Estimating latent feature-feature interactions in large feature-rich graphs

Corrado Monti[1] and Paolo Boldi[1]

[1]Dipartimento di Informatica, Università degli Studi di Milano, Italy,
{monti,boldi}@di.unimi.it

Abstract

Complex networks arising in nature are usually modeled as (directed or undirected) graphs describing some connection between the objects that are identified with their nodes. In many real-world scenarios, though, those objects are endowed with properties and attributes (hereby called features). In this paper, we shall confine our interest to binary features, so that every node has a precise set of features; we assume that the presence/absence of a link between two given nodes depends on the features that the two nodes exhibit.

Although the situation described above is truly ubiquitous, there is a limited body of research dealing with large graphs of this kind. Many previous works considered homophily as the only possible transmission mechanism translating node features into links: two nodes will be linked with a probability that depends on the number of features they share. Other authors, instead, developed more sophisticated models (often using Bayesian Networks [30] or Markov Chain Monte Carlo [20]), that are indeed able to handle complex feature interactions, but are unfit to scale to very large networks.

We study a model derived from the works of Miller *et al.* [47], where interactions between pairs of features can foster or discourage link formation. In this work, we will investigate how to estimate the latent feature-feature interactions in this model. We shall propose two solutions: the first one assumes feature independence and it is essentially based on a Naive Bayes approach; the second one consists in using a learning algorithm, which relaxes the independence assumption and is based on perceptron-like techniques. In fact, we show it is possible to cast the model equation in order to see it as the prediction rule of a perceptron. We analyze how classical results for the perceptrons can be interpreted in this context; then, we define a fast and simple perceptron-like algorithm for this task. This approach (that we call *Llama*, Learning LAtent feature-feature MAtrix) can process hundreds of millions of links in minutes. Our experiments show that our approach can be applied even to very large networks.

We then compare these two techniques in two different ways. First we produce synthetic datasets, obtained by generating random graphs following the model we adopted. These experiments show how well the Llama algorithm can reconstruct latent variables in this model. These experiments also provide evidence that the Naive independence assumptions made by the first approach are detrimental in practice. Then we consider a real, large-scale citation network where each node (i.e., paper) can be described by different types of characteristics. This second set of experiments confirm that our algorithm can find meaningful latent feature-feature interactions. Furthermore, our framework can be used to assess how well each set of features can explain the links in the graph.

1 Introduction

The problem of finding a model that describes how complex networks shape their structure is well studied but still elusive in its full generality. In many scenarios, though, it is reasonable

to assume that the network arises in some way from a complex interweaving of some features of the nodes. For example, in a co-authorship network, a link stems more easily between authors with similar interests; similarly, in a genetic regulatory network, links are affected by the different biological functions of the regulators.

Many models have been proposed for describing complex network where arcs are influenced by some features of the nodes. For example, Lattanzi and Sivakumar [38] described a model where arcs form at random, or as a consequence of shared common features; Caldarelli *et al.* in [7] proposed a model where arcs are determined by an arbitrary function of the "fitness" of the nodes (i.e., a real-valued property possessed by each node). More models proposed along this line of research will be described in Section 2.

Although in some cases the relation between features and links is *homophily* (a link stems more easily between nodes that share a large portion of *the same* features), we would like to design a model that is able to capture also more complex behaviors. For example, feature h could foster links to feature k also when $h \neq k$: e.g., in the case of semantic relations, a concept tagged as belonging to the category "Movies" will often link to a concept tagged as belonging to "Directors". If we consider directed networks, we would like this relationship between features to be directed: feature h could foster links towards feature k but not the other way around. For example, in a citation network, we could easily expect a paper within the sociology realm to cite a statistics paper, but a link in the opposite direction will be much harder to find. Finally, some pairs of features could not foster but rather inhibit link formation: as *"Romeo and Juliet"* narrates, belonging to rival families could discourage the creation of a link in a long-term romantic relationship graph.

The theoretical model we are going to describe (based on the work by Miller, Griffiths and Jordan [47]) is able to represent all the aforementioned kinds of behavior within a unified framework, while at the same time being simple enough to be computationally useful and scalable, as we will show in the second part of this work. In this work, we will see how the estimation of the latent parameters of the model is fundamentally related to perceptron-like prediction rules, and we will turn this insight into a scalable algorithm able to extract information also from very large graphs.

In our model a special role is played by the feature-feature matrix $\mathbf{W}$. This matrix can express the various kinds of interplay between features and links, as described above; it is a latent, unobservable element of the model, that can compactly explain the observable links. The question is basically the following: assuming to know the links of a network and the features every node bears, how can we estimate how features interact with each other – i.e., estimating the matrix $\mathbf{W}$?

This question has a lot of practical implications. Consider for example a semantic graph [12], where nodes are concepts, arcs are semantic relations, and each concept can belong to different categories. Here, the matrix element $W_{h,k}$ describes how two categories h and k relate to each other: it summarizes if they interact positively, negatively and how much; it can therefore be used for measuring the semantic connection between the two categories. In a linguistic graph (maybe obtained from a large corpus of text), where a link exists between words used as subjects and those used as objects for a certain verb, $\mathbf{W}$ describes the semantic areas a given verb can connect. In a citation network where features are areas of scientific research, the set $S_k = \{h|W_{h,k} > 0\}$ contains the fields for which the field k is useful, and so forth.

Many other examples are possible; it is however important to note how many of these applications require to deal with graphs having a huge number of nodes and links. We will present concrete examples dealing with tens of millions of nodes. Operating at this scale demands new techniques; as we will see in Section 2, many of the existing techniques are not able to scale to this size.

A first idea, that we will describe in Section 4.1, is to just estimate the probability of a link from the category pairs we see in the data. We will derive formally this approach,

showing that it can be ascribed to the family of Naive Bayes learning. In particular, we will see that this estimation requires independence assumptions that are particularly unrealistic in most practical cases. For example, consider the semantic link between the entity *"Ronald Reagan"* and the 1954 Western film *"Cattle Queen of Montana"*; such an approach will increment, because of the presence of this link, the element of $\mathbf{W}$ corresponding to ($films$, *U.S. presidents*), regardless of the fact that this link could already be well explained by ($films$, $actors$).

Based on the latter observation, we will need to streamline the model: we will make it deterministic, by fixing its activation function ϕ. As we will describe in Section 4.2, this fact will allow us to see our model equation as the prediction rule of a perceptron and in the end to develop a more sophisticated approach based on online machine learning. What we will do is to see A (the links in the graph) as partially unknown, much like in the link prediction problem; we will show that, while learning A, the internal state of the perceptron will tend to $\mathbf{W}$. This approach, that we will call Llama – Learning LAtent MAtrix – will overcome the naive assumptions of the previous model.

In Section 5 we will test this approach on our model, by simulating graphs obeying the model and then observing how this way of reconstructing $\mathbf{W}$ behaves. More precisely, we will show how Llama is able to reconstruct the $\mathbf{W}$ matrix, and how instead the independence assumptions make the Naive algorithm very far from the goal.

Finally, in Section 6 we will consider a real, large-scale citation network, where nodes are papers and links represent citations. Here, each node can be described by different types of characteristics; we will consider institutions of the authors, and fields of research the paper belongs to. We will define a notion of *explainability*, a way to measure how a certain set of features can explain links in a graph according to our framework. Then, we will prove how real-data experiments confirm that our algorithm can find meaningful, latent feature-feature interactions from a real network.

2 Related works

The interplay between features and links in a network was investigated separately in different fields. Indeed, interpreting links as a result of features of each node has in fact a solid empirical background. For example, the dualism between "persons and groups" as an underlying mechanism for social connections was first investigated by Breiger [6] in 1974. Within sociology, the simple phenomenon of homophily – "similarity breeds connection" – received a great deal of attention: McPherson *et al.* [42] presented evidence and investigated on the role of homophily in social ties; considered features included race and ethnicity, social status, and geographical location. Bisgin *et al.* [3] studied instead the role of interests in online social media (specifically, Last.fm, LiveJournal, and BlogCatalog), finding however that the role of interests as features is weak on those online networks—at least when considering homophily only.

In some fields, behaviors more complex than homophily were considered as well. Tendencies of such kind, where nodes with certain features tend to connect to other types of nodes, are called *mixing patterns* in sociology and are often described by a matrix, where the element (i, j) describes the relationship between a feature i and a feature j. In epidemiology, mixing patterns have proven to be greatly beneficial in analyzing the spread of contagions. For example, they appeared to be a crucial factor in tracking the spread of sexual diseases [2] as well as in modeling the transmission of respiratory infections [49]. For this reason, such matrices are also called "Who Acquires Infection From Whom" (WAIFW) matrices, and have been empirically assessed in the field [27, 31]. In biology and bioinformatics, a seminal study by Menche *et al.* [45] highlighted the connections between the interactome (the network of the physical and metabolical interactions within a cell) and the diseases each component was associated with, observing a clustering of disease-associated proteins.

The empirical evidence presented in various fields, combined with the existence of large datasets available in the web, and the increase of computational resources, fostered some investigation of models of graph endowed with features.

Class models. A popular framework has been that of *latent class models*: in these models, every node belongs to exactly one class, and this class influences the links it may be involved into. The best-known example is the stochastic block model [50, 58]: in this model, it is assumed that each pair of classes has a certain probability of determining a link, and Snijders and Nowicki [58] study how to infer those probabilities; they also investigate how to determine the class assignments, leading to a sort of community detection algorithm. Hofman and Wiggins [30] devised a variant of this scheme, by specifying only within-class probabilities and between-class probabilities. Another useful adaptation involves sharing only the between-class probability and specifying instead the within-class probabilities separately for each class, allowing to characterize each with a certain degree of homophily. Both these approaches exemplify the need to reduce the number of parameters of the original block model, in order to facilitate the estimation of its parameters. Kemp *et al.* [33], and Xu *et al.* [62], studied and applied a non-parametric generalization of the model which allows for an infinite number of classes (therefore called *infinite relational model*). It permits application on data where the information about class is not provided directly. They use a Gibbs sampling technique to infer model parameters.

A well-known shortcoming of the class-based models is the proliferation of classes [47], since dividing a class according to a new feature leads to two different classes: if we have a class for "students" and then we wish to account for the gender too, we will have to split that class in "female students" and "male students". This approach is impractical and in many cases it leads to overlook significant dynamics. In order to overcome this limitation, some authors [1] extended classical class-based models to allow mixed membership. Here, the model of classes remains, but with a fuzzy approach: each node can be "split" among multiple classes, and in practice class assignments become represented by a probability distribution.

Feature models. Contrary to class-based models, *feature-based models* propose a more natural approach for nodes with multiple attributes: in those models, each node is endowed with a whole vector of features. Therefore, feature-based models can be seen as a generalization of class-based models: in fact, when all the vectors have exactly one non-zero component, the model has the same expressive power of class-based ones. Features can be real-valued – as in [29] – or binary, where the set of nodes exhibiting a feature is crisp, and not fuzzy, like in [44].

Many works in this direction proposed models that only allow for homophily, forbidding any other interaction among features. A seminal example is that of *affiliation networks* [38] by Lattanzi and Sivakumar; in that work, a social graph is produced by a latent bipartite network of *actors* and *societies*; links among actors are fostered by a connection to the same society. Gong *et al* [21] analyzed a real feature-rich social network – Google+ – through a generative, feature-based network model based on homophily.

Our attention will focus instead on models able to grasp more complex behavior than homophily, following the aforementioned empirical evidence from social networks, epidemiology and bioinformatics.

MAG model family. Within this stream of research, an important line of work has been explored by Kim, Leskovec and others [36], under the name of *multiplicative attribute graphs*. There, every feature is described by a two-by-two matrix, with real-valued elements. Those elements describe the probabilities of the creation of a link in all the possible cases of that feature appearing or not appearing on a given pair of nodes. As a consequence, it can be thought as a feature-rich special case of their previous Kronecker model [40]. This

model has been further extended to include many other factors; notably, they have modified it to be dynamic [37]: features can be born and die, and only alive features bear effects. However, the complexity of this model prevents it from being used on large-scale networks. The same authors have proposed [35] an expectation-maximization algorithm to estimate the parameters of their base model; nonetheless, reported experiments are on graphs with thousands of nodes at most. In the dynamic version, they report examples on hundreds of nodes (e.g., they find that by fitting the interactions of characters in a Lord of the Ring movie, their features effectively model the different subplots). In this work, instead, we wish to handle networks of much larger size: in the experimental part, we will show examples with many millions of nodes, for which we are able to estimate model parameters very efficiently.

MGJ model family. In 2009, Miller, Griffiths and Jordan [47] proposed a feature-based model to describe the link probability between two nodes by considering interactions between all the pairs of features of the two nodes. They show how by inferring features and their interactions on small graphs (hundreds of nodes), they are able to predict links with a very high accuracy (measured through the area under the ROC curve). The estimation technique they propose is not exact (since this would be intractable [23]), but it is based on a Markov Chain Monte Carlo (MCMC) method [20].

