Inherent Limits on Topology-Based Link Prediction

Anonymous authors
Paper under double-blind review

Abstract

Link prediction systems (*e.g.* recommender systems) typically use graph topology as one of their main sources of information. However, automorphisms and related properties of graphs beget inherent limits in predictability. We calculate hard *upper* bounds on how well graph topology alone enables link prediction for a wide variety of real-world graphs. We find that in the sparsest of these graphs the upper bounds are surprisingly low, thereby demonstrating that prediction systems on sparse graph data are inherently limited and require information in addition to the graph topology.

1 Introduction

Graph-based link prediction systems are widely used to recommend a wide variety of products and services. These modern-day systems are even used to find friends and surface possible romantic partners. Usually, these recommendations are based on a combination of node features and the topology (*i.e.* the link-structure) of the graph-data. However, most graphs are known to possess symmetries (*i.e.* automorphisms) in their topology. These automorphisms reduce the prediction task to guesswork and therefore place inherent limits on the predictability of many tasks and datasets. This raises two foundational questions in machine learning on graphs: What are the inherent limits on a graph structure's predictability, and do contemporary systems approach these performance limits?

The goal of the present work is to answer these questions. To do so we investigate how much information a graph's topology alone can provide to a link prediction algorithm. For example, consider the following scenario illustrated in Fig. 1: Imagine you are shown both a link prediction task and the answer to the same task (*i.e.* the links you must predict); imagine further that you are asked to perform link prediction on an anonymized version of the same problem. We call this the *maximally informed link prediction task*. We then ask: what is the best that an algorithm could do at this maximally informed link prediction task? Answering this question gives a hard upper bound on the performance that *any* algorithm could achieve when presented with the standard prediction task.

At first, it may seem that the maximally informed link prediction task is trivially easy. However, as Figure 1 shows, *symmetries* in the graph can render several possible edge-predictions structurally identical and therefore equally valid.

Of course, in practice, most systems will perform much worse at the standard link prediction task than an ideal algorithm could perform at the maximally informed task. Any link prediction algorithm will have some inherent (often implicit) modeling assumptions about the graph. For example, a simple model like triadic closure assumes that the likelihood of an edge is proportionate to the number of triangles the edge would be involved in Bianconi et al. (2014); Klimek & Thurner (2013); Jin et al. (2001); Davidsen et al. (2002). Expressed in a Bayesian fashion, we can think of a link prediction algorithm as conditionalizing on evidence, where the data the algorithm sees is its evidence and the algorithm's inherent assumptions form its prior. One could think about an algorithm performing the maximally informed link prediction task as an algorithm doing standard link prediction with a perfect prior (*i.e.* 100% confidence on the correct graph). We focus on the maximially informed link prediction task because it enables us to establish limits that exists in the data itself, without making *any* assumptions about the link prediction algorithm.

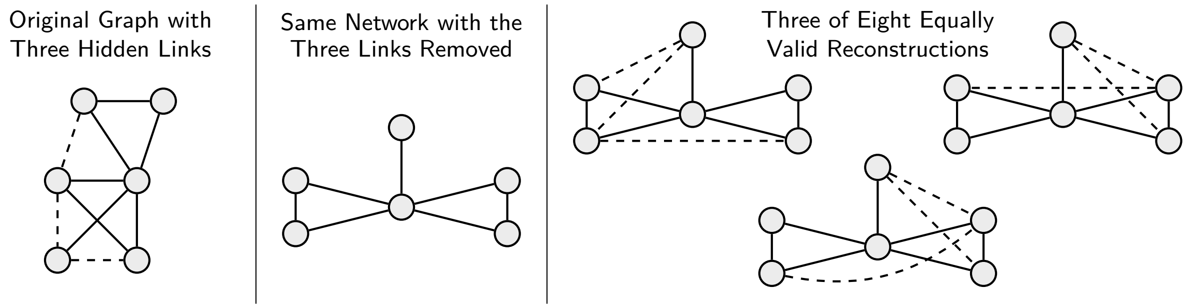
Original Graph with Three Hidden Links
Same Network with the Three Links Removed
Three of Eight Equally Valid Reconstructions

Figure 1: Toy example graph with three held-out edges for link prediction testing. In this case, there exist eight equally plausible edge sets that form a graph isomorphic to the original. However, only one of these eight is “correct” from the perspective of standard link prediction evaluation.

To our knowledge, this is the first work to quantify a maximal performance score that link predictors can obtain on a given task. Other work in predictibility has focused on measures of predictibility distinct from evaluation scores. For instance, Abeliuk et. al. study how the predictability of time-series data degrades as the amount of data available decreases; they quantify predictability in terms of permutation entropy and signal self-correlation, as well as actual prediction performance of specific models Abeliuk et al. (2020). Permutation entropy has also been found to be useful to measure predictability in ecology and physics, and self(auto)-correlation in finance Abeliuk et al. (2020); Bandt & Pompe (2002); Garland et al. (2014); Lim et al. (2013).

Scholars have also analyzed predictability limits in other domains. For instance, some have used notions of entropy to measure predictability limits on human travel Lu et al. (2013); Song et al. (2010) and disease outbreaks Scarpino & Petri (2019). Predictability is related to system complexity and chaos Boffetta et al. (2002). For instance, minute uncertainties on initial conditions can greatly limit one’s ability to make accurate weather forecasts Zhang et al. (2019).

Others have done excellent work on the related but distinct case that the ground truth (*i.e.* correct output) itself is uncertain or inherently fuzzy. For instance, in these sorts of settings one might need an alternate way of scoring a classifier, such as Survey Equivalence Resnick et al. (2021). Rather than fuzzy ground-truth, the present work focuses on cases where the correct output is clearly known during evaluation, but where limits in predictibility come from symmetries within the input data.

Ultimately, we find that commonly used graph datasets have surprisingly low predictability limits and that some GNNs appear to have exceeded these upper bounds in their reported results. We offer some possible explanations for these findings in our discussion.

