Inference in networks

Cristopher Moore, Santa Fe Institute

joint work with
Aaron Clauset, Mark Newman,
Xiaoran Yan, Yaojia Zhu,
Lenka Zdeborová, Florent Krzakala,
Aurelien Decelle, Pan Zhang,
Jean-Baptiste Rouquier, and Tiffany Pierce
Learning statistics
Learning statistics

I used to think correlation implied causation.
Learning statistics

I used to think correlation implied causation.

Then I took a statistics class. Now I don't.
Learning statistics

I used to think correlation implied causation.

Then I took a statistics class. Now I don't.

Sounds like the class helped.

Well, maybe.
What is structure?
What is structure?

Structure is that which...
What is structure?

Structure is that which...

makes data different from noise: makes a network different from a random graph
What is structure?

Structure is that which...

- makes data different from noise: makes a network different from a random graph
- helps us compress the data: describe the network succinctly
What is structure?

Structure is that which...

- makes data different from noise: makes a network different from a random graph
- helps us compress the data: describe the network succinctly
- helps us generalize from data we’ve seen from data we haven’t seen:
What is structure?

Structure is that which...

- makes data different from noise: makes a network different from a random graph
- helps us compress the data: describe the network succinctly
- helps us generalize from data we’ve seen from data we haven’t seen:
  from one part of the network to another,
What is structure?

Structure is that which...

- makes data different from noise: makes a network different from a random graph
- helps us compress the data: describe the network succinctly
- helps us generalize from data we’ve seen from data we haven’t seen:
  - from one part of the network to another,
- or from one network to others generated by the same process
What is structure?

Structure is that which...

- makes data different from noise: makes a network different from a random graph
- helps us compress the data: describe the network succinctly
- helps us generalize from data we’ve seen from data we haven’t seen: from one part of the network to another,
  or from one network to others generated by the same process
- helps us coarse-grain the dynamics (?)
What is structure?

Structure is that which...

- makes data different from noise: makes a network different from a random graph
- helps us compress the data: describe the network succinctly
- helps us generalize from data we’ve seen from data we haven’t seen: from one part of the network to another, or from one network to others generated by the same process
- helps us coarse-grain the dynamics (?)
What is structure?

Structure is that which...

- makes data different from noise: makes a network different from a random graph
- helps us compress the data: describe the network succinctly
- helps us generalize from data we’ve seen from data we haven’t seen: from one part of the network to another,
- or from one network to others generated by the same process
- helps us coarse-grain the dynamics (?)
Statistical inference
Statistical inference

imagine that our data $G$ is drawn from an ensemble, or “generative model”: some probability distribution $P(G|\theta)$ with parameters $\theta$
Statistical inference

imagine that our data $G$ is drawn from an ensemble, or “generative model”: some probability distribution $P(G|\theta)$ with parameters $\theta$

$\theta$ can be continuous or discrete: represents the structure of the data
Statistical inference

imagine that our data $G$ is drawn from an ensemble, or “generative model”: some probability distribution $P(G|\theta)$ with parameters $\theta$

$\theta$ can be continuous or discrete: represents the structure of the data
given $G$, find the $\theta$ that maximize $P(G|\theta)$
Statistical inference

imagine that our data $G$ is drawn from an ensemble, or “generative model”: some probability distribution $P(G|\theta)$ with parameters $\theta$

$\theta$ can be continuous or discrete: represents the structure of the data

given $G$, find the $\theta$ that maximize $P(G|\theta)$

or (Bayes) compute the posterior distribution $P(\theta|G)$
Statistical inference

imagine that our data $G$ is drawn from an ensemble, or “generative model”: some probability distribution $P(G|\theta)$ with parameters $\theta$

$\theta$ can be continuous or discrete: represents the structure of the data
given $G$, find the $\theta$ that maximize $P(G|\theta)$
or (Bayes) compute the posterior distribution $P(\theta|G)$
The Erdős-Renyí model
The Erdős-Renyí model

every pair of vertices $i, j$ is connected independently with probability $p$
The Erdős-Renyí model

every pair of vertices $i, j$ is connected independently with probability $p$

average degree $d=np$
The Erdős-Renyí model

every pair of vertices $i, j$ is connected independently with probability $p$

average degree $d=np$

degree distribution is Poisson with mean $d$
The Erdős-Renyí model

every pair of vertices $i, j$ is connected independently with probability $p$

average degree $d=np$

degree distribution is Poisson with mean $d$

if $d<1$, almost all components are trees, and max component has size $O(\log n)$
The Erdős-Renyí model