Their model can be interpreted as a generative model; they chose, however, not to investigate its structural properties in terms of the resulting network structure. Subsequent work [51] focused on this goal, being able to generate feature-rich graphs with realistic features, but they did not try to estimate the latent variables of the model necessary to predict links. In this work, we will build on the evidence gained in our previous work [5], that shows how the Miller-Griffiths-Jordan model (further extended to exhibit competition dynamics in feature generation) can be a powerful tool to generate networks with realistic, global properties (e.g. distance distribution, degree distribution, fraction of reachable pairs, etc.). As explained in previous work [5], this model can at the same time be used to synthesize realistic graphs by itself, or as a way to generate, given a real graph, a different one with similar characteristics.

Despite the capabilities of the MGJ model [47], however, the choice of using a MCMC technique in the original work [20] revealed itself inadequate to work on datasets larger than some hundreds of nodes. As noted by Griffiths *et al.* in 2010 [23], there is a need for computationally efficient models as well as reliable inference schemes for modeling multiple memberships. Menon and Elkan [46] noted how the inadequacy in handling large graphs underpinned this work, and many similar ones, and ascribed this flaw to the MCMC method.

There has been, since, a certain amount of studies on how to apply the MGJ model on larger graphs. The two aforementioned works, for example, tried to solve this problem in different ways. Griffiths and Ghahramani [23] described a simpler model: they removed from the original model [47] the possibility of having negative interaction between features; also, they fixed the activation function of the model (a component which we will carefully explain in the next section); in this way, they obtained a framework that is more computationally efficient, and can be applied to graphs of up to ten thousand nodes.

Menon and Elkan [46], instead, slightly enriched the model, by introducing a bias term for each node; then, they proposed a new estimation technique, based on stochastic gradient descent. A main focus there was to avoid undersampling non-links to overcome class imbalance, since despite it being "the standard strategy to overcome imbalance in supervised learning" it has the "disadvantage of necessarily throw[ing] out information in the training set". To overcome this problem (that we solve instead in the standard way of undersampling, see section 4), they design a sophisticated approach centered on the direct optimization of the area under the ROC curve.[1] With this technique, they can handle graphs with thou-

[1]Recent works [63] have indicated empirically as well as theoretically how employing this measure in link prediction leads to severely misleading results.

sands of nodes. A different approach, that obtained similar results on graphs of the same size, is [17], where the authors propose an SVM-based estimation of parameters, and report it being able to run on graphs as large as two thousands nodes in 42 minutes. Our approach runs in around 15 minutes on graphs three orders of magnitude bigger.

The task we are defining ultimately falls into the realm of latent variable models [18], since we are trying to explain a set of manifest variables – links and features – through a set of latent variables – the feature-feature interaction weights, i.e., the elements $W_{h,k}$ of the matrix $\mathbf{W}$. If, like in our case, manifest variables are categorical, we usually talk about *latent structure models*, that have been studied as such by statisticians and social scientist since the 1950's. Lazarsfeld started studying the statistics behind these models in an effort to explain people answers to psychological questions (specifically, in [59], answers from World War II soldiers) through quantifiable, latent traits [39]. These techniques were improved by later studies [22, 26]; however, these techniques—conceived for traditional social studies—were designed for small groups; the use cases described there do not usually involve more than a hundred of nodes. We require our techniques to work with millions of nodes, and hundreds of millions of links.

Previous literature has also treated linked document corpora, where features are the words contained in each document (e.g., [41] and [11]). In these works, authors build a link prediction model obtained from LDA, that considers both links and features of each node. However, the largest graphs considered in these works have about 10^3 nodes (with $\sim 10^4$ possible features), and they do not provide the time required to train the model. [25] developed an LDA approach explicitly tailored for "large graphs" — but without any external feature information for nodes: they rather reconstruct this external information from scratch; the largest graph they considered has about 10^4 nodes and 10^5 links, for which they report a running time of $45 - 60$ minutes.

In this work, we too will employ a model of the Miller-Griffiths-Jordan family, that we will thoroughly describe in Section 3. As mentioned above, a way to generate realistic graphs with this model was studied in [5]. Here, we will propose some further considerations on that model that will lead (in section 4) to various techniques aimed at estimating the main parameter of the model, i.e., the feature-feature matrix. We will test those methods on synthetic data generated by our model in section 5. In section 6 we will try our methods empirically on real networks whose size is unmatched by previous literature.

3 Our framework

Let us briefly present the main actors in our theoretical framework. In this work, we will treat the following objects as (at least partially) observable:

- The (possibly directed) graph $G = (N, A)$, where N is the set of n nodes, whereas $A \subseteq N \times N$ is the set of links; for the sake of simplicity, in this work we assume that self-loops (i.e., links of the form (i, i)) are allowed.
- A set F of m features.
- A node-feature association $Z \subseteq N \times F$.

We will denote these objects through their matrix-equivalent representation. More precisely, $G = (N, A)$ will be represented as a matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ (fixing some arbitrary ordering on the nodes); Z will be represented as a matrix $\mathbf{Z} \in \{0, 1\}^{n \times m}$ (again fixing some arbitrary ordering on the features). In the following, $A_{i,j}$ will refer to the element in the i-th row and j-th column of the matrix $\mathbf{A}$.

The – typically unobservable – objects that will define our network model will be the following:

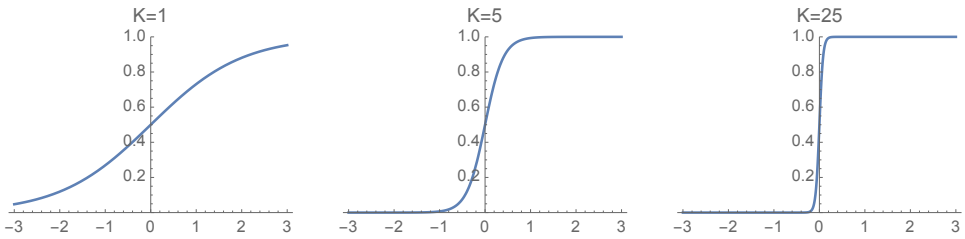


Figure 1: A sigmoid activation function ϕ, with different choices for K. K regulates its smoothness, and for $K \to \infty$ it approaches a step function.

- A matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$, that represent how features interact with each other. The idea is that a high value for $W_{h,k}$ means that the presence of feature h in a node i and of feature k in node j will foster the creation of a link from i to j. Conversely, a negative value will indicate that such a link will be inhibited by h and k. Naturally, the magnitude of $\left|W_{h,k}\right|$ will determine the force of these effects.

  We will refer to $\mathbf{W}$ as the *latent feature-feature matrix*.

- A monotonically increasing function $\phi : \mathbb{R} \to [0, 1]$ that will assign a probability to a link (i, j), given the real number resulting from applying $\mathbf{W}$ to the features of i and j; we will call such a function our *activation function*, in analogy with neural networks [28].

The relationship between those actors is described formally by the following equation, that fully defines our model:

$$\mathbb{P}\Big((i,j) \in A\Big) = \phi\Big(\sum_h \sum_k Z_{i,h} W_{h,k} Z_{j,k}\Big) \tag{1}$$

In other words, the probability of a link is higher when the sum of $W_{h,k}$ is higher, where h, k are all the (ordered) pairs of features appearing in the considered pair of nodes. We will now carefully detail this equation in the following sections.

3.1 Model parameters

Analysis of the latent feature-feature matrix. Let us point out how different choices for $\mathbf{W}$ can lead to many different kinds of interplay between links and features. The simplest case is $\mathbf{W} = I$ (the identity matrix). Since its only non-zero elements are those of the form (k, k), the only non-zero elements in the summation are those with $Z_{i,k} = Z_{j,k} = 1$. Therefore, the behavior of the model in this case is that of pure *homophily*: the more features in common, the higher the probability of a link (remember that ϕ is monotonic).

More generally, as we said, a positive entry $W_{h,k} > 0$ will indicate a positive influence on the formation of a link from nodes with feature h to nodes with feature k. In the special case of an undirected graph, we will have a symmetric matrix – that is, $W_{h,k} = W_{k,h}$ for all h and k.

$\mathbf{W}$ can be used to express also other behaviors. If $\sum_k W_{h,k}$ is large, this fact will indicate that nodes with feature h will be highly connected – specifically, they will have a large number of out-links. A large sum for a column of $\mathbf{W}$, that is a large value for $\sum_k W_{k,h}$, will imply, in turn, that nodes with feature h to have many in-links.

Choice of the activation function. The activation function ϕ will determine how the real numbers resulting from $\sum_h \sum_k Z_{i,h} W_{h,k} Z_{j,k}$ will be translated into a probability for the event $\big\{(i, j) \in A\big\}$. Since we require ϕ to be monotonically increasing, its role is just to *shape* the resulting distribution.

Throughout this work, and following previous literature [47], we will focus on activation functions that can be expressed as a sigmoid:

$$\phi(x) = \left(e^{K(\vartheta - x)} + 1\right)^{-1} \tag{2}$$

The parameter $\vartheta \in \mathbb{R}$ is the center of the sigmoid, whereas $K \in (0, \infty)$ regulates its smoothness. Figure 1 depicts how K influences the resulting probabilities (when $\vartheta = 0$). We will look at both these quantities as *a priori* parameters of the model. We will also extend the domain of K to the special value $K = \infty$, for which ϕ is the step function[2] $\chi_{(\vartheta,\infty)}$. Letting $K = \infty$ will make our model fully deterministic—all the probabilities become either 1 or 0. We will see how this simplification can turn our model into an important framework for mining information from a complex network.

3.2 An algebraic point of view

For some applications, it will be useful to consider the model expressed by (1) as a matrix operation. As introduced in the previous section, $\mathbf{Z}$ is the $n \times m$ node-feature indicator matrix.

With this notation, we can express (1) as

$$\mathbf{P} = \phi\Big(\mathbf{Z}\mathbf{W}\mathbf{Z}^T\Big) \tag{3}$$

where ϕ here denotes the natural element-wise generalization of our activation function — i.e., it simply applies it to all the elements of the matrix. The resulting matrix $\mathbf{P}$ is a matrix that describes the probabilities of $\mathbf{A}$: that is, its element $P_{i,j}$ defines the probability that $L_{i,j} = 1$ or equivalently that $(i, j) \in A$. You can think of P as an uncertain graph [34, 52], of which A is a realization (sometimes called a *world* [15]). Uncertain graphs are a convenient representation of graph distributions, in the same spirit as the classical Erdős-Rényi model: in an uncertain graph the node set is fixed and each arc has a certain probability of being present (arcs are independent from one another). Many useful statistical properties of the graph distribution associated to an uncertain graph (e.g., the expected number of connected components) can be connected to properties of the uncertain graph itself, seen as a simple weighted graph; it is this connection that made uncertain graphs particularly popular in some contexts.

While this view is simple and concise, it may be of little use from a computational perspective. In concrete applications n will be very large; also, algorithms that could be of use in dealing *directly* with this representation do not run in linear time—the most notable example being matrix factorization (e.g., computing the SVD [60]).

It is useful, however, to view (3) separately for each row of the matrix. In practice, this means computing the set of out-links of a single node. This operation allows us to treat a single node at a time, permitting the design of *online* algorithms, requiring a single pass on all the nodes.

Moreover, this interpretation renders $\mathbf{W}$ a (possibly asymmetric) similarity function: if we represent nodes i and j through their corresponding rows in $\mathbf{Z}$ (indicating them as $\mathbf{z}_i$ and $\mathbf{z}_j$) then our feature-feature matrix can be seen as a function that given these two vectors computes a real number representing a weight for the pair (i, j). In the special case of $\mathbf{W} = I$ this is the standard inner product $\langle \mathbf{z}_i, \mathbf{z}_j \rangle$; in this case the similarity of those two vectors is just the number of features they share, thus implementing homophily. Instead, for a general $\mathbf{W}$ this similarity is $\langle \mathbf{z}_i, \mathbf{z}_j \rangle_{\mathbf{W}}$ (although $\mathbf{W}$ is not necessarily symmetric or positive definite).

[2] We will use the notation

$$\chi_I(x) = \begin{cases} 1 & \text{if } x \in I \\ 0 & \text{if } x \notin I. \end{cases}$$

In this sense $\mathbf{W}$ can be seen as a function $W : 2^F \times 2^F \to \mathbb{R}$, that acts as a *kernel* for sets of features.

3.3 Intrinsic dimensionality and explainability

Every fixed graph G has a probability that depends on the feature-feature matrix and, of course, on Z, that is, on the choice of the features that we associate with every node, and ultimately on the set of features we choose.

Some sets of features will make the graph more probable than others; we might then say that the *explainability* is a property of the chosen set of features for a certain graph G. We will measure it in practice in some scenarios in the third, experimental, part of this work.

For the moment, let us point out that the number of features can be seen as an *intrinsic dimensionality* of the graph G: if the graph could be explained by our model without any error at all, then the same information of G is in fact contained in $\mathbf{Z}$ and $\mathbf{W}$. In that case, we might say that the out-links of node i (described in the graph by ℓ_i, the i-th row of the adjacency matrix $\mathbf{A}$) could be equally represented by $\mathbf{z}_i$, thus with a much smaller dimension: specifically, with a vector of m elements.