2 Formalisms

2.1 Graphs

We represent a graph G as $G = (V, E)$ where V is the set of vertices (*i.e.* , nodes) and E is the set of edges. The edges are pairs of vertices. If the graph’s connections are considered to have a direction, we say that $E \subseteq V \times V$ and that the graph’s *non-edges* are $(V \times V) \setminus E$. If the connections do not have a direction, then the edges are unordered pairs: $E \subseteq \{\{a, b\} \mid (a, b) \in V \times V\}$. However, for simplicity, it is standard to always write (a, b) rather than $\{a, b\}$ even when talking about undirected graphs. An edge of the form (a, a) is called a *self-loop*.

2.2 Isomorphisms

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we say that they are *isomorphic* if there exists a way to align the two graphs' vertices so that the structures overlap perfectly. Formally, G_1 and G_2 are isomorphic (expressed as $G_1 \cong G_2$) if there exists a bijection between the vertices $f : V_1 \rightarrow V_2$ such that $(a, b) \in E_1 \leftrightarrow (f(a), f(b)) \in E_2$. In this case the function f is called an *isomorphism.* In this paper, whenever we refer to two graphs as being *equivalent* or *identical* we mean that they are isomorphic.

If f is an isomorphism between two graphs G_1 and G_2 we will sometimes denote this as $G_1 \cong_f G_2$.

2.3 Automorphism Orbits

Within the context of a single graph, the *automorphism orbit* of an object (*i.e.* a vertex or an edge) captures its equivalence with other objects in the graph. Two objects are in the same orbit if and only if the data *in no way* distinguishes between the two objects.

An *automorphism* of a graph is an isomorphism of the graph with itself. That is, an automorphism of a graph $G = (V, E)$ is a bijective function $f : V \rightarrow V$ such that:

$$(a, b) \in E \leftrightarrow (f(a), f(b)) \in E$$

The set of all automorphisms of a graph G form the *automorphism group* of the graph and is denoted $\text{Aut}(G)$.

The *automorphism orbits* of a graph typically refer to collections of equivalent vertices; however, they can also refer to collections of equivalent edges. The orbit of a vertex a in graph G is the set $\text{AO}_G(a) = \{f(a) \mid f \in \text{Aut}(G)\}$. Similarly, the orbit of an edge $e = (a, b)$ in graph G is the set $\text{AO}_G(e) = \{(f(a), f(b)) \mid f \in \text{Aut}(G)\}$. Note that $a \in \text{AO}_G(a)$ and $e \in \text{AO}_G(e)$ due to the trivial automorphism $f(x) = x$.

We can even consider the orbits of *non-existent* edges (*i.e.* non-edges). Let $(a, b) \notin E$ be an edge which is not in G. We can still define the orbit of (a, b) to be $\{(f(a), f(b)) \mid f \in \text{Aut}(G)\}$. These orbits are collections of edges not in G which are equivalent given G.

In the context of this paper, two objects (vertices or edges) are considered equivalent if they are in each other's orbits. We will denote the set of all automorphism orbits for the vertices of a graph G as $\mathcal{AO}_{V,G}$ – likewise with the edges and non-edges: $\mathcal{AO}_{E,G}$ and $\mathcal{AO}_{\bar{E},G}$ respectively. The set of all automorphism orbits forms a partition over the entities.

2.4 K-hop Neighborhoods

In practice, most link prediction algorithms do not use the entire graph when predicting the probability of edge membership. Rather, they tend to use local context. We formalize one intuitive notion of local context here that will be used throughout the paper.

Given a node or an edge, we can consider the nodes surrounding the entity to be the collection of nodes you could reach by beginning at the node (or the edge's endpoints), and taking up to k steps across edges for some value k. We can express this formally as follows:

Given a vertex $x \in V$, let $N(x)$ be x's neighbors – that is, $N(x) = \{y \mid (y, x) \in E \vee (x, y) \in E\}$. Now, given a set of vertices $S \subseteq V$, we can define that set's neighbors to be $N(S) = \bigcup_{x \in S} N(x)$. For any natural number k, we define the k-hop neighbors of S to be $N_k(S) = S$ when $k = 0$ and $N_k(S) = N_{k-1}(S) \cup N(N_{k-1}(S))$ when $k > 0$.

We can now define the k-hop neighborhood of a set of vertices S. It is the induced subgraph on the k-hop neighbors. Formally, it is the graph

$$G_k(S) = (N_k(S), \{(a, b) \mid (a, b) \in E \wedge a, b \in N_k(S)\})$$

For an edge $e = (a, b)$ we define its k-hop neighborhood $G_k(e)$ to be the k-hop neighborhood of its endpoints: $G_k(\{a, b\})$.

3 Link Predictors and Their Evaluation

3.1 Link Predictors

A link predictor is essentially a binary classifier for non-edges. It produces a verdict indicating whether the (non-)edge is or should be a member of the graph or not.

Let $G = (V, E)$ be a graph and $\bar{E}$ be the set of non-edges in G; that is, $\bar{E} = \{(a, b) \mid a, b \in V \wedge (a, b) \notin E\}$. A hard link predictor (*i.e.* hard binary classifier) for G and $\bar{E}$ is a process/algorithm/function $\ell_G : \bar{E} \to \{\texttt{Positive}, \texttt{Negative}\}$ that gives a non-edge a label (Positive/Negative). A soft link predictor (*i.e.* , soft binary classifier) for G and $\bar{E}$ is a function $\ell_G : \bar{E} \to \mathbb{R}$ that gives a non-edge a score. The higher the score, the more likely the non-edge is considered to be one of the Positives; the lower the score, the more likely the non-edge is considered to be a Negative. The function may be the result of training a model on a collection of correct edges/non-edges via manual parameter tuning, statistical analysis, or any number of other methods.

In practice, soft classifier scores are often probabilities. Further, the scores are often turned into hard labels by picking a threshold value t and giving all entities with a score $\geq t$ the Positive label and all others the Negative label.

In the present work we require that a soft link predictor be deterministic when scoring edges and that the score that it gives for one non-edge does not depend on whether the classifier has already given a score to another non-edge. These requirements, while often desirable, might not hold true of some actual procedures/algorithms.

