life Sciences*, pages 35–49, 2006.