In fact, n is a natural upper bound for m. Let us use the nodes themselves as features (i.e., $F = V$), associating with every node i the only feature i (i.e, setting $\mathbf{Z} = I$). If $\mathbf{W} = A$ then the graph will be always perfectly explained: it would be enough to choose ϕ as the step function $\chi_{(0,\infty)}$ to make the results of our model identical to the graph, since

$$\mathbb{P}\Big((i,j) \in A\Big) = P_{i,j} = \left[\phi\left(\mathbf{Z}\mathbf{W}\mathbf{Z}^T\right)\right]_{i,j} = \phi(W_{i,j}) = \begin{cases} 1 & \text{if } (i,j) \in A \\ 0 & \text{otherwise.} \end{cases}$$

Naturally, this choice of features does not tell us much; in practice, we obviously want $m \ll n$. For this, we allow for the introduction of some degree of approximation; some links will be wrongly predicted by our model, because it will expect their categories to link to each other. We shall call this effect *generalization error*. We will see in experiments how it can be measured and how it is intimately connected with the explainability of a set of features in a graph.

3.4 Introducing normalization

Let us now present some interesting variants of the proposed model. In many real-world scenarios, we can speculate that not all features are created equal. For example, in the formation of a friendship link between two people, discovering that they both have watched a very popular movie may not give us much insight; knowing instead that they both have seen an underground movie that few people have appreciated could give to their friendship link a more solid background. In other words, in some cases rarest features matter more.

Column normalization. To implement this effect, we can normalize $\mathbf{Z}$ by column (recall that columns correspond to features) in our equation, defining

$$\overleftarrow{Z}_{i,h} = \frac{Z_{i,h}}{||Z_{-,h}||_p}$$

where $Z_{-,h}$ denotes the h-th column of $\mathbf{Z}$ and $||-||_p$ represents the ℓ^p norm, for some chosen p. The notation $\overleftarrow{\mathbf{Z}}$ is used to emphasize the fact that, if $p = 1$, this normalization yields a left-stochastic (i.e., column-stochastic) matrix. Each column can be seen in this case as a probability distribution among nodes, uniform on nodes having that feature and null on the

others. If we plug $\overleftarrow{\mathbf{Z}}$ in place of $\mathbf{Z}$ in (1), we obtain

$$\mathbb{P}\Big((i,j) \in A\Big) = \phi\Big(\sum_h \sum_k \overleftarrow{Z}_{i,h} W_{h,k} \overleftarrow{Z}_{j,k}\Big) = \phi\Big(\sum_h \sum_k \frac{Z_{i,h} W_{h,k} Z_{j,k}}{||Z_{-,h}||_p \cdot ||Z_{-,k}||_p}\Big) \qquad (4)$$

thus reaching the effect we wanted: inside the summation, rare features will bear more weight, and common features will be of lesser importance. This can also be seen as an adaptation of a tf-idf-like schema [61] to our context.

Row normalization. In other contexts, row normalizations might be desirable instead. The fact that two people x and y are friends of the same individual z in Facebook may be a sign indicating that they have some common interest, and that they may become friends in the future; however, if x is a public figure then the fact that he is friend with z is not really significant, and does not tell us much about possible future friendship with y. In other words, nodes with few features may matter more.

Formally, row normalization is defined as

$$\overrightarrow{Z}_{i,h} = \frac{Z_{i,h}}{||Z_{i,-}||_p}.$$

where $Z_{i,-}$ is the i-th row of $\mathbf{Z}$. Again, we used the notation $\overrightarrow{\mathbf{Z}}$ because when $p = 1$ we obtain a right-stochastic matrix.

4 Inferring feature-feature interaction

The fundamental agent in shaping the graph in our framework is, as stated in the previous section, the feature-feature matrix $\mathbf{W}$. In many applications, however, the information represented by $\mathbf{W}$ is not directly available: in a social network, we can observe friendship links and characteristics of each person, but the relationship between the characteristics is latent and not observable. This is the case for many other scenarios: in a linked document corpora where documents are described by a set of topics, we do not know how different topics foster or discourage links. Knowing (at least partially) links and features of each node, but ignoring how features interact with each other, is also a common trait of all the examples we mentioned before.

As discussed in Section 1, knowing the latent feature-feature matrix has a lot of practical implications: it can summarize effectively how features interact with each other – in the case of a semantic network tagged with categories, it means getting a hold of which categories are semantically connected, for a citation network it means being able to identify which fields of research are being useful for a certain field, and so on. More generally, as we discussed in Section 3, knowing $\mathbf{W}$ means being able to represent all the information expressed by the graph in a more succinct way.

The problem we wish to solve is therefore the following: assuming to know A and $\mathbf{Z}$, how can we reconstruct a plausible $\mathbf{W}$? In other words: if we know the arcs in a graph, and each node is characterized by a set of (binary) features, how can we estimate how features interact with each other?

4.1 A naive approach

Let us first describe a naive approach to construe the latent feature-feature matrix $\mathbf{W}$; remember that we are assuming (1), where Z, A and ϕ are fixed (the role of ϕ will be discussed below) and we aim at choosing $\mathbf{W}$ as to maximize the probability of A.

More precisely, we shall use a naive Bayes technique [4], estimating the probability of existence of a link through maximum likelihood and assuming independence between features;

that is, we are going to assume that the events $\{Z_{i,h} = 1\}$ and $\{Z_{i,k} = 1\}$ are independent for h and k.

Let us introduce the following notation:

- let $N_k \subseteq N$ be the set of nodes with the feature k, i.e. $N_k = \{i \in N | Z_{i,k} = 1\}$;
- conversely, let us write F_i for the set of features sported by a node i, that is $F_i = \{k \in F | Z_{i,k} = 1\}$;
- let us also use $\mathcal{Z}_{i,k}$ to denote the event $\{Z_{i,k} = 1\}$.

Now, fixing two features h and k, let us consider the probability $p_{h,k}$ that there is a link between two arbitrary nodes with those features, such as $i \in N_h$ and $j \in N_k$:

$$p_{h,k} := \mathbb{P}\Big((i,j) \in A \Big| \mathcal{Z}_{i,h} \cap \mathcal{Z}_{j,k}\Big).$$

Said otherwise, $p_{h,k}$ represents the probability that two nodes (i,j) happen to be connected, if we assume that i has feature h and j has feature k. This quantity can be estimated as the fraction of pairs (i,j) such that both $\mathcal{Z}_{i,h}$ and $\mathcal{Z}_{j,k}$ are true, that happen to be links. In other words,

$$p_{h,k} = \frac{|(N_h \times N_k) \cap A|}{|N_h| \cdot |N_k|}$$

Here, and in the following, we are assuming that self-loops are allowed. For a specific pair of nodes (i,j), the probability of the presence of a link under the full knowledge of $\mathbf{Z}$ is given by

$$\mathbb{P}\Big((i,j) \in A \Big| \big(\bigcap_{h \in F_i} \mathcal{Z}_{i,h}\big) \cap \big(\bigcap_{h \in F_j} \mathcal{Z}_{j,h}\big)\Big).$$

This is the probability that (i,j) are connected, given that we know their common features. Let us naively assume that $\mathcal{Z}_{i,h}$ and $\mathcal{Z}_{j,k}$ are independent for all i, j, h, k with $i \neq j$ and $h \neq k$; we also assume that they are independent even under the knowledge that $(i,j) \in A$. Then, under these naive independence assumptions, the last probability can be expressed as

$$\prod_{h \in F_i} \prod_{k \in F_j} \mathbb{P}\Big((i,j) \in A \Big| \mathcal{Z}_{i,h} \cap \mathcal{Z}_{j,k}\Big) = \prod_{h \in F_i} \prod_{k \in F_j} p_{h,k}$$

Let us define $\mathbf{W}$ as:

$$W_{h,k} = \log \frac{|(N_h \times N_k) \cap A|}{|N_h| \cdot |N_k|} \tag{5}$$

We will now check that such a matrix is correct. Considering again the definition of our model (1) and plugging in the matrix $\mathbf{W}$ just defined, we obtain:

$$\begin{aligned}
\mathbb{P}\Big((i,j) \in A\Big) = \phi\Big(\sum_{h \in F_i} \sum_{k \in F_j} W_{h,k}\Big) = \phi\Big(\log \prod_{h \in F_i} \prod_{k \in F_j} \frac{|(N_h \times N_k) \cap A|}{|N_h| \cdot |N_k|}\Big) = \\
= \phi\Big(\log \prod_{h \in F_i} \prod_{k \in F_j} p_{h,k}\Big) = \phi\Big(\log \mathbb{P}\big((i,j) \in A \big| \big(\bigcap_{h \in F_i} \mathcal{Z}_{i,h}\big) \cap \big(\bigcap_{h \in F_j} \mathcal{Z}_{j,h}\big)\big)\Big) = \\
= \phi\Big(\log \mathbb{P}\big((i,j) \in A \big| \mathbf{Z}\big)\Big)
\end{aligned} \tag{6}$$

This fact confirms that, for a certain choice of ϕ (namely[3], $\phi(x) = \min(1, e^x)$) and under the previously mentioned independence assumptions, this estimate of $\mathbf{W}$ is correct for our model.

4.2 A perceptron-based approach

The independence assumptions behind the naive approach (hereby referred to as NAIVE) are not realistic. One of the potentially undesirable consequences of such assumptions is that the responsibility for the existence of a link are shared among all the features of the two involved entities. To understand how misleading this approach can be, consider a semantic link between the entity for "Ronald Reagan" and the one for *"Cattle Queen of Montana"* (a 1954 Western film starring Ronald Reagan). NAIVE will count that link as a member of the set $\{(N_{presidents} \times N_{movies}) \cap A\}$, and it will consequently increase the corresponding entry $(W_{presidents,\ movies})$ in the feature-feature matrix. We would like instead to design an algorithm that is able to recognize that this link is already well explained by the matrix element $W_{actors,\ movies}$ and that does not enforce a false association between politicians and Western movies. In other words, we want an algorithm that perceives if some feature is already explaining a link, and updates its estimate of $\mathbf{W}$ only if it is not. In this perspective, we want to properly cast our problem in the setting of machine learning.

A deterministic model. In order to obtain this result, let us simplify our framework by letting ϕ be the step function $\chi_{(0,\infty)}$, that is

$$\phi(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

This can be also seen as a sigmoid (2) whose parameters are $\vartheta = 0$ and $K \to \infty$. In previous work [5], we found that, even if such an activation function produces a more disconnected network, the network degree distribution will converge even more sharply to a power law.

It is important to note that this choice will make our model fully deterministic. In other words, given the complete knowledge of $\mathbf{Z}$ and $\mathbf{W}$, the model will not allow for any missing or wrong link. For this reason, with this model we can not measure the *likelihood* of a real network; instead, we will just separate its links into *explained* and *unexplained* by the model with respect to a certain set of feature F.

A decision rule. By using this deterministic activation function, the equation of our model (1) becomes:

$$(i,j) \in A \iff \sum_h \sum_k Z_{i,h} Z_{j,k} W_{h,k} > 0 \tag{7}$$

Let us indicate the i-th row of $\mathbf{Z}$ with $\mathbf{z}_i$ (as a column vector), the outer product with $\otimes$ and the Hadamard product with $\circ$. Then, we can alternatively write the above rule in one of these two equivalent forms:

$$(i,j) \in A \iff \mathbf{z}_i^T \mathbf{W} \mathbf{z}_j > 0$$

or

$$(i,j) \in A \iff \sum_{h,k} \Big[\big(\mathbf{z}_i \otimes \mathbf{z}_j \big) \circ \mathbf{W} \Big]_{h,k} > 0 \tag{8}$$

[3]We need the min in this formula to respect our assumption that ϕ only has values in $[0, 1]$. However, it does not change anything in practice, since in (6) the argument of the logarithm is a probability, and therefore it is forced to be in $[0, 1]$.

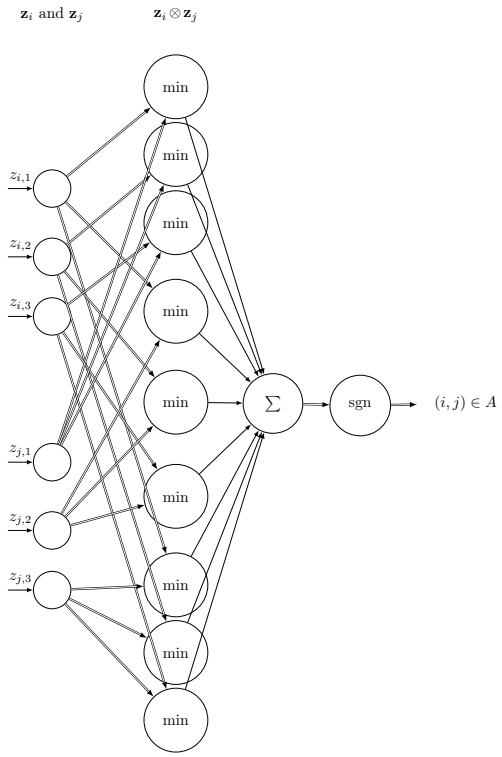


Figure 2: A neural-network view of the perceptron-like algorithm, for the case of $m = 3$ features. We indicate fixed weights with double lines, with min those nodes activating only if and only if both input nodes are active (that is, the min of their inputs), and with sgn the sign function. The only non-fixed weights (learned by the perceptron update rule) are those from the $\mathbf{z}_i \otimes \mathbf{z}_j$ layer to the $\sum$ neuron: they correspond to the matrix $\mathbf{W}$ appearing in our model.