We again note that the present work focuses exclusively on the graph topology. Any notion of labels or ids or other information on the nodes and edges is therefore ignored for the purposes of the present work. This means that the representation or ordering of vertex labels does not change the algorithm's prediction. It makes no difference to the algorithm whether the vertices are labeled 1 through n, n through 1, or labeled with names like "Alice" and "Bob." Formally, we express these requirements with the following:

$$e_1 \in \mathrm{AO}_G(e_2) \to \ell_G(e_1) = \ell_G(e_2)$$

The point of this is to formalize the principle that: *When all information the predictor uses to classify two non-edges is structurally the same, then the output scores are the same.* When we consider link predictors that only use the k-hop neighborhood of an edge to score the edge, this principle becomes:

$$\big(\exists f.\ G_k(e_1) \cong_f G_k(e_2) \wedge f(e_1) \in \mathrm{AO}_{G_k(e_2)}(e_2)\big) \to \ell_G(e_1) = \ell_G(e_2)$$

This is to say that if edges e_1 and e_2 play an identical role in identical k-hop neighborhoods, then they are given the same score. Note that by definition $e_1 \in \mathrm{AO}_G(e_2)$ implies $\big(\exists f.\ G_k(e_1) \cong_f G_k(e_2) \wedge f(e_1) \in \mathrm{AO}_{G_k(e_2)}(e_2)\big)$ for any k.

3.2 Performance Scores for Link Predictors

In practice, almost all link prediction classifiers are soft classifiers. There are a number of nuances to how these scores are obtained that are worth highlighting here.

When evaluating soft classifiers, researchers tend to evaluate the predictors across different thresholds t. Each (soft predictor, threshold) pair represents a possible hard link predictor. Thus performance of a soft predictor can be considered to be the goodness of the collection of hard predictors it offers. This can be measured in terms of different criterion. One common criterion is the relationship between the predictor's True Positive Rate (TPR) and False Positive Rate (FPR), which generates the widely used ROC curve. Another common criterion is the relationship between the predictor's Precision and Recall, which leads to the Precision-Recall curve. For an in-depth analysis exploring the relationship between ROC curves and Precision-Recall curves (PR curves), we recommend the paper by Davis and Goadrich Davis & Goadrich (2006).

In the context of ROC and PR curves, an interpolation between points represents a way of combining the two hard classifiers (the two points) into a new hard classifier. This can be done by picking a value $\alpha \in [0, 1]$ and tossing an α-weighted coin every time an entity is scored to decide which of the two hard classifiers to use for the entity. This is implicitly how we as well as Davis and Goadrich perform interpolation. It turns out that the popular trapezoidal interpolation is incorrect for Precision-Recall space, because hard classifiers cannot be combined to get precision-recall pairs that interpolate linearly Davis & Goadrich (2006).

Sometimes, rather than calculate the AUPR curve exactly, it can be approximated with a measure called Average Precision (AP). Rather than doing a complex interpolation between two precision-recall points, Average Precision simply uses the precision of the rightmost point (the point with the higher recall).

4 Optimal Prediction Performance

Recall that a graph $G = (V, E)$, and $\bar{E}$ is the set of non-edges in G. Let $L : \bar{E} \to \{\texttt{Positive}, \texttt{Negative}\}$ be the correct labeling of those non-edges and let $P = \{e \mid e \in \bar{E} \wedge L(e) = \texttt{Positive}\}$ be the set of positives.

4.1 ROC, AUPR, and AP

We prove in the Appendix that the optimal ROC and AUPR scores a soft classifier can obtain equals the ROC/AUPR scores obtained from a classifier $\ell^*_G : \bar{E} \to \mathbb{R}$ which satisfies the following property:

$$\ell^*_G(e_1) \geq \ell^*_G(e_2) \leftrightarrow \frac{|\text{AO}_G(e_1) \cap P|}{|\text{AO}_G(e_1)|} \geq \frac{|\text{AO}_G(e_2) \cap P|}{|\text{AO}_G(e_2)|} \tag{1}$$

Amongst other things, this means that a classifier which correctly outputs the probabilities that an object is a Positive is an optimal classifier in these metrics. Note that if incorrect (*e.g.* trapezoidal) interpolation is used for AUPR calculation then this fact no longer holds true.

This property of optimal classifiers permits us to easily compute the maximal ROC/AUPR scores that any algorithm could have obtained on a given dataset and task. We provide code both for proper AUPR calculation and for optimal ROC/AUPR scores at [repository link to be added upon paper acceptance].

Let $\langle O_1, O_2, ..., O_{|\mathcal{AO}_{\bar{E},G}|}\rangle$ be an ordering of the automorphism orbits that respects the property in Equation 1. Further, let $p_0 = 0$ and $p_i = p_{i-1} + |P \cap O_i|$ for $1 \leq i \leq |\mathcal{AO}_{\bar{E},G}|$. Likewise let $t_0 = 0$ and $t_i = t_{i-1} + |O_i|$ for $1 \leq i \leq |\mathcal{AO}_{\bar{E},G}|$.

Once we can assume this ordering, we can simply apply the standard formulae for ROC and AUPR. Expressed in terms of the notation we are using, these formulae are as follows:

$$\text{Max ROC} = \sum_{i=0}^{|\mathcal{AO}_{\bar{E},G}|-1} \frac{p_{i+1} - p_i}{|P|} \cdot \frac{(t_{i+1} - p_{i+1}) + (t_i - p_i)}{2|\bar{E} \setminus P|} \tag{2}$$

$$\text{Max AUPR} = \sum_{i=0}^{|\mathcal{AO}_{\bar{E},G}|-1} \frac{p_{i+1} - p_i}{|P|} \cdot \frac{p_{i+1} - p_i}{t_{i+1} - t_i} \cdot \left(1 + \left(\frac{p_i}{p_{i+1} - p_i} - \frac{t_i}{t_{i+1} - t_i}\right) \cdot \ln\left(\frac{t_{i+1}}{t_i}\right)\right) \tag{3}$$

The AUPR formula comes from integrating over the precision-recall point interpolation technique discussed above in Sec. 3.2. Note that when $p_{i+1} - p_i = 0$ (*i.e.* $|P \cap O_{i+1}| = 0$), the formula inside the summation becomes zero; there are no divisions by zero from $p_{i+1} - p_i = 0$. Further $t_{i+1} - t_i = |O_{i+1}| > 0$. Lastly, when $t_i = 0$, then $p_i = 0$, and because $\lim_{x \to 0^+} x \ln \frac{1}{x} = \lim_{x \to 0^+} x\,(\ln(1) - \ln(x)) = 0$, we do not get a division by zero issue with t_i.