every pair of vertices $i, j$ is connected independently with probability $p$

average degree $d=np$

degree distribution is Poisson with mean $d$

if $d<1$, almost all components are trees, and max component has size $O(\log n)$

if $d>1$, a unique giant component appears
The Erdős-Renyí model

every pair of vertices $i, j$ is connected independently with probability $p$

average degree $d=np$

degree distribution is Poisson with mean $d$

if $d<1$, almost all components are trees, and max component has size $O(\log n)$

if $d>1$, a unique giant component appears

at $d=\ln n$, completely connected
The Erdős-Renyí model

every pair of vertices $i, j$ is connected independently with probability $p$

average degree $d=np$

degree distribution is Poisson with mean $d$

if $d<1$, almost all components are trees, and max component has size $O(\log n)$

if $d>1$, a unique giant component appears

at $d=\ln n$, completely connected

ring + Erdős-Renyí = Watts-Strogatz
The Erdős-Renyí model

every pair of vertices \( i, j \) is connected independently with probability \( p \)

average degree \( d=np \)

degree distribution is Poisson with mean \( d \)

if \( d<1 \), almost all components are trees, and max component has size \( O(\log n) \)

if \( d>1 \), a unique giant component appears

at \( d=\ln n \), completely connected

ring + Erdős-Renyí = Watts-Strogatz
The stochastic block model
The stochastic block model
take a discrete attribute into account:
The stochastic block model

take a discrete attribute into account:

each vertex $i$ has a type $t_i \in \{1,\ldots,k\}$, with prior distribution $q_1,\ldots,q_k$
The stochastic block model

take a discrete attribute into account:

each vertex $i$ has a type $t_i \in \{1,\ldots,k\}$, with prior distribution $q_1,\ldots,q_k$

$k\times k$ matrix $p$
The stochastic block model

take a discrete attribute into account:

each vertex $i$ has a type $t_i \in \{1,\ldots,k\}$, with prior distribution $q_1,\ldots,q_k$

$k \times k$ matrix $p$

if $t_i = r$ and $t_j = s$, there is an edge $i \to j$ with probability $p_{rs}$
The stochastic block model

take a discrete attribute into account:

each vertex $i$ has a type $t_i \in \{1,...,k\}$, with prior distribution $q_1,...,q_k$

$k \times k$ matrix $p$

if $t_i = r$ and $t_j = s$, there is an edge $i \rightarrow j$ with probability $p_{rs}$

$p$ is not necessarily symmetric, and we don’t assume that $p_{rr} > p_{rs}$
The stochastic block model

take a discrete attribute into account:

each vertex $i$ has a type $t_i \in \{1, \ldots, k\}$, with prior distribution $q_1, \ldots, q_k$

$k \times k$ matrix $p$

if $t_i = r$ and $t_j = s$, there is an edge $i \rightarrow j$ with probability $p_{rs}$

$p$ is not necessarily symmetric, and we don’t assume that $p_{rr} > p_{rs}$

given a graph $G$, we want to simultaneously
The stochastic block model

take a discrete attribute into account:

each vertex \( i \) has a type \( t_i \in \{1,...,k\} \), with prior distribution \( q_1,...,q_k \)

\( k \times k \) matrix \( p \)

if \( t_i = r \) and \( t_j = s \), there is an edge \( i \rightarrow j \) with probability \( p_{rs} \)

\( p \) is not necessarily symmetric, and we don’t assume that \( p_{rr} > p_{rs} \)

given a graph \( G \), we want to simultaneously

label the nodes, i.e., infer the type assignment \( t : V \rightarrow \{1,...,k\} \)
The stochastic block model

take a discrete attribute into account:

each vertex $i$ has a type $t_i \in \{1,\ldots,k\}$, with prior distribution $q_1,\ldots,q_k$

$k \times k$ matrix $p$

if $t_i = r$ and $t_j = s$, there is an edge $i \rightarrow j$ with probability $p_{rs}$

$p$ is not necessarily symmetric, and we don’t assume that $p_{rr} > p_{rs}$

given a graph $G$, we want to simultaneously

label the nodes, i.e., infer the type assignment $t : V \rightarrow \{1,\ldots,k\}$

learn how nodes affect connections, i.e., infer the matrix $p$
The stochastic block model

take a discrete attribute into account:

each vertex \( i \) has a type \( t_i \in \{1,...,k\} \), with prior distribution \( q_1,...,q_k \)