4.2.1 A perceptron.

Equation (8) is in fact a special case of the *decision rule* of a perceptron [53], the simplest neural network classifier. The idea here is that by learning how to separate links from non-links (in fact a form of link prediction), the classifier infers $\mathbf{W}$ as its internal state.

Let us briefly recall the standard definition: a perceptron is a binary classifier whose internal state is represented by a vector[4] $\mathbf{w} \in \mathbb{R}^p$, and it classifies an instance $\mathbf{x} \in \mathbb{R}^p$ as positive if and only if $\text{sgn}(\mathbf{w} \cdot \mathbf{x}) > 0$.

The internal state $\mathbf{w}$ is typically initialized at random; then, during the learning phase, for each $i \in \{0, 1, \ldots, t-1\}$:

1. the perceptron observes an *example* $\mathbf{x}_i \in \mathbb{R}^p$;
2. it emits a *prediction* $\hat{y}_i = \text{sgn}(\mathbf{w} \cdot \mathbf{x}_i)$;
3. it receives the *true label* $y_i \in \{-1, 1\}$;
4. if $y_i \neq \hat{y}_i$, it updates its internal state with $\mathbf{w} = \mathbf{w} + y_i \lambda \mathbf{x}_i$, where $\lambda \in (0, 1]$ is a parameter called *learning rate*.

[4]For the purposes of this paper, we limit ourselves to describing perceptrons with null bias.

The key point here is that the decision rule for emitting a prediction can be cast to be fundamentally the same as in our model. Specifically, if we view the latent feature-feature matrix $\mathbf{W}$ as a vector of length m^2, and we do the same for $\mathbf{z}_i \otimes \mathbf{z}_j$, then we can see that the decision rule $\text{sgn}(\mathbf{w} \cdot \mathbf{x}_i) = 1$ corresponds to (8), if we set $\mathbf{W}$ as the vector $\mathbf{w}$ and $\mathbf{z}_i \otimes \mathbf{z}_j$ as the example $\mathbf{x}$.

Note that in our case an *example* for the perceptron will be a pair of nodes (i, j), represented not by a vector but by the $m \times m$ matrix $\mathbf{z}_i \otimes \mathbf{z}_j$: this is a matrix whose element $[\mathbf{z}_i \otimes \mathbf{z}_j]_{h,k}$ is 1 if and only if the first node exhibits the feature h and the second exhibits the feature k. This trick is sometimes called the *outer product kernel*: we are embedding a pair of vectors of dimension $2m$ into a higher-dimensional representation of dimension m^2. This $m \times m$-matrix in fact can be alternatively thought of as a vector of size m^2, allowing us to use such vectors as training examples for the perceptron, where the label is $y = 1$ if and only if $(i, j) \in A$, and $y = -1$ otherwise. The learned vector $\mathbf{w}$ will be, if seen as a matrix, the desired $\mathbf{W}$ appearing in (7), as we are going to analyze next.

To recap, the perceptron we are going to use operates like this: given a T sequence of pairs of nodes (elements of $N \times N$):

1. the perceptron observes the next pair $(i, j) \in T$, through their binary feature vectors $(\mathbf{z}_i,\ \mathbf{z}_j)$;
2. it computes a prediction on whether they form a link, according to (8); more precisely, the prediction will be $\hat{y}_{i,j} = \text{sgn}(\mathbf{z}_i^T \mathbf{W} \mathbf{z}_j)$
3. it receives the ground-truth: $y_{i,j} = 1$ if $(i, j) \in A$, and $y_{i,j} = -1$ otherwise;
4. if the prediction was wrong, the updates its internal state by adding to $\mathbf{W}$ the quantity $y_{i,j} \lambda (\mathbf{z}_i \otimes \mathbf{z}_j)$.

In doing this, we are using m^2 features, in fact a kernel projection of a space of dimension $2m$ into the larger space of size m^2. Similarly, the weight vector to be learned has size m^2. Positive examples are those that correspond to existing links. We can view this as a shallow, simple neural network, as depicted in Figure 2.

Interpretation of the error bound. One advantage of casting our approach to the perceptron algorithm is that the latter is a well studied and its performance was analyzed in all details. In particular, many bounds on its accuracy are known: let us consider the bounds discussed[5] in [10, Theorem 12.1]. Casting it to our case, some easy manipulations get the following bound for the number of misclassifications $M = |\{(i, j) \in T \ s.t.\ \hat{y}_{i,j} \neq y_{i,j}\}|$:

$$M \leq \inf_{\mathbf{U} \in \mathbb{R}^{m \times m}} \left(H(\mathbf{U}) + \left(R \|\mathbf{U}\| \right)^2 + R \|\mathbf{U}\| \sqrt{H(\mathbf{U})} \right) \tag{9}$$

where $\| - \|$ denotes the Frobenius norm and

- $H(\mathbf{U}) = \sum_{(i,j) \in T} \max\left(0, 1 - \mathbf{z}_i^T \mathbf{U} \mathbf{z}_j\right)$ is the sum of the so-called *hinge losses* and
- $R = \max_{(i,j) \in T} \|\mathbf{z}_i \otimes \mathbf{z}_j\|$ is called the *radius* of the examples.

Let us try to give an interpretation of this bound, by looking at all factors affecting the number M of errors of the algorithm. In the following, we want to use the bound above to compute the number of misclassification which we undergo using (8). For this purpose, let us set $\mathbf{U} = \mathbf{W}$ as in (8). Suppose also, for the sake of simplicity, that $T = A$ (that is, that we are using all and only the links as examples). We can define two subsets of T:

$$\begin{aligned} E_{\mathbf{U}} &= \left\{ (i, j) \in A \;\middle|\; \mathbf{z}_i^T \mathbf{U} \mathbf{z}_j \leq 0 \right\} \\ B_{\mathbf{U}} &= \left\{ (i, j) \in A \;\middle|\; 0 < \mathbf{z}_i^T \mathbf{U} \mathbf{z}_j < 1 \right\}. \end{aligned}$$

[5] In fact, for the sake of simplicity we are considering only Euclidean norm and standard hinge loss.

The set $E_{\mathbf{U}}$ contains the examples that are *incorrectly classified* (i.e., those which are not classified as links according to (8)); the set $B_{\mathbf{U}}$ contains the examples that are correctly classified but with a very small margin. We have that

$$H(\mathbf{U}) = \sum_{(i,j)\in A} \max\left(0, 1 - \mathbf{z}_i^T \mathbf{U} \mathbf{z}_j\right) = \sum_{(i,j)\in E_{\mathbf{U}} \cup B_{\mathbf{U}}} \left(1 - \mathbf{z}_i^T \mathbf{U} \mathbf{z}_j\right) \leq (1+a)\,|E_{\mathbf{U}}| + b\,|B_{\mathbf{U}}|\,, \quad (10)$$

for some $a, b > 0$ with $b < 1$. In other words, the term $H(\mathbf{U})$ in the right-hand-side of (9) is connected with the amount of misclassifications and borderline-correct classifications: each misclassification has a cost that is larger than one, whereas borderline-correct classifications are paid less than one each. In a way, $H(\mathbf{U})$ is a measure of how well our model could fit *in the best case* this particular feature-rich graph.

One way to reduce the number of borderline-correct classifications would be to multiply $\mathbf{U}$ by a constant larger than one: note that this operation does *not* change the classification of (8), but at the same time it increases the cost of misclassifications (the coefficient a of (10)) *and* the norm of $||\mathbf{U}||$, that also appears on the right-hand-side of (9). The presence of $||\mathbf{U}||$ in the bound is explained by the fact that a model with a large norm is (apart from scaling) more complex: e.g., a very sparse $\mathbf{U}$ (one where only a few pairs of features interact) will have a very low norm.

The last term appearing in (9) is R^2, that can be rewritten as

$$R^2 = \max_{(i,j)\in T} \sum_h \sum_k z_{i,h} z_{j,k} = \max_{(i,j)\in T} |F_i| \cdot |F_j|.$$

In other words, it measures *how many pairs of features* we need to consider in our set of examples. More precisely, this is the number of possible pairs among the features of the source and the target of each arc. Of course $R^2 \leq m^2$: this fact means that the bound is smaller if we need less features to explain the graph. It is also small if there is little overlap of features (i.e., if $\max_{i\in N} |F_i|$ is small).

In the case of a feature-rich graph that can be perfectly explained by a latent feature-feature matrix $\mathbf{W}$ (according to our deterministic model), we have $H(\mathbf{W}) = 0$. In this case, in fact, all the elements of the sum (that is, the losses suffered by the algorithm) would be null. This can be seen using for example the inequality given in (10): the set $|E_{\mathbf{W}}|$ would be empty, and the same can be said for $|B_{\mathbf{W}}|$, possibly scaling $\mathbf{W}$ by a constant. In this special case, the bound simplifies to $M < \left(R\|\mathbf{W}\|\right)^2$. This is the perceptron convergence theorem [54], which in our case tells us that if a perfect $\mathbf{W}$ exists, the algorithm will converge to it.

4.2.2 A passive-aggressive algorithm

Online learning. In general, what we did was to recast our goal in the framework of online binary classification. Binary classification, in fact, is a well-known problem in supervised machine learning; *online* classification simplifies this problem by assuming that examples are presented in a sequential fashion and that the classifier operates by repeating the following cycle:

1. it observes an example;
2. it tries to predict its label;
3. it receives the true label;
4. it updates its internal state consequentially, and moves on to the next example.

Algorithm 1
LLAMA, the passive-aggressive algorithm to build the latent feature-feature matrix $\mathbf{W}$.

INPUT:
The graph $G = (N, A)$, with $A \subseteq N \times N$
Features $F_i \subseteq F$ for each node $i \in N$
A parameter $\kappa > 0$
OUTPUT:
The feature-feature latent matrix $\mathbf{W}$

1. $\mathbf{W} \leftarrow \mathbf{0}$
2. Let $(i_1, j_1), \ldots, (i_T, j_T)$ be a sequence of elements of $N \times N$.
3. For $t = 1, \ldots, T$
   (a) $\rho \leftarrow 1/(|F_{i_t}| \cdot |F_{j_t}|)$
   (b) $\mu \leftarrow \sum_{h \in F_{i_t}} \sum_{k \in F_{j_t}} W_{h,k}$
   (c) **If** $(i_t, j_t) \in A$
   $\delta \leftarrow \min(\kappa, \max(0, \rho(1 - \mu)))$
   else
   $\delta \leftarrow -\min(\kappa, \max(0, \rho(1 + \mu)))$
   (d) For each $h \in F_{i_t}$, $k \in F_{j_t}$:
   $W_{h,k} \leftarrow W_{h,k} + \delta$

An online learning algorithm, generally, needs a constant amount of memory with respect to the number of examples, which allows one to employ online algorithms in situations where a very large set of voluminous input data is available. A survey is available in [9].

A well-known type of online learning algorithms are the so-called perceptron-like algorithms. They all share the same traits of the perceptron: each example must be a vector $\mathbf{x}_i \in \mathbb{R}^p$; the internal state of the classifier is also represented by a vector $\mathbf{w} \in \mathbb{R}^p$; the predicted label is $y_i = sign(\mathbf{w} \cdot \mathbf{x}_i)$. The algorithms differ on how $\mathbf{w}$ is built. However, since their decision rule is always the same, they all lead back to the decision rule of our model (8). This observation allow us to employ *any* perceptron-like algorithm for our purposes.

Perceptron-like algorithms (for example, ALMA [19] and Passive-Aggressive [13]) are usually simple to implement, provide tight theoretical bounds, and have been proved to be fast and accurate in practice.

A Passive-Aggressive algorithm. Among the existing perceptron-like online classification frameworks, we will heavily employ the well-known Passive-Aggressive classifier, characterized by being extremely fast, simple to implement, and shown by many experiments [8, 48] to perform well on real data.