Equipped with these formulae, we can now begin to calculate the maximum possible performance scores on actual prediction tasks.

As we mentioned above, Average Precision (AP) is sometimes used to approximate AUPR. However, the nice result we prove for ROC and AUPR concerning equivalence class ordering does *not* hold for AP. Fortunately, our upper bound on AUPR is also an upper bound on AP, so we can still upper bound the AP scores that one might obtain. We provide a short proof of this in the appendix.

5 Methodology

Our main experiment is to calculate how maximum link prediction scores vary with the amount of information given to an idealized algorithm. We run this test on a wide variety of real-world graphs. The procedure runs as follows:

1. Begin with a graph $G = (V, E)$ and an edge removal probability p (we set $p \leftarrow 0.1$).
2. Define the set of negatives N as all edges not in G.
3. Remove each edge in G with probability p (independently) and add the removed edges to the set of positives P. Call the resulting graph $H \leftarrow (V, E \setminus P)$.
4. Get a (hashed) canonical representation for each non-edge's automorphism orbit in H.
5. Use the collected information to calculate the maximum scores via equations 2 and 3.
6. Assign $k \leftarrow 1$.
7. Get a (hashed) canonical representation of the k-hop neighborhood for each non-edge in H where the non-edge's endpoints are given a distinct color from the rest of the nodes.
8. Use the collected information to calculate the maximum scores when using at most k hops of information about a non-edge.
9. If the performance limit just obtained from step 8 is equal to (or within 0.005 of) the performance limit obtained from step 5, then stop. Otherwise, assign $k \leftarrow k + 1$ and go to step 7.

We perform the above procedure multiple times for each graph. Each iteration corresponds to different, random possible sets of missing edges; each set of missing edges can be slightly different in terms of the limit on its predictability. We get the mean value and 95% confidence interval for each distinct value of k.

We tested the link-prediction limits on a wide variety of real-world graphs. They are listed in Table 1.

6 Results

6.1 Sparsity Tends to Lower the Upper-Bound

We found that on most graphs, the upper bounds were near 100%, even when using 1-hop neighborhoods; we suspect that this is because when degrees are high enough there is still a large number of possible 1-hop neighborhoods such that the hypothetical optimal algorithm can take advantage of the slightest difference between neighborhoods. However, we found that on the sparsest graphs the results told a different and very interesting story.

We show the results for the four sparsest graphs: the Cora and Citeseer citation (sub)graphs, the CCSB-YI1 Protein-Protein Interaction graph, and a US Powergrid network. The results are in Figure 2. In particular, we focus on the AUPR values, because even though link prediction papers often report ROC scores, link predictors can easily get large ROC scores due to the class imbalance (the sheer number of non-edges) Yang et al. (2015).

In summary, our results give good evidence that when data becomes sparse enough, graph toplogy alone is severely limited in its ability to indicate a difference between genuine and fake missing edges.