\( k \times k \) matrix \( p \)

if \( t_i = r \) and \( t_j = s \), there is an edge \( i \rightarrow j \) with probability \( p_{rs} \)

\( p \) is not necessarily symmetric, and we don’t assume that \( p_{rr} > p_{rs} \)

given a graph \( G \), we want to simultaneously

- label the nodes, i.e., infer the type assignment \( t : V \rightarrow \{1,...,k\} \)
- learn how nodes affect connections, i.e., infer the matrix \( p \)

how do we get off the ground?
Assortative and disassortative
The likelihood
The likelihood

the probability of $G$ given the types $t$ and parameters $\theta=(p,q)$ is

$$P(G \mid t, \theta) = \prod_{(i,j) \in E} p_{t_i,t_j} \prod_{(i,j) \notin E} (1 - p_{t_i,t_j})$$
The likelihood

the probability of $G$ given the types $t$ and parameters $\theta=(p,q)$ is

$$P(G \mid t, \theta) = \prod_{(i,j)\in E} p_{t_i,t_j} \prod_{(i,j)\notin E} (1 - p_{t_i,t_j})$$

so the probability of $t$ given $G$ is

$$P(t \mid G, \theta) = \frac{P(t \mid \theta) P(G \mid t, \theta)}{\sum_{t'\in\{1,\ldots,k\}^n} P(G \mid t', \theta)} \propto \prod_{i\in V} q_{t_i} \prod_{(i,j)\in E} p_{t_i,t_j} \prod_{(i,j)\notin E} (1 - p_{t_i,t_j})$$
The likelihood

the probability of $G$ given the types $t$ and parameters $\theta=(p,q)$ is

$$P(G \mid t, \theta) = \prod_{(i,j) \in E} p_{t_i,t_j} \prod_{(i,j) \notin E} (1 - p_{t_i,t_j})$$

so the probability of $t$ given $G$ is

$$P(t \mid G, \theta) = \frac{P(t \mid \theta) P(G \mid t, \theta)}{\sum_{t' \in \{1, \ldots, k\}^n} P(G \mid t', \theta)}$$

$$\propto \prod_{i \in V} q_{t_i} \prod_{(i,j) \in E} p_{t_i,t_j} \prod_{(i,j) \notin E} (1 - p_{t_i,t_j})$$

call this the Gibbs distribution on $t$. How do we maximize it, or sample from it?
Maximizing the likelihood
Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type
Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type
if we like, we can jointly maximize $P(G|t,\theta)$ as a function of $t$ and $p$ by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}, \quad q_r = \frac{n_r}{n}$$
Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type

if we like, we can jointly maximize $P(G|t,\theta)$ as a function of $t$ and $p$ by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}, \quad q_r = \frac{n_r}{n}$$

this works reasonably well on small networks...
I record that I was born on a Friday
Maximizing the likelihood
Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type
if we like, we can jointly maximize $P(G|t,p)$ as a function of $t$ and $p$ by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}, \quad q_r = \frac{n_r}{n}$$

this works reasonably well on small networks... but it isn’t really what we want
Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type

if we like, we can jointly maximize $P(G|t,p)$ as a function of $t$ and $p$ by setting

$$p_{rs} = \frac{e_{rs}}{n_r n_s}, \quad q_r = \frac{n_r}{n}$$

this works reasonably well on small networks... but it isn’t really what we want

the probability of $\theta$ given $G$ is a proportional to a partition function

$$P(G | \theta) = \sum_{t \in \{1, \ldots, k\}^n} P(G | t, \theta)$$
Maximizing the likelihood

single-site heat-bath dynamics: choose a random vertex and update its type

if we like, we can jointly maximize \( P(G|t,p) \) as a function of \( t \) and \( p \) by setting

\[
p_{rs} = \frac{e_{rs}}{n_r n_s}, \quad q_r = \frac{n_r}{n}
\]

this works reasonably well on small networks... but it isn’t really what we want

the probability of \( \theta \) given \( G \) is a proportional to a partition function

\[
P(G|\theta) = \sum_{t \in \{1,\ldots,k\}^n} P(G|t,\theta)
\]

and \(-\log P(G|\theta)\) is a free energy, not a ground state energy
Belief propagation (a.k.a. the cavity method)
Belief