Le us now describe the well-known Passive-Aggressive algorithm [13], while showing how to cast this algorithm for our case. To do this let us consider a sequence of pairs of nodes

$$(i_1, j_1), \ldots, (i_T, j_T) \in N \times N$$

(to be defined later). Define a sequence of matrices $\mathbf{W}^0, \ldots, \mathbf{W}^T$ and of slack variables $\xi_1, \ldots, \xi_T \geq 0$ as follows:

- $\mathbf{W}^0 = \mathbf{0}$

- $\mathbf{W}^{t+1}$ is a matrix minimizing $\|\mathbf{W}^{t+1} - \mathbf{W}^t\| + \kappa\xi_{t+1}$ subject to the constraint that

$$y_{i_t,j_t} \cdot \sum_{h \in F_{i_t}} \sum_{k \in F_{j_t}} W_{h,k}^{t+1} \geq 1 - \xi_{t+1}, \tag{11}$$

  where, as before

$$y_{i_t,j_t} = \begin{cases} -1 & \text{if } (i,j) \notin A \\ 1 & \text{if } (i,j) \in A \end{cases},$$

  $\|-\|$ denotes again the Frobenius norm and κ is an optimization parameter determining the amount of aggressiveness.

The intuition behind the above-described optimization problem, as discussed in [13], is the following:

- the left-hand-side of the inequality (11) is positive if and only if $\mathbf{W}^{t+1}$ correctly predicts the presence/absence of the link (i_t, j_t); its absolute value can be thought of as the confidence of the prediction;
- we would like the confidence to be at least 1, but allow for some error (embodied in the slack variable ξ_{t+1});
- the cost function of the optimization problem tries to keep as much memory of the previous optimization steps as possible (minimizing the difference with the previous iterate), and at the same time to minimize the error contained in the slack variable.

By merging the Passive-Aggressive solution to this problem with our aforementioned framework, we obtain the algorithm described in Algorithm 1. We will refer to this algorithm as LLAMA: *Learning LAtent MAtrix*.

Normalization. For perceptron-like algorithms, normalizing example vectors (in our case, the matrix $\mathbf{z}_i \otimes \mathbf{z}_j$) often gives better results in practice [14]. This is equivalent to using the ℓ^2-row-normalized version of our model, as discussed in Section 3.4 (setting $p = 2$). The assumption behind that model is in fact that nodes with fewer features provide a stronger signal for the small set of features they have; nodes with many features bear less information about those feature.

It is immediate to see that Algorithm 1 can be adapted to use the ℓ^2-row-normalization by changing step (c) to:

$$\begin{aligned} &\text{(c) } \textbf{If } (i_t, j_t) \in A\text{:} \\ &\quad \delta \leftarrow \sqrt{\rho}\min(\kappa, \max(0, 1 - \sqrt{\rho}\mu)) \\ &\textbf{else}\text{:} \\ &\quad \delta \leftarrow -\sqrt{\rho}\min(\kappa, \max(0, 1 + \sqrt{\rho}\mu)) \end{aligned} \tag{12}$$

Similar adaptations would allow one to implement *any* row normalization.

Sequence of pairs. Finally, let us discuss how to build the sequence of examples. We want $\mathbf{W}$ to be built through a single-pass online learning process, where we have all positive examples at our disposal (and they are in fact all included in the training sequence), but where negative examples cannot be all included, because they are too many and they would produce overfitting.

Both the Passive-Aggressive construction described above and the Perceptron algorithm depend crucially on the sequence of positive and negative examples $(i_1, j_1), \ldots, (i_T, j_T)$ that is taken as input. In particular, as discussed in [32], it is critical that the number of negative

and positive examples in the sequence is balanced. Taking this suggestion into account – and also considering [63] suggestions about uniform sampling – we build the sequence as follows: we draw uniformly at random $|A|$ node pairs (i, j) s.t. $(i, j) \notin A$; then, nodes are enumerated (in arbitrary order), and for each node $i \in N$, all arcs of the form $(i, \bullet) \in A$ are added to the sequence, followed by all non-links node pairs of the form $(i, \bullet)$. Of course, in the end the sequence contain $T = 2 \cdot |A|$ node pairs – that is, $|A|$ links along with $|A|$ non-links.

Obviously, there are other possible ways to define the sequence of examples and to select the subset of negative examples. However, we chose to adopt this technique (single pass on a balanced random sub-sample of pairs) in order to define and test our methodology with a single, natural and computationally efficient approach. However, when experimenting with real data in Section 6, we will also test whether the ordering of nodes affects the results, by comparing natural (i.e. chronological) and random order.

Error bound for Passive-Aggressive. The analysis of the error bound for misclassifications of the perceptron (9) can be made more precise for the case of the Passive-Aggressive algorithm: using Theorem 4 of [13], the bound becomes:

$$M \leq \inf_{\mathbf{U} \in \mathbb{R}^{m \times m}} \max\left(R^2, 1/\kappa\right) \left(2\kappa H(\mathbf{U}) + \|\mathbf{U}\|^2\right). \tag{13}$$

If $\kappa = 1/R^2$, the bound reduces to

$$M \leq \inf_{\mathbf{U} \in \mathbb{R}^{m \times m}} 2H(\mathbf{U}) + \left(R\|\mathbf{U}\|\right)^2,$$

and our discussion of (9) is essentially confirmed. We encounter $R^2 = \max_{(i,j) \in T} |F_i| \cdot |F_j|$, that is the maximum number of pairs of features we observe at the same time; $H(\mathbf{U})$, the total loss of the "best" (in terms of the infimum in the equation) possible feature-feature matrix; and $\|\mathbf{U}\|$, the norm of such a matrix, which is fundamentally a measure of its complexity. Also for Passive-Aggressive, these factors define the performance of the algorithm on a specific instance of feature-rich graph.

A truly on-line approach with unnormalized samples will require a constant κ (in our experiments we set 1.5), which yields

$$M \leq \inf_{\mathbf{U} \in \mathbb{R}^{m \times m}} cR^2 H(\mathbf{U}) + \left(R\|\mathbf{U}\|\right)^2,$$

for some constant c.

5 Experiments on synthetic data

In this section, we will test how the methods described in this paper perform on synthetic graphs generated within our framework using the techniques described in previous work [5]; in the next section we will see how they behave on real-world data.

We are in fact building upon previous methods [5] to generate a realistic node-feature association $\mathbf{Z}$ that, when used as input to the model of (1), is able to synthesize feature-rich networks with the same traits (e.g., distance distribution, degree distribution, fraction of reachable pairs, etc) as typical real complex networks. In particular, in [5] we discuss how to generate a synthetic feature-rich graph with the same properties as a given real one. These experiments allow us to employ graphs generated through this approach as a test bed for the algorithms presented in the Section 4.

Avg. features per node

	S	χ	exp
$\mathcal{B}$	5.84 ± 1.63	5.17 ± 1.63	5.22 ± 1.51
$\mathcal{N}$	5.76 ± 1.43	5.30 ± 1.56	5.51 ± 1.31

Avg. degree

	S	χ	exp
$\mathcal{B}$	109.4 ± 325	163.2 ± 329	15.6 ± 217
$\mathcal{N}$	10.9 ± 145	11.8 ± 138	26.3 ± 299

Mean harmonic distance

	S	χ	exp
$\mathcal{B}$	2.16 ± 92	$2.43 \pm 1\,339$	$2.02 \pm 3\,290$
$\mathcal{N}$	$2.31 \pm 3\,034$	$11.0 \pm 2\,472$	$2.01 \pm 1\,606$

Table 1: Properties of the synthetic feature-rich graphs. The 6 generated graph families are indicated according to the ϕ function used (S is the sigmoid, χ is the step function, and exp is the exponential) and to the distribution of the values of $\mathbf{W}$ (Bernoullian or normal). The listed properties represents the median, inside each graph family, of: the average number of features per node, the average degree and and the mean harmonic distance.

5.1 Experimental setup

To generate each network, we first produced its node-feature association $\mathbf{Z}$ with the Indian Buffet Model method [5], using the same parameter values adopted in previous work: $\alpha = 3$, $\beta = 0.5$, $c = 0$. Then, we fed these matrices $\mathbf{Z}$ to our model equation (1) to generate a number of graphs. For the graph model, we employed the following parameters:

- We used $n = 10\,000$ nodes.
- We applied three different types of activation function ϕ, to compare their results:
  1. The classic sigmoid function $S(x) = \left(e^{K(\vartheta - x)} + 1\right)^{-1}$, cited in Section 3.1 as well as in [5] as the standard approach; we set $\vartheta = 0$ and $K = 5$. Please note that this function does *not* respect the assumptions for which we derived Llama, nor those of Naive.
  2. The step function $\chi_{(0,\infty)}$, characterizing the model behind Llama.
  3. The exp function, which characterizes the model behind Naive.
- The latent matrix $\mathbf{W}$ was generated assuming that its entries are i.i.d., with the following two value distributions:
  1. A generalized Bernoulli distribution $W_{h,k} \sim \mathcal{B}(p)$ that assumes the value 10 with probability $p = \frac{10}{m}$ and -1 with probability $1 - p$. This choice was determined through experiments, with the purpose of obtaining graphs with a realistic density independently from the number of features m.
  2. A normal distribution $W_{h,k} \sim \mathcal{N}(\mu, \sigma)$ with mean and variance identical to the previous Bernoulli distribution.

3. We had to slightly modify these distributions for the case $\phi = \exp$, in order to obtain realistic graphs also in that case: in particular, when $\phi = \exp$ we used a Bernoulli distribution with value 1 with probability $p = \frac{1}{m}$ and -1 with probability $1 - p$, and a normal distribution that had the same mean and variance as the just-described Bernoulli distribution. In the following, when we say that $\phi = \exp$ we imply that we used one of these two modified distributions to generate $\mathbf{W}$.

With these three choices for ϕ and two choices for the generation of $\mathbf{W}$, we obtained six different families of feature-rich graphs. For each graph family, we generated 100 different graphs. The properties of these networks are summed up in Table 1. They represent a wide range of realistic traits we could actually observe in complex networks.

5.2 Evaluation

First of all, even if the aim of both LLAMA and NAIVE is to reconstruct the matrix $\mathbf{W}$, we are not interested in the actual values of the elements of $\mathbf{W}$. Our goal is to find a feature-feature matrix for which our model works: it is not important if the values are scaled up or shifted as long as the predictions of our model for the links remain correct.

For this reason, we will measure directly how accurate our methods are in terms of predicting if a node pair (i, j) forms a link, given their features. To keep this evaluation meaningful, our algorithms will not be allowed to see the whole graph: we will use the standard approach of 10-fold cross-validation; i.e., we divide the set of nodes N into ten subsets (folds) of equal size, and we used nine folds to train the algorithm and the tenth remaining fold to test the results (for each possible choice of the latter).

Our evaluation closely resembles the approach followed for *link prediction*. There are of course some differences: first of all, we are using an external source of information (the node features) that is not available to link-prediction methods; second, our aim is to evaluate our model and our algorithms to find $\mathbf{W}$ *through* link prediction. That is, we are not interested in finding the best existing link predictor, but in measuring if our algorithms can correctly fit our model on a specific instance of feature-rich graph $(G, \mathbf{Z})$. However, we followed the evaluation guidelines for link prediction recently stated by Yang *et al.* [63].

- We evaluated how accurate our algorithms are in prediction by showing precision/recall curves: Yang *et al.* [63], in fact, observe that other alternatives, such as the ROC curve, are heavily biased after undersampling negative examples and can yield misleading results; since tied scores do affect results (especially for NAIVE), we employed the techniques described in [43] to compute precision and recall values for tied scores.
- Using precision/recall curves allow us to avoid using a fixed threshold between "link"/"not link"; it is important, in fact, to evaluate the *scores* themselves; on the contrary, by choosing a threshold ϑ and then converting each score x to a binary event $x > \vartheta$ would make the comparison unfair; we instead used directly the score computed by our model (the argument of ϕ in (1)) since the larger this score, the more probable that link should be.
- We used the same test set for all the tested algorithms.
- Although in our case it was necessary to undersample negatives (the total number of node pairs would be unmanageable), we took care of sampling uniformly the edges missing from the test network: we draw node pairs (i, j) such that $(i, j) \notin A$ uniformly from the set $N \times N$, until we had a number of non-arcs equal to the number of arcs.

Since our methods are not influenced by the distances of the pairs of nodes involved (contrarily to standard link prediction approaches), we avoided to gather our results by geodesic distance.

	AUPR	Time (s)
NAIVE	0.824 ± 0.028	0.034 ± 0.034
LLAMA	0.893 ± 0.020	0.097 ± 0.097
SVM	0.915 ± 0.014	6439.303 ± 6439.303

Table 2: Area under the precision-recall curve (on average across 10 folds and 4 experiments) and the required training time in seconds. For each value we report the mean and the standard deviation.

With the above considerations in mind, we proceeded to evaluate our approach using precision-recall curves. For each of the networks and for each fold, we gave the training graph as input to the algorithm (LLAMA or NAIVE) and obtained an estimated matrix $\mathbf{W}$. This matrix is defined by[6] (5) for NAIVE and by[7] Algorithm 1 for LLAMA. Each method then assigned its score (i.e., the argument of ϕ in (1)) to each node pair in the test set, according to our model.

5.3 Training time

Before discussing the results, let us present a measure of the training times of the algorithms we propose, in comparison with SVM, a baseline previously employed in the literature for feature-rich graphs [17]. For this algorithm, we are using an efficient implementation (the one from WEKA [24]), written in the same language as our own algorithms, and using therefore the same methods for I/O. We employed a linear kernel (the fastest) for the SVM.