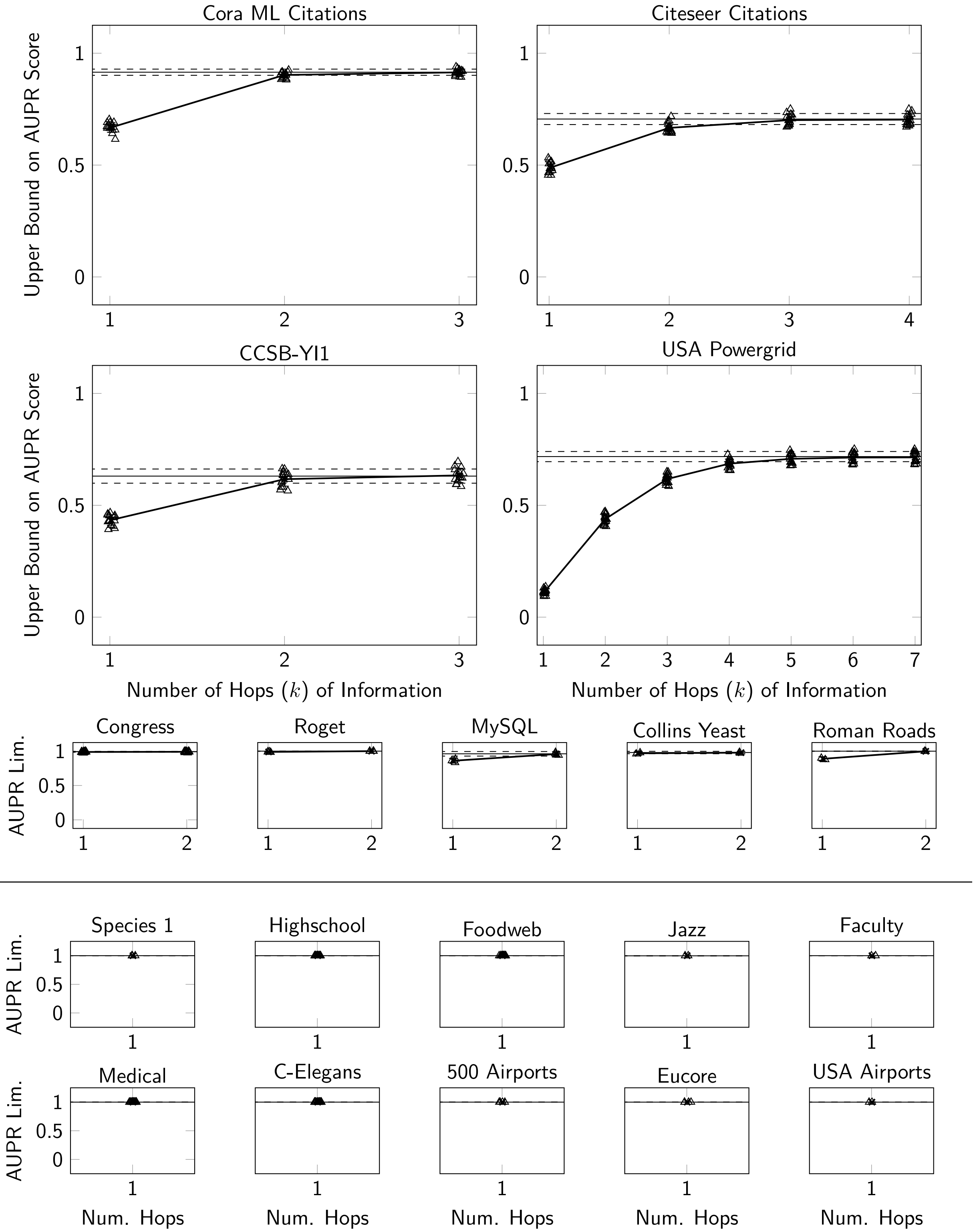

Figure 2: Hard upper bounds on link prediction performance as it varies with the amount of information given to a link prediction algorithm. The horizontal line shows the limit when using the entire graph ($k = \infty$). Ten percent of the graph's edges were randomly selected as test edges. The multiple points at a single value of k are from different sets of randomly chosen test edges; left-right jitter is employed to aid in visualization. Error bars are 95% confidence intervals.

Graph	(**Un**)**D**irected	(**Un**)**W**eighted	$\lvert V\rvert$	$\lvert E\rvert$	# Self-loops
Species 1 Brain [1]	D	U	65	1139	0
Highschool Friendships [2]	D	W	70	366	0
Foodweb [2]	D	U	183	2476	18
Jazz Collaboration [2]	U	U	198	5484	0
Faculty Hiring (C.S.)	D	W	206	2929	124
Congress Mentions [2]	D	W	219	586	2
Medical Innovation [2]	D	U	241	1098	0
C-Elegans Metabolic [2]	U	U	453	2025	0
USA Top 500 Airports (2002) [3]	D	U	500	5960	0
Eucore Emails [4]	D	U	1005	24929	642
Roget Concepts [3]	D	U	1010	5074	1
CCSB-YI1 [5]	U	U	1278	1641	168
MySQL Fn. Calls [6]	D	U	1501	4212	13
USA Airports (2010) [3]	D	U	1574	28236	0
Collins Yeast [3]	U	U	1622	9070	0
Cora Citation [1]	D	U	2708	5429	0
Citeseer Citation [1]	D	U	3264	4536	0
Roman Roads (1999) [3]	D	U	3353	8870	0
USA Powergrid [3]	U	U	4941	6594	0

Table 1: Graphs used for tests – The edge count does not include self-loops, which are listed separately – Sources: [1]: Rossi & Ahmed (2015), [2]: Kunegis (2013), [3]: Clauset et al. (2016), [4]: Leskovec & Krevl (2014), [5]: Yu et al. (2008), [6]: Myers (2003)

6.2 Negative Sampling Methodologies Produce Artificially High Scores

We were curious to see how these fundamental limits compared to recent reports of link-prediction performance. As a small case study, we considered the Graph Convolutional Neural Network Auto-Encoder (GCNAE), a widely used and referenced model that can perform topology-only link-prediction Kipf & Welling (2016b). In the original GCNAE paper, the authors tested their model on undirected versions of the Cora and Citeseer citation networks. They reported ROC scores and AP scores.

We found that AP scores they reported for the Citeseer network were well *above* our upper bound, indicating that there was a difference in the calculated AP. We suspect the difference is that in the GCNAE paper's tests the number of negative edges was downsampled, perhaps to one negative test edge per positive test edge. This sort of downsampling is common when performing link prediction evaluation with AP or AUPR; however downsampling tends to boost the AP and AUPR scores significantly relative to what they would have been if the full set of negatives was used in testing Yang et al. (2015). The GCNAE paper does not specify if (and/or how much) downsampling occurred.

The paper's reported ROC scores are well below our upper bound on ROC. This makes sense as the ROC score is not affected by downsampling Yang et al. (2015). If we downsample the number of negative edges to one negative edge per positive edge when calculating the AP limits, we get that the GCNAE's AP performance is also well below the upper bound. We show the numeric results in Table 2.

The point of this case study is twofold. Firstly, a metric (*e.g.* AP) may have different meanings depending on how it is used, and our methodology may be able to help retroactively determine which approach was used if the original paper does not specify.

Secondly, and perhaps of greater interest, state of the art link prediction systems using the topology of a network do not reach the topology-based upper limit on performance. We take this to suggest either that state of the art link prediction systems have room for improvement in their use of graph topology *or* that what structurally differentiates the k-hop neighborhoods of true edges from the k-hop neighborhoods of false edges in our tests is basically noise that an algorithm should not pay attention to if it wishes to generalize

Graph	GCNAE ROC	ROC Upper-Bound	GCNAE AP	AP Upper-Bound	AP Upper-Bound (1:1 Downsampling)
Cora	0.843 ± 2e-4	0.99992 ± 3e-5	0.881 ± 1e-4	0.903 ± 0.020	0.99999 ± 9e-6
Citeseer	0.787 ± 2e-4	0.9981 ± 3e-4	0.841 ± 1e-4	0.686 ± 0.019	0.9989 ± 5e-4

Table 2: Comparison to the GCNAE's Reported Results – The ± symbol indicates the 95% confidence interval. We conclude that the GCNAE paper is likely downsampling negative test edges in the process of calculating AP. More importantly, once downsampling is factored in, there is a notable gap between the hypothetical ideal performance and state of the art topology-based performance. We discuss this more in Sec. 6.2. Note: These results are for undirected versions of the graphs, whereas the results in Fig. 2 are for the directed versions (GCNAE only does link prediction on undirected graphs). Also, note that the version of Citeseer that the GCNAE paper used has some extra nodes with no links, whereas the version we used for Fig. 2 does not.

well. After all, if for example the 1-hop neighborhoods of two different non-edges both have 20 edges per neighborhood and differ in only one place, should we expect a link prediction algorithm to always treat that difference as significant? We propose some future work in Section 7.1 for exploring how the upper limit on performance changes when the resolution of the data is a bit blurrier, thereby reducing this noise.