The results we show are about a single graph family of the ones discussed above (specifically, the case where $W_{h,k} \sim \mathcal{N}(\mu, \sigma)$ and the sigmoid function $S(x)$ is used as an activation function). These are the most common cases treated in the literature. Also, we needed to set a lower number of nodes $n = 1000$ in order for the SVM to terminate.

Our results (Table 2) show a training time for the SVM that is four orders of magnitude longer than NAIVE or LLAMA, i.e., taking on the scale of hours for graphs of thousands of nodes. These results are consistent with the previous literature. Perceptron-like algorithms are known to be much less computationally expensive than traditional SVMs [56]. However, despite them to be unusable at the scale we want to operate (tens millions of nodes), it is worth noting that their performance is (slightly) better than LLAMA in this particular case.

5.4 Results

We report detailed performance results for NAIVE and LLAMA in Table 3. There, we show the average AUPR (Area Under Precision-Recall curve) obtained across all the graphs inside each graph family considered. To compute the AUPR we used the technique described by Davis and Goadrich [16].

Following the previous suggestions [63], we use this area as an overall measure of the goodness of our approach. We can see how the results of LLAMA are on average above 95% for both the step function and the sigmoid activation function. The exp case, in fact, is the

[6] In the case $(N_h \times N_k) \cap A = \emptyset$, the Naive approach as described by (5) would set $W_{h,k} = \log 0$. We tried two alternative strategies to solve this issue: (i) setting $W_{h,k}$ equal to a large negative number for those pairs (*de facto* putting a lower bound to $W_{h,k}$); (ii) employing an add-one smoothing [55], i.e., using $\log(x+1)$ in place of $\log(x)$. The experimental results are essentially the same in the two cases. The figures presented in this section are the ones obtained by (i).

[7] We tried also the normalized version of LLAMA expressed in (12), for different values of p, leaving the model unchanged. Again, the experimental results obtained are the same on our dataset, so we are here presenting the values obtained by the unnormalized version of the algorithm.

	$S, \mathcal{B}$	$S, \mathcal{N}$	$\chi, \mathcal{B}$	$\chi, \mathcal{N}$	$\exp, \mathcal{B}$	$\exp, \mathcal{N}$
NAIVE	$.843 \pm .060$	$.951 \pm .148$	$.599 \pm .288$	$.798 \pm .258$	$.931 \pm .232$	$.972 \pm .084$
LLAMA	$.974 \pm .016$	$.951 \pm .151$	$.973 \pm .018$	$.967 \pm .117$	$.529 \pm .279$	$.880 \pm .155$

Table 3: Area under the precision-recall curve of NAIVE and of LLAMA. For each of the considered graph families, we report the mean and the standard deviation across all the graphs.

one where NAIVE works better – as it was expected from the theory. In the normal case, the performance of LLAMA is still good; a Bernoulli distribution with an exponential activation function is instead the only case when LLAMA performance is inadmissible. As we shall see in Section 6, though, the exp case does not correspond to a realistic setting.

Let us discuss in details the results obtained, gathering them by the activation function employed to generate the graph. To be able to grasp what happens across different folds in a single graph, and to avoid overcrowding the plots, we will report the precision-recall plots for a single graph inside each family.

Step activation (Figure 3 and 4). Let us first consider the case of the networks generated with a step activation function. Note that by using $\chi_{(0,\infty)}$ as the activation function we are making our model deterministic — a pair forms a link if and only if its score is positive. Furthermore, this is precisely the activation function for which we have formal guarantees on the LLAMA performances. In fact, its results are remarkably good, as testified by an area under curve beyond 96% in both the Bernoullian and the Gaussian case.

NAIVE is able to take advantage of this clean activation function only with a normal distribution on the values of $\mathbf{W}$ (where its performance is around 80%); in the bernoullian case, it degrades toward a random classifier.

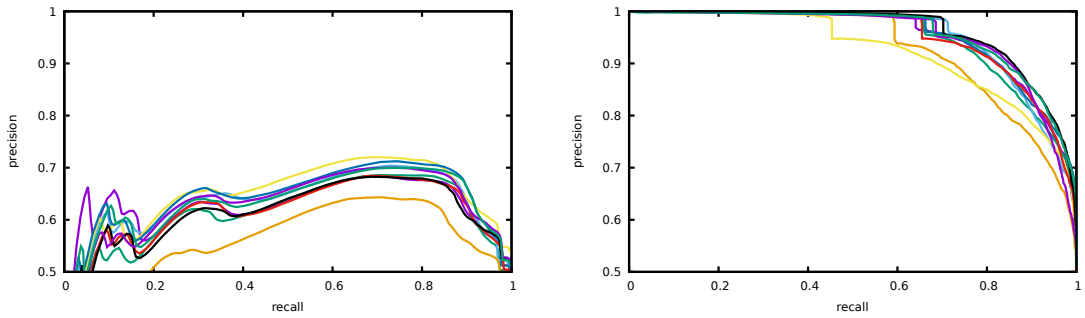
Exponential activation (Figure 5 and 6). Let us now look at the exponential activation function, for which we have formally derived NAIVE. The results obtained by NAIVE are in fact very good at all recall levels.

LLAMA, on the other hand, obtains its worst performance on this simulation, due to the fact that the exponential function is mostly dissimilar from LLAMA's natural one (the step function). In the bernoullian case its performance is chaotic, and depends very much on the training set; instead, in the normally-distributed case, the area under the precision-recall curve is definitely better, around 80% on average.

Sigmoid activation (Figure 7 and 8). Finally, let us look at the results obtained when the activation function is a sigmoid (2) with $K = 5$. We emphasize that this activation function is one for which we have no theoretical guarantees, neither for LLAMA (which assumes a step function) nor for NAIVE (which assumes an exponential); also, it is the function of choice in previous literature (e.g. [47]).

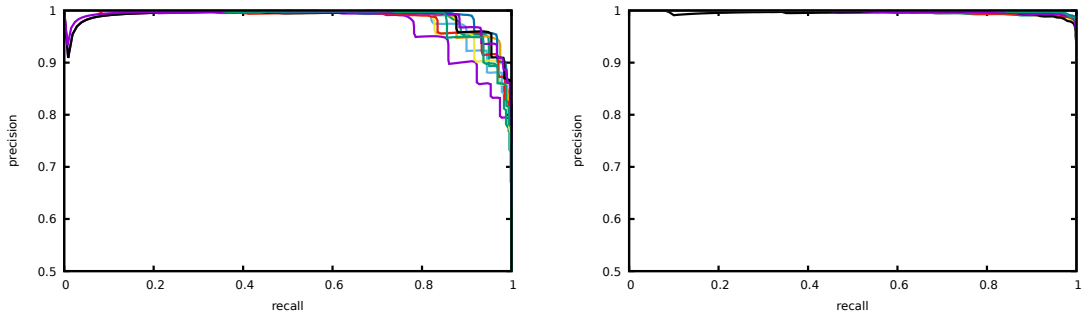
We report in Figure 7 the precision-recall curves for the case of the Bernoulli distribution and in Figure 8 the precision-recall curves for the case of the normal distribution. We can see how NAIVE performances display a high variance and are way behind the LLAMA performances, especially in the Bernoulli-distributed case. LLAMA performances in fact are almost as good as in its natural step function case, with an area under curve consistently beyond 95%.

The unambiguous prevalence of LLAMA in this "natural" case could explain the results we are showing in the next section.



NAIVE LLAMA

Figure 3: Precision-recall curves in the network $\chi, \mathcal{B}$. Different colors represent different folds used in cross-validation.



NAIVE LLAMA

Figure 4: Precision-recall curves in the network $\chi, \mathcal{N}$. Different colors represent different folds used in cross-validation.

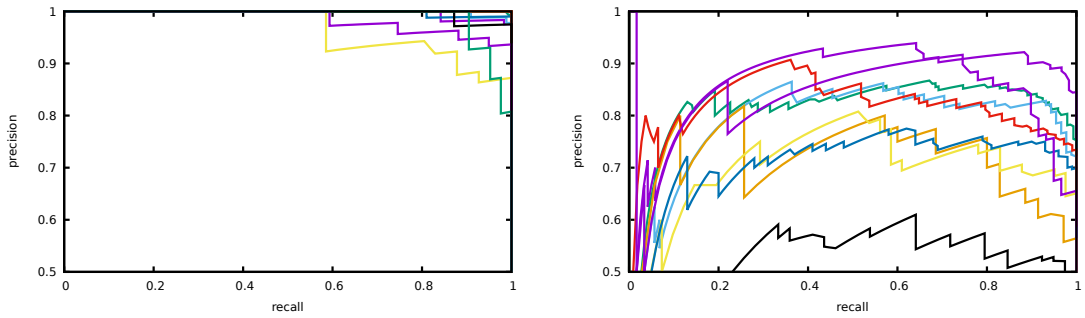
6 Experiments on real data

In this section, we will focus on (1) how our algorithms behave on real-world feature-rich networks and (2) how our framework can be used to evaluate the relationship between a network and a particular set of features for its nodes. In particular, we will consider the fitness of our model as a measure of how much a certain set of features can explain the links in such a graph.

Explainability. Given a graph $G = (N, A)$ and a particular set of features $\hat{F}$ that can be associated to its nodes (with $\hat{Z} \subseteq N \times \hat{F}$), we can define the *explainability* of $\hat{F}$ for G to be the area under the precision-recall curve obtained by the scores provided by our model; with "score" we mean, as before, the argument of ϕ in (1), where the matrix $\mathbf{W}$ is the one found by Algorithm 1 when it is given G and $\hat{Z}$ as input. We again use the AUPR (Area Under Precision-Recall curve) as a measure of fitness, as we did in Section 5.3.

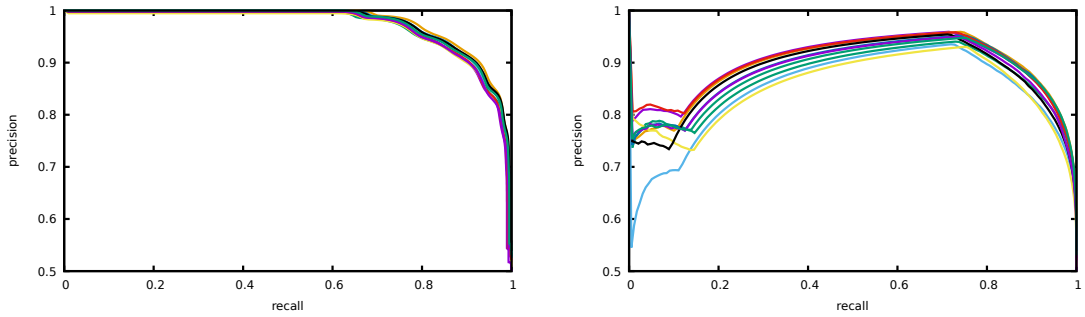
6.1 Experimental setup

We are going to consider a scientific network recently released by Microsoft Research, and known as the Microsoft Academic Graph [57]. It represents a very large (tens of millions), heterogeneous corpus of scientific works; each scientific work has some metadata associated



NAIVE LLAMA

Figure 5: Precision-recall curves in the network exp, $\mathcal{B}$. Different colors represent different folds used in cross-validation.



NAIVE LLAMA

Figure 6: Precision-recall curves in the network exp, $\mathcal{N}$. Different colors represent different folds used in cross-validation.

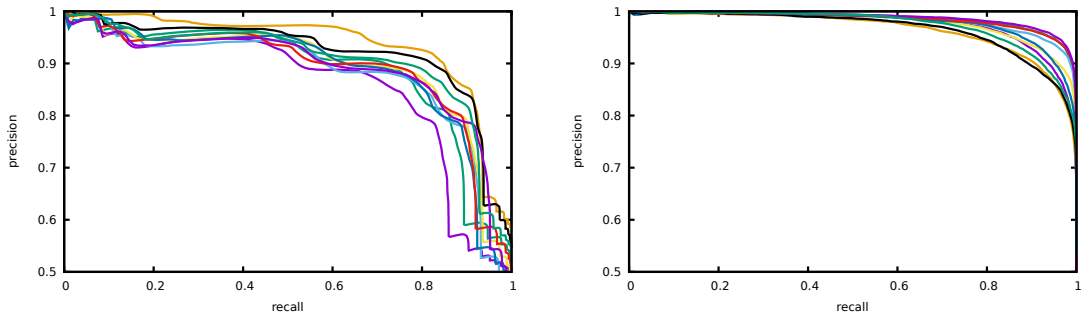
with it.

We will consider the citation network formed by these papers: this is a directed graph whose nodes are the papers, and with an arc $(i, j) \in A$ if and only if paper i contains a citation to paper j. As for the features, we will consider the following alternative sets of node features:

- authors' *affiliations*: for each paper, all the institutions that each author of the paper claims to be associated to. "University of Milan" and "Google" are examples of affiliations.
- the set of *fields of study*: the field of study associated by the dataset curators [57] to the keywords of the paper. "Complex network" and "Vertebrate paleontology" are examples of fields of study.

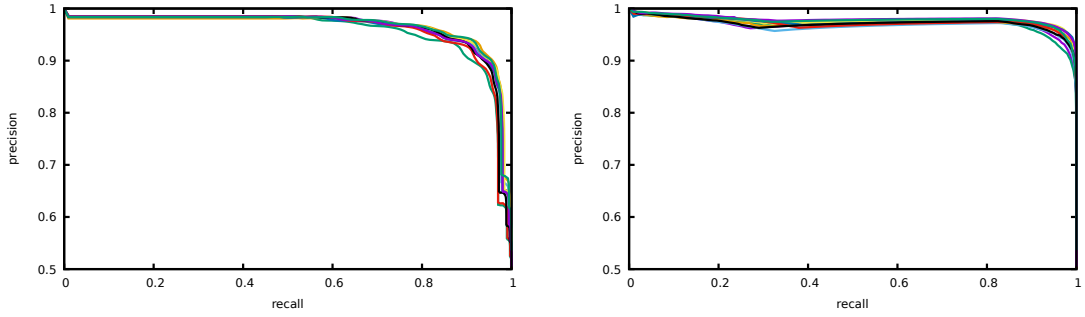
These features fully respect all the assumptions we made: they are attributes of the nodes, they are binary (a node can have a feature or not, without any middle ground), they are possibly overlapping (a paper can have more than one affiliation/field associated with it).

Our goal now is to compare the explainability (as defined above) of these two sets of features for the citation network. Since we want to compare them fairly, we reduced the dataset to those nodes for which the dataset specifies both features: that is, papers for which both the affiliations and the fields of study are reported. In this way we obtained:



NAIVE LLAMA

Figure 7: Precision-recall curves in the network $S, \mathcal{B}$. Different colors represent different folds used in cross-validation.



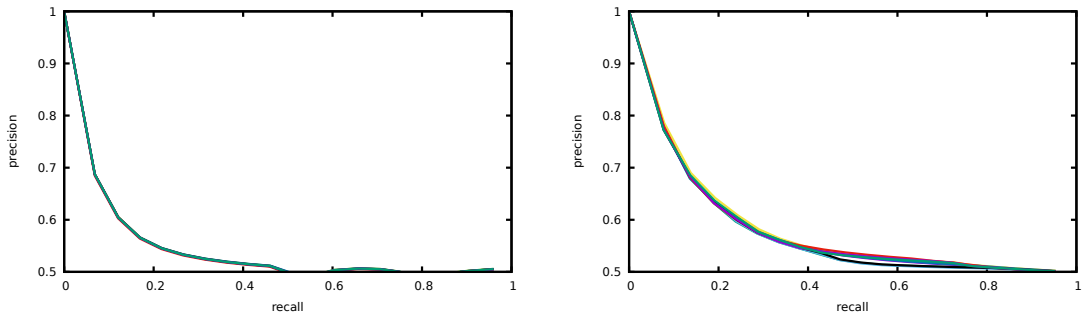
NAIVE LLAMA

Figure 8: Precision-recall curves in the network $S, \mathcal{N}$. Different colors represent different folds used in cross-validation.

- A graph $G = (N, A)$ where N is a set of 18 939 155 papers, and A contains the 189 465 540 citations between those papers.
- A set F_a of 19 834 affiliations, and the association $\mathbf{Z}_a$ between papers and affiliations. Each paper has between 1 and 182 affiliations; on average, we have 1.36 affiliations per paper.
- A set F_f of 47 269 fields, and the association $\mathbf{Z}_f$ between papers and those fields of study. Each paper involves between 1 and 200 fields; on average, we have 3.88 fields per paper.
- As a further type of test, we performed the experiments also on the union $F_a \cup F_f$.

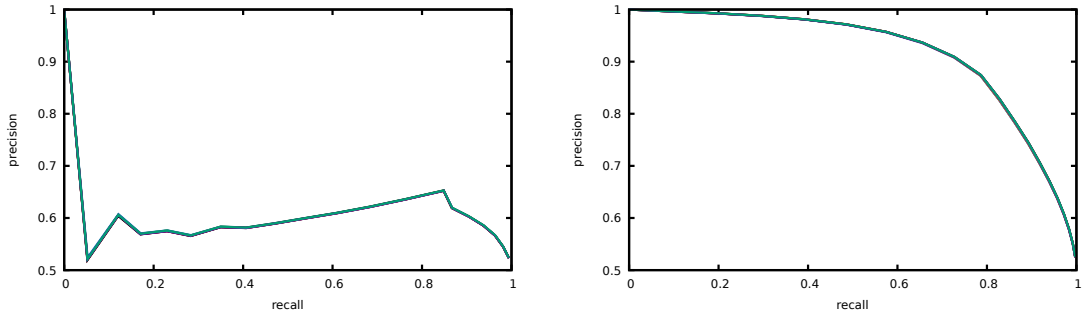
We proceeded then to evaluate the explainability of F_a and F_f for G with the same approach presented in Section 5.3:

1. We divide the set N in ten folds $N_0, \ldots, N_9$.
2. For each fold N_i:
   (a) We apply Algorithm 1 to the part of A and $\mathbf{Z}$ related to the training set $\cup_{j \neq i} N_j$.
   (b) We obtain a matrix $\mathbf{W}$.



NAIVE LLAMA

Figure 9: Precision-recall curves of the Naive baseline and of LLAMA, when explaining the citation network using the affiliation of authors as features. Different colors represent different folds used in cross-validation.



NAIVE LLAMA

Figure 10: Precision-recall curves of the Naive baseline and of LLAMA, when explaining the citation network using the fields of study of each paper as features. Different colors represent different folds used in cross-validation.

(c) We compute the scores of our model with $\mathbf{W}$ on the test set N_i.

(d) We measure the precision-recall curve for these scores.

In order to validate on real data the results we obtained in Section 5.3 for synthetic data, we also carried out the same procedure also with the $\mathbf{W}$ matrix found by NAIVE. As a result, we obtained two ten-folded precision-recall curves for each of the three set of features considered: F_a, F_f and $F_a \cup F_f$.

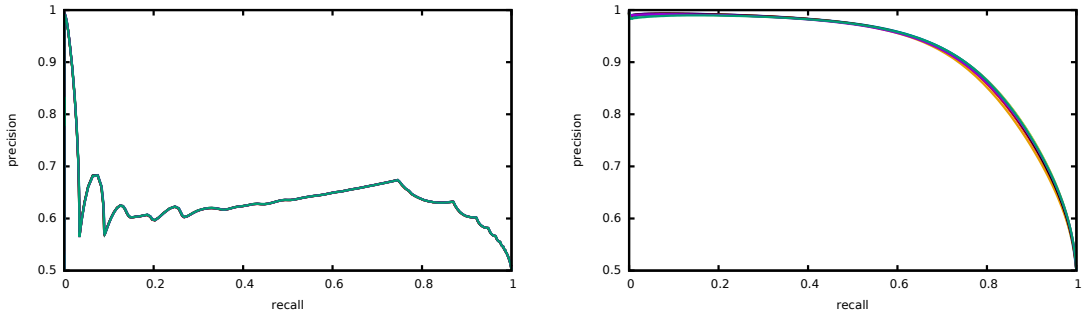
Furthermore, we are comparing two different orderings for node sequences in LLAMA: one is purely random (the one we suggested in Section 4.2.2), while the other is the natural order of nodes in this case, i.e., the chronological order of paper publication. Please note, however, that the 10-fold cross-validation is still operated a random (each train-test split is performed randomly, regardless of ordering).

6.2 Results

In Table 4 we report the explainability we obtained (measured as the area under the precision-recall curves shown). We report in Figure 9, 10 and 11 the precision-recall curves for NAIVE and for LLAMA concerning the feature set F_a, F_f and $F_a \cup F_f$, respectively.

	Affiliations	Fields of study	Both
Llama	$\mathbf{.5551} \pm .0028$	$\mathbf{.9162} \pm .0003$	$.9210 \pm .0012$
Llama (natural order)	$.5446 \pm .0013$	$.9063 \pm .0004$	$.9176 \pm .0002$
Naive	$.5237 \pm .0005$	$.6007 \pm .0004$	$.6345 \pm .0002$

Table 4: Area under the precision-recall curve of the Naive baseline and of Llama. For each of the feature sets considered, we report the mean and the standard deviation across the ten folds. We highlighted the *explainability* for the citation network of the affiliations and of the fields of study, respectively.



Naive Llama

Figure 11: Precision-recall curves of the Naive baseline and of Llama, when explaining the citation network using both the affiliations and the fields, together, as features. Different colors represent different folds used in cross-validation.

From the table, we can see that the explainability of the fields of study for the citation network is much larger than that of the authors affiliations: the first is above 92%, while the second is 56%. In this sense, our model allows us to say that the fields of study of a paper explain very well its citations, while the affiliations of its authors do not. This might not come as a surprise (the relationship between the fields a paper belongs to and its citations is quite natural) but our contribution here is the formal framework which allows us to back this assertion with solid numbers, through (1) and Algorithm 1.

We can further validate this statement by looking at the explainability for $F_a \cup F_f$: its value of 92.1% is just faintly over the value of 91.6% obtained for fields alone, implying that the gain obtained by including the whole new set F_a of 19 834 features is practically negligible.

Finally, it is worth noting that the ordering of the node does not affect much the results, that go from 91% of the usual random order to 90% for the natural order.

We can grasp more details by looking at the specific precision-recall curves. By comparing the Llama curve for affiliations in Figure 9 and the one for fields in Figure 10, we can see immediately that the latter depicts a valid classification instrument; there, the precision/recall break-even point is around 83%. Also, we can see some specific characteristic of the affiliation feature set: it is in fact able to reach a large precision, but only in the very low range of recall. Here, a precision of 83% is possible only with a recall lower than 7%: the reason behind this is that an author's affiliation is effective in encouraging a citation in a very limited set of circumstances; we can conjecture that homophily within small institutions could be an example.

Finally, let us remark how the results we obtained on synthetic data in Section 5.3 are

fully confirmed by the real data we presented here: LLAMA, in all the three cases, behaves much better than NAIVE. This is especially true for the feature set that actually explains the network: for the fields of study, LLAMA is able to get a 91% value for AUPR, while the **W** matrix found by Naive approach can barely get a 63%. In particular, precision-recall curves look similar to the one shown in Figure 7, corresponding to the simulation obtained with ϕ set to a sigmoid and **W** having a Bernoulli distribution; real data is actually less shaky, due to the fact that we have 18 millions of nodes instead of the 10 000 used in the simulation. Besides confirming the validity of LLAMA, this observation also confirms the goodness of our model in explaining a real graph.

7 Conclusions and future work

In this work, we investigated large, feature-rich complex networks (networks where each node is characterized by set of features). Specifically, we wanted to analyze a model where node features induce the formation of the links we observe. This hypothesis is reasonable in many scenarios (the citation networks used in our experiments are just one example). As discussed in Section 2, we employed the Miller-Griffiths-Jordan model as our starting point. The problem we dealt with was how to infer the latent feature-feature matrix: this matrix is the main *unknown* of the model; it determines how features interact between each other to give raise to the observed links.

Specifically, we focused on the following scenario: assume to have complete knowledge of a node-feature association matrix – i.e., to know for every node, the features it exhibits (embodied in the binary matrix **Z**); also, assume to have an (at least partial) knowledge of the links between these nodes (the graph G). Our goal was, given these elements, to find the latent interaction between features that governs link formation in the graph G; i.e., to discover the latent matrix **W** of our model (1). This estimate alone allows us to use our model as a possible way to predict which pair of nodes form a link. Other possible applications include dimensionality reduction of the features, measuring semantic distance, discovering hidden relationships, and so on.

While many possible methods are available in literature to attack these problems, they generally only can handle small/medium sized networks, while we are interested in large-scale networks. This ruled out many well-known techniques, like MCMC. Our first approach was guided by a Naive Bayes scheme: we demonstrated that a very simple equation to estimate the matrix can be derived by assuming (naively) independence between features, and by making a few assumptions to restrict our model. However, we pointed out how its naive assumptions can cause problems in practical applications, and for this reason we described a more sophisticated approach, based on perceptrons.

To link it formally with our model of choice, we assumed it to be deterministic by choosing a step activation function ϕ in (1). This assumption allowed us to align our model equation to a perceptron decision rule, by applying an outer product kernel to the binary vectors $\mathbf{z}_i$ and $\mathbf{z}_j$ representing the features in nodes i and j, and to make the perceptron predictions represent whether they form a link or not. In this way, the internal state of the perceptron converges to the latent feature-feature matrix **W**. We described this learning-based approach, and analyzed what a classical bound on the number of errors of a perceptron means in this case. Then, since any perceptron-like algorithm can be adapted for this purpose, we chose the simple and fast Passive-Aggressive algorithm [13] to concretely implement this approach (Algorithm 1).