6.2.1 Caveats

There are three caveats to our assessment that the GCNAE paper downsampled. Firstly, it is hypothetically possible that the authors could have gotten extremely lucky in terms of the set of edges randomly removed to form the test set. Our analysis does not exhaustively consider all possible test sets; it merely considers a number of randomly sampled test sets. However, we consider an explanation of the score in terms of a lucky test edge set rather unlikely given the narrow band of upper bounds our samples form and because the reported ROC scores are well below the upper limit – it would be difficult to get very lucky on the harsher metric (AP) but not the easier one (ROC).

The second caveat is that even when running in *featureless* mode (*i.e.* topology-only prediction), the GCNAE model is given each node's index as a feature. If the nodes' indices were randomly assigned, these features would simply be noise and would do nothing to help link-prediction performance. However, it is possible that the way the datasets were originally compiled ordered the nodes such that, for example, edges between nodes listed close together were more common in the dataset than they would have been if the dataset's node indices were randomly assigned.

The third caveat is that in a paper by the same authors with the same datasets, the authors specify that they add a self-loop to each node Kipf & Welling (2016a); if they allow those self-loop edges to be among the test set, those edges become relatively easy to predict.

7 Discussion

7.1 Applications, Extensions, and Limitations

In addition to the fact that our methodology gives insights about topology-based link prediction, we believe the kind of analysis we offer in this paper can be extended and expanded. We observed how maximum possible performance on a particular binary classification task (*i.e.* link prediction) varies with the amount of information available to the classifier. At a certain resolution, inputs to the algorithm look identical. In our analysis, the differing resolutions were the differing k for the k-hop neighborhood subgraphs. However, these resolutions could hypothetically be any reasonable representation of the data.

If these kinds of equivalence classes can be created at widely varying resolutions for a classification tasks, then researchers will begin to be able to say things like "our algorithm works as well on the full data as an optimal algorithm would work on data of resolution X."

The key ingredient for our analysis was that at a given resolution we were able to establish equivalence classes on the objects being classified (the non-edges); this let us calculate how well a hypothetical optimal algorithm would perform on those classes. We were able to get equivalence classes because our equality relation on two test objects was isomorphic equivalence of the objects' k-hop neighborhoods *and thus the relation was transitive*. Yet we expect that even in cases where a transitive equality relation is not immediately available, one could create such a relation by using a distance measure to cluster test inputs and then defining equality as being in the same cluster. The more fine-grained the clusters, the higher the resolution of data given to the hypothetical optimal algorithm.

The main limit we are aware of for this kind of analysis is that at high data resolution, noise can easily dominate the analysis. That is to say, at high resolution, random noise tends to render each entity to be classified unique, and thus the hypothetical, optimal algorithm with a perfect prior will be able to correctly distinguish any two entities and get a perfect score.

For example, consider link prediction on pure noise: That is, consider the process of randomly generating a graph where each edge is present independently with some probability p and then randomly hiding some fraction of the edges to create a link prediction task. Such a random graph will likely have no global symmetry Erdős et al. (1960), so at high data resolution (*e.g.* $k = 3$) every non-edge will be unique and thus the hypothetical, optimal algorithm with a perfect prior will obtain a perfect score, even though a real-world algorithm that does not mystically foreknow the answer can do no better than considering each edge to be equally likely, because that is in fact how the graph was constructed.

Fortunately for our kind of analysis, real-world algorithms are usually designed to ignore noise in the first place, so a data resolution that successfully filters out noise can simultaneously be relevant and provide a non-trivial upper bound on optimal performance.

7.2 Conclusion

We presented a methodology for calculating hard limits on how well a link prediction algorithm could perform when using structural information only. This helps analyze how much information graph structure does or does not provide for link prediction. We found that very sparse graphs give rise to significant inherent difficulties and therefore contain strong caps on optimal performance.

We also observed that a state of the art topology-based link prediction method performs well below the upper bound in some cases, which we believe either means that the link prediction algorithms have serious room for improvement *or* that our test sometimes picks up on "noise" that indeed differentiates edges from non-edges but which an algorithm should not be expected to pick up on because that noise would not behave in any consistent or infer-able manner. These observations prompted our discussion on further avenues of discovery and extensions of our methodology; we expect that an analysis similar to ours which finds a way to obtain performance upper bounds at varying degrees of blurring the noise would provide further insights.

Acknowledgments

[To be added upon paper acceptance.]

References

Andrés Abeliuk, Zhishen Huang, Emilio Ferrara, and Kristina Lerman. Predictability limit of partially observed systems. *Scientific reports*, 10(1):1–10, 2020.

Owen Astrachan. Bubble sort: an archaeological algorithmic analysis. *ACM Sigcse Bulletin*, 35(1):1–5, 2003.

Christoph Bandt and Bernd Pompe. Permutation entropy: a natural complexity measure for time series. *Physical review letters*, 88(17):174102, 2002.

Ginestra Bianconi, Richard K Darst, Jacopo Iacovacci, and Santo Fortunato. Triadic closure as a basic generating mechanism of communities in complex networks. *Physical Review E*, 90(4):042806, 2014.

Guido Boffetta, Massimo Cencini, Massimo Falcioni, and Angelo Vulpiani. Predictability: a way to characterize complexity. *Physics reports*, 356(6):367–474, 2002.

Aaron Clauset, Ellen Tucker, and Matthias Sainz. The colorado index of complex networks (2016). *URL https://icon. colorado. edu*, 2016.

Jörn Davidsen, Holger Ebel, and Stefan Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical review letters*, 88(12):128701, 2002.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, 2006.

Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5 (1):17–60, 1960.

Joshua Garland, Ryan James, and Elizabeth Bradley. Model-free quantification of time-series predictability. *Physical Review E*, 90(5):052910, 2014.

Emily M Jin, Michelle Girvan, and Mark EJ Newman. Structure of growing social networks. *Physical review E*, 64(4):046132, 2001.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016b.

Peter Klimek and Stefan Thurner. Triadic closure dynamics drives scaling laws in social multiplex networks. *New Journal of Physics*, 15(6):063008, 2013.

Jérôme Kunegis. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*, pp. 1343–1350, 2013. URL `http://dl.acm.org/citation.cfm?id=2488173`.

Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

Kian-Ping Lim, Weiwei Luo, and Jae H Kim. Are us stock index returns predictable? evidence from automatic autocorrelation-based tests. *Applied Economics*, 45(8):953–962, 2013.

Xin Lu, Erik Wetter, Nita Bharti, Andrew J Tatem, and Linus Bengtsson. Approaching the limit of predictability in human mobility. *Scientific reports*, 3(1):1–9, 2013.

Christopher R Myers. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. *Physical review E*, 68(4):046116, 2003.

Paul Resnick, Yuqing Kong, Grant Schoenebeck, and Tim Weninger. Survey equivalence: A procedure for measuring classifier accuracy against human labels. *arXiv preprint arXiv:2106.01254*, 2021.

Ryan Rossi and Nesreen Ahmed. The network data repository with interactive graph analytics and visualization. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015. URL `https://networkrepository.com`.

Samuel V Scarpino and Giovanni Petri. On the predictability of infectious disease outbreaks. *Nature communications*, 10(1):1–8, 2019.

Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015.

Haiyuan Yu, Pascal Braun, Muhammed A Yıldırım, Irma Lemmens, Kavitha Venkatesan, Julie Sahalie, Tomoko Hirozane-Kishikawa, Fana Gebreab, Na Li, Nicolas Simonis, et al. High-quality binary protein interaction map of the yeast interactome network. *Science*, 322(5898):104–110, 2008.

Fuqing Zhang, Y Qiang Sun, Linus Magnusson, Roberto Buizza, Shian-Jiann Lin, Jan-Huey Chen, and Kerry Emanuel. What is the predictability limit of midlatitude weather? *Journal of the Atmospheric Sciences*, 76(4):1077–1091, 2019.

A Appendix - Proofs

A.1 Maximum ROC

Let n be the number of distinct inputs to the classifier (*i.e.* the number of equivalence classes on the items to be labeled).

Further, let s be a classifier that achieves an optimal ROC score for these equivalence classes (hereafter referred to as classes - not to be confused with the True/False labels being predicted). Because ROC is calculated with respect to the different (true positive rate, false positive rate) pairs obtainable by using different score thresholds to convert a soft classifier into various hard classifiers, then without loss of generality s assigns a distinct score to each class; if s did not, then we could just consider the classes given the same scores as being the same class with a larger total size and larger number of positives.

Let $t_1, t_2, ..., t_n$ be the total sizes of the classes, where the classes are ordered by the score s gives to them (t_1 is the total size of the class with the highest score). Likewise, let $p_1, p_2, ..., p_n$ be the number of positives in the respective classes. For notational convenience, we define $t_0 = p_0 = 0$.

Let $T_i = \sum_{j=1}^{i} t_j$, $P_i = \sum_{j=1}^{i}$, and $N_i = T_i - P_i$. Also define $T = T_n$, $P = P_n$, and $N = N_n$.

Assume for sake of contradiction that there exists an $i \in [n-1]$ such that $\frac{p_i}{t_i} < \frac{p_{i+1}}{t_{i+1}}$. Now imagine an alternate classifier s^* that gives the exact same scores as s except that it reverses the scores for the i'th and $(i+1)$'th classes.

Then we get a new set of variables $t_1^*, t_2^*, ..., t_n^*$ and $p_1^*, p_2^*, ..., p_n^*$ as well as corresponding $T_j^*, P_j^*,$ and N_j^* where the values correspond to the classes as ordered by classifier s^*. Due to the definition of s^*, $t_j = t_j^*$ and $p_j = p_j^*$ for all j except $j \in \{i, i+1\}$, in which case $t_i = t_{i+1}^*$, $t_{i+1} = t_i^*$, $p_i = p_{i+1}^*$, and $p_{i+1} = p_i^*$. Once again, for notational simplicity we pad the beginning of these lists with $t_0^* = p_0^* = 0$.

This gives us a total of $n+1$ (true positive rate, false positive rate) pairs (i.e. (TPR, FPR) pairs). The false positive rate is the x axis of the curve and the true positive rate is the y axis.

$$\text{True Positive Rate } j \text{ for } s \text{ (i.e. TPR}_j) = \frac{P_j}{P} \text{ for } j \in \{0, 1, ..., n\}$$

$$\text{False Positive Rate } j \text{ for } s \text{ (i.e. FPR}_j) = \frac{N_j}{N} \text{ for } j \in \{0, 1, ..., n\}$$

Likewise for s^* we have:

$$\text{True Positive Rate } j \text{ for } s^* \text{ (i.e. TPR}_j^*) = \frac{P_j^*}{P} \text{ for } j \in \{0, 1, ..., n\}$$

$$\text{False Positive Rate } j \text{ for } s^* \text{ (i.e. FPR}_j^*) = \frac{N_j^*}{N} \text{ for } j \in \{0, 1, ..., n\}$$

The ROC curve then interpolates linearly between these points. Note that by definition $\text{TPR}_0 = \text{TPR}_0^* = \text{FPR}_0 = \text{FPR}_0^* = 0$ and $\text{TPR}_n = \text{TPR}_n^* = \text{FPR}_n = \text{FPR}_n^* = 1$.

We can consider the interpolation between the j'th (TPR, FPR) point and the $(j+1)$'th (TPR, FPR) point as corresponding to a variable $\alpha \in [0,1]$ where:

$$\text{TPR}_{j,\alpha} = \frac{P_j + \alpha p_{j+1}}{P}$$

$$\text{FPR}_{j,\alpha} = \frac{N_j + \alpha(t_{j+1} - p_{j+1})}{N}$$

This leads to the following:

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\text{TPR}_{j,\alpha} = \frac{p_{j+1}}{P}$$

and separately:

$$\alpha = \frac{N \cdot \text{FPR}_{j,\alpha} - N_j}{t_{j+1} - p_{j+1}}$$

$$\frac{\mathrm{d}}{\mathrm{dFPR}_{j,\alpha}}\alpha = \frac{N}{t_{j+1} - p_{j+1}}$$

Using the chain rule gives us:

$$\frac{\mathrm{d}}{\mathrm{dFPR}_{j,\alpha}}\text{TPR}_{j,\alpha} = \frac{\mathrm{d}\alpha}{\mathrm{dFPR}_{j,\alpha}} \cdot \frac{\mathrm{dTPR}_{j,\alpha}}{\mathrm{d}\alpha} = \frac{N}{P} \cdot \frac{p_{j+1}}{t_{j+1} - p_{j+1}}$$

For s^*'s curve this becomes:

$$\frac{\mathrm{d}}{\mathrm{dFPR}^*_{j,\alpha}}\text{TPR}^*_{j,\alpha} = \frac{N}{P} \cdot \frac{p^*_{j+1}}{t^*_{j+1} - p^*_{j+1}}$$

Now, because t_j and t^*_j only differ at $j = i$ and $j = i+1$, and the same holds for p_j, etc., and because the values at i and $i+1$ are the reverse of each other, then we can conclude that the $(i-1)$'th TPR-FPR point is the same for both s and s^* as well as the $(i+1)$'th point. The only difference is at the i'th point. To see that the $(i+1)$'th point is the same, note that $P_{i+1} = P_{i-1} + p_i + p_{i+1} = P^*_{i-1} + p^*_{i+1} + p^*_i = P^*_{i+1}$.

Further, because $\frac{p_i}{t_i} < \frac{p^*_i}{t^*_i}$ we obtain that $\frac{p_i}{t_i - p_i} < \frac{p^*_i}{t^*_i - p^*_i}$ which in turn means that:

$$\frac{\mathrm{d}}{\mathrm{dFPR}_{i-1,\alpha}}\text{TPR}_{i-1,\alpha} < \frac{\mathrm{d}}{\mathrm{dFPR}^*_{i-1,\alpha}}\text{TPR}^*_{i-1,\alpha}$$

In other words, the slope leading from the $(i-1)$'th point to the i'th point is greater in s^*'s ROC curve than in s's. Since both curves up to and including their $(i-1)$'th point are identical, and since they are also identical at the $(i+1)$'th point and thereafter, this means that s^*'s curve has a larger area underneath it.

Ergo, we obtain a contradiction, for s^* obtains a higher ROC score than s. Thus the assumption must have been false. This means that s orders the classes such that $\frac{p_j}{t_j} > \frac{p_{j+1}}{t_{j+1}}$ for all $j \in [n-1]$. In other words, s completely sorts the classes as we intended to show. ☐

A.2 Maximum AUPR

Davis and Goadrich have shown that if one ROC curve dominates another, then the corresponding (properly interpolated) precision-recall curves yield the same dominance Davis & Goadrich (2006). Thus the AUPR result follows directly from our ROC result above.

A.3 Maximum AP vs. Maximum AUPR

The nice ordering property we prove for ROC and AUPR does not hold for AP. For example, consider the following three equivalence classes with number of (positives, negatives) total: $\langle(10,0),(2,2),(9,7)\rangle$. Ordering these in decreasing $\frac{\text{Positives}}{\text{Positives}+\text{Negatives}}$ yields an AP of approximately 0.856 whereas ordering them in the order listed yields an AP of approximately 0.858.

Fortunately, we can still calculate a hard upper limit on AP scores by simply using the upper limit on the AUPR score. Remember our ordering rule for an optimal curve: Order by $\frac{\text{Positives}}{\text{Positives}+\text{Negatives}}$ in a descending order. Remember also that via the ROC and AUPR proofs, we showed that given *any* AUPR curve obtained by some ordering of the equivalence classes where some pair of classes disobeyed our ordering rule, you could get an AUPR curve that *dominates* it by swapping the ordering of the incorrect pair. Now, observe that any curve which does not follow the ordering rule can be converted into the curve that does by a succession of swaps where the swaps are of adjacent, incorrectly ordered classes; this corresponds to the naive "bubble sort" algorithm Astrachan (2003). During this process of swaps, each new curve dominates the former. Given that the last curve is the one corresponding to our ordering, it follows that our optimal curve not only has a larger area underneath it than any other curve, but it also *dominates* any other curve. Last but not least, note that if you follow the optimal AUPR curve from right to left (*i.e.* from recall of 1 to recall of 0) that the curve never decreases in precision.

Now let us turn our attention to AP curves. Recall that an AP curve is obtained in a very similar manner to an AUPR curve. In both cases you first get a collection of precision-recall points given the ordering the classifier gives to the equivalence classes (we call these precision-recall points the "base points"); the only difference between AP and AUPR is in how the two kinds of curves interpolate between adjacent base points. AP curves interpolate between any two precision-recall points by using the precision from the point with the higher recall.

We can observe two things: First, given our observations about the optimal AUPR curve, regardless of the ordering of the equivalence classes used to get the base points, all the base points of the AP curve are either on or are strictly dominated by the points in the optimal AUPR curve. Second, as you move from right to left along the interpolated AP points, the precision value remains constant until you hit a new base point. By contrast, as we discusse above the optimal AUPR curve's precision value is either constant *or increasing* as you move from right to left. Thus any AP curve is always dominated by the ideal AUPR curve. ☐