In the experimental section, we tested how this algorithm behaves on synthetic data. We generated graphs and node-feature associations according to the model presented in [5], under different assumptions. In measuring the outcomes, we adopted the same techniques as suggested in [63]: specifically we measured the link prediction capability of the estimated **W** through a ten-fold cross-validation. Results showed how our learning approach outperforms

the Naive baseline in all the analyzed cases, except for the exp activation function.

Finally, we conducted an experiment on a real dataset, a citation network composed by 18 939 155 nodes and 189 465 540 link; running the algorithm required about 20 minutes. In fact, we used the tools we developed for estimating the feature-feature matrix in order to validate their performance on real data, and to show how they can be used to assess which feature set can be more useful in explaining the links of a network.

In this work, our main contribution consisted in laying out a bridge between perceptron-like learning algorithms and feature-rich graph models; we formally presented the connection between them, and we showed how they can be valuable from a practical point of view when analyzing graphs that have tens of millions of nodes or more.

We hope that the intersection of machine learning and complex network models will attract more research in the future; many questions are left open on these topics. Given a specific graph (possibly with features) how can we understand what is the best model that can explain its links? Can this model also offer a learning algorithm that allows us to make predictions about unknown nodes? A full answer to this question would look, from one side, like a network "family tree": it would enumerate possible models of networks by describing the formation of their links, each being more or less reasonable depending on the specific network at hand. From the other side, such a "family tree" would look like a toolbox in the hands of the network scientist: each model should offer algorithms for link prediction that could be more or less accurate or computationally efficient.

Regarding the efficiency of algorithms for our models, there are some alternatives that are left unexplored: for example other online algorithms, like PEGASOS; also, we would like to investigate better formal connections between neural networks and complex networks; for example, can deeper neural networks also be read as a sensible feature-rich graph model?

Other future directions stem, on the contrary, from modifying our model. The latent matrix $\mathbf{W}$, as reconstructed by the algorithms described in this paper, will be dense; what happens if we reduce its density (e.g., by thresholding the absolute value of its entries)? How much would that impact on our ability to reconstruct $\mathbf{A}$? This density/precision tradeoff can be taken into consideration from start: we may want to try to construct a latent matrix that satisfies some constraints (e.g., on its density, or on its norm). This constrained version of the problem may shed new light on the relation between features and links, and can be a fruitful research direction.

Finally, we remark how it would be definitely important to test the proposed techniques on other real feature-rich complex networks, in order to see in which concrete cases they can improve over the current techniques for link prediction and, more generally, for understanding hidden patterns in network data.

We consider these questions of primary importance, in order to be able to avoid viewing graph mining algorithms as black boxes, but considering instead what they could say about the structure and the evolution of specific complex networks.

References

[1] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing, *Mixed membership stochastic blockmodels*, J. Mach. Learn. Res. **9** (June 2008), 1981–2014.

[2] S. O Aral, J.P. Hughes, B. Stoner, W. Whittington, H.H. Handsfield, R.M. Anderson, and K.K. Holmes, *Sexual mixing patterns in the spread of gonococcal and chlamydial infections.*, American Journal of Public Health **89** (1999), no. 6, 825–833.

[3] Halil Bisgin, Nitin Agarwal, and Xiaowei Xu, *A study of homophily on social media*, World Wide Web **15** (2012), no. 2, 213–232.

[4] C.M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, Springer-Verlag New York, Inc., 2006.

[5] Paolo Boldi, Irene Crimaldi, and Corrado Monti, *A network model characterized by a latent attribute structure with competition*, Information Sciences (2016), –.

[6] Ronald L. Breiger, *The Duality of Persons and Groups*, Social Forces **53** (1974), no. 2, 181–190.

[7] Guido Caldarelli, Andrea Capocci, Paolo De Los Rios, and Miguel A Munoz, *Scale-free networks from varying vertex intrinsic fitness*, Physical review letters **89** (2002), no. 25, 258702.

[8] V.R. Carvalho and W.W. Cohen, *Single-pass online learning: Performance, voting schemes and online feature selection*, Proc. of the 12th acm sigkdd, 2006, pp. 548–553.

[9] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile, *On the generalization ability of on-line learning algorithms*, IEEE Transactions on Information Theory **50** (2004), no. 9, 2050–2057.

[10] Nicolo Cesa-Bianchi and Gábor Lugosi, *Prediction, learning, and games*, Cambridge university press, 2006.

[11] J. Chang and D.M. Blei, *Relational topic models for document networks*, International conference on artificial intelligence and statistics, 2009, pp. 81–88.

[12] Michel Chein and Marie-Laure Mugnier, *Graph-based knowledge representation: computational foundations of conceptual graphs*, Springer Science & Business Media, 2008.

[13] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, *Online passive-aggressive algorithms*, J. Mach. Learn. Res. **7** (2006), 551–585.

[14] Nello Cristianini and John Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*, Cambridge university press, 2000.

[15] Nilesh Dalvi and Dan Suciu, *Efficient query evaluation on probabilistic databases*, The VLDB Journal **16** (2007), no. 4, 523–544.

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

[17] Janardhan Rao Doppa, Jun Yu, Prasad Tadepalli, and Lise Getoor, *Learning algorithms for link prediction based on chance constraints*, Joint european conference on machine learning and knowledge discovery in databases, 2010, pp. 344–360.

[18] B Everett, *An introduction to latent variable models*, Springer Science & Business Media, 2013.

[19] C. Gentile, *A new approximate maximal margin classification algorithm*, J. Mach. Learn. Res. **2** (2002), 213–242.

[20] Charles J. Geyer and Minnesota Univ. (Minneapolis School Of Statistics), *Markov Chain Monte Carlo Maximum Likelihood.*, Defense Technical Information Center, 1992.

[21] Neil Zhenqiang Gong, Wenchang Xu, Ling Huang, Prateek Mittal, Emil Stefanov, Vyas Sekar, and Dawn Song, *Evolution of social-attribute networks: measurements, modeling, and implications using google+*, Proceedings of the 2012 acm conference on internet measurement conference, 2012, pp. 131–144.

[22] Leo A Goodman, *Exploratory latent structure analysis using both identifiable and unidentifiable models*, Biometrika **61** (1974), no. 2, 215–231.

[23] Thomas L. Griffiths and Zoubin Ghahramani, *Infinite latent feature models and the indian buffet process*, Advances in neural information processing systems, 2005, pp. 475–482.

[24] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten, *The weka data mining software: an update*, ACM SIGKDD explorations newsletter **11** (2009), no. 1, 10–18.

[25] K. Henderson and T. Eliassi-Rad, *Applying latent dirichlet allocation to group discovery in large graphs*, Proc. 2009 acm symposium on applied computing, 2009, pp. 1456–1461.

[26] Neil W Henry, *Latent structure analysis*, Encyclopedia of statistical sciences (1983).

[27] N. Hens, N. Goeyvaerts, M. Aerts, Z. Shkedy, P. Van Damme, and P. Beutels, *Mining social mixing patterns for infectious disease models based on a two-day population survey in belgium*, BMC Infectious Diseases **9** (2009), no. 1, 5.

[28] John Hertz, Anders Krogh, and Richard G Palmer, *Introduction to the theory of neural computation*, Vol. 1, Basic Books, 1991.

[29] P.D. Hoff, *Multiplicative latent factor models for description and prediction of social networks.*, Computational and Mathematical Organization Theory **15** (2009), no. 4, 261–272.

[30] Jake M Hofman and Chris H Wiggins, *Bayesian approach to network modularity*, Physical review letters **100** (2008), no. 25, 258701.

[31] Lorenzo Isella, Mariateresa Romano, Alain Barrat, Ciro Cattuto, Vittoria Colizza, Wouter Van den Broeck, et al., *Close encounters in a pediatric ward: measuring face-to-face proximity and mixing patterns with wearable sensors*, PloS one **6** (2011), no. 2, e17144.

[32] N. Japkowicz and S. Stephen, *The class imbalance problem: A systematic study*, Intelligent data analysis **6** (2002), no. 5, 429–449.

[33] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda, *Learning systems of concepts with an infinite relational model*, Aaai, 2006, pp. 5.

[34] Arijit Khan and Lei Chen, *On uncertain graphs modeling and queries*, Proceedings of the VLDB Endowment **8** (2015), no. 12, 2042–2043.

[35] Myunghwan Kim and Jure Leskovec, *Modeling social networks with node attributes using the multiplicative attribute graph model*, arXiv preprint arXiv:1106.5053 (2011).

[36] ———, *Multiplicative attribute graph model of real-world networks*, Internet Mathematics **8** (2012), no. 1-2, 113–160.

[37] ———, *Nonparametric multi-group membership model for dynamic networks*, Advances in neural information processing systems, 2013, pp. 1385–1393.

[38] Silvio Lattanzi and D. Sivakumar, *Affiliation networks*, Proc. of the forty-first annual acm symposium on theory of computing, 2009, pp. 427–434.

[39] Paul F Lazarsfeld, *Latent structure analysis*, Psychology: A study of a science **3** (1959), 476–543.

[40] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani, *Kronecker graphs: An approach to modeling networks*, Journal of Machine Learning Research **11** (2010), no. Feb, 985–1042.

[41] Y. Liu, A. Niculescu-Mizil, and W. Gryc, *Topic-link lda: joint models of topic and author community*, Proc. 26th annual international conference on machine learning, 2009, pp. 665–672.

[42] Miller McPherson, Lynn Smith-Lovin, and James M Cook, *Birds of a feather: Homophily in social networks*, Annual Review of Sociology **27** (2001), no. 1, 415–444.

[43] Frank McSherry and Marc Najork, *Computing information retrieval performance measures efficiently in the presence of tied scores*, European conference on information retrieval, 2008, pp. 414–421.

[44] Edward Meeds, Zoubin Ghahramani, Radford M Neal, and Sam T Roweis, *Modeling dyadic data with binary latent factors*, Advances in neural information processing systems, 2006, pp. 977–984.

[45] Jörg Menche, Amitabh Sharma, Maksim Kitsak, Susan Dina Ghiassian, Marc Vidal, Joseph Loscalzo, and Albert-László Barabási, *Uncovering disease-disease relationships through the incomplete interactome*, Science **347** (2015), no. 6224, 1257601.

[46] Aditya Krishna Menon and Charles Elkan, *Link prediction via matrix factorization*, Joint european conference on machine learning and knowledge discovery in databases, 2011, pp. 437–452.

[47] K.T. Miller, T.L. Griffiths, and M.I. Jordan, *Nonparametric latent feature models for link prediction.*, In nips, 2009, pp. 1276–1284.

[48] Corrado Monti, A. Rozza, G. Zappella, M. Zignani, A. Arvidsson, and E. Colleoni, *Modelling political disaffection from twitter data*, Proc. of the 2nd int. wisdom, 2013, pp. 3.

[49] J. Mossong, N. Hens, M. Jit, et al., *Social contacts and mixing patterns relevant to the spread of infectious diseases*, PLoS Med **5** (2008), no. 3, e74.

[50] Krzysztof Nowicki and Tom A B Snijders, *Estimation and prediction for stochastic blockstructures*, Journal of the American Statistical Association **96** (2001), no. 455, 1077–1087.

[51] Joseph J Pfeiffer III, Sebastian Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher, *Attributed graph models: Modeling network structure with correlated attributes*, Proceedings of the 23rd international conference on world wide web, 2014, pp. 831–842.

[52] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios, *K-nearest neighbors in uncertain graphs*, Proceedings of the VLDB Endowment **3** (2010), no. 1-2, 997–1008.

[53] Frank Rosenblatt, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review **65** (1958), no. 6, 386.

[54] ———, *Principles of neurodynamics. perceptrons and the theory of brain mechanisms*, DTIC Document, 1961.

[55] Stuart Russell and Peter Norvig, *Artificial intelligence: A modern approach. 2010*, Prentice Hall, 2010.

[56] D Sculley, *Online active learning methods for fast label-efficient spam filtering.*, Ceas, 2007, pp. 143.

[57] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-june Paul Hsu, and Kuansan Wang, *An overview of microsoft academic service (mas) and applications*, Proceedings of the 24th international conference on world wide web, 2015, pp. 243–246.

[58] Tom A.B. Snijders and Krzysztof Nowicki, *Estimation and prediction for stochastic blockmodels for graphs with latent block structure*, Journal of Classification **14** (1997), no. 1, 75–100.

[59] Samuel A Stouffer, Louis Guttman, Edward A Suchman, Paul F Lazarsfeld, Shirley A Star, and John A Clausen, *Measurement and prediction.* (1950).

[60] Lloyd N Trefethen and David Bau III, *Numerical linear algebra*, Vol. 50, Siam, 1997.

[61] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok, *Interpreting tf-idf term weights as making relevance decisions*, ACM Transactions on Information Systems (TOIS) **26** (2008), no. 3, 13.

[62] Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel, *Learning infinite hidden relational models*, Uncertainity in Artificial Intelligence (UAI2006) (2006).

[63] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla, *Evaluating link prediction methods*, Knowledge and Information Systems **45** (2015), no. 3, 751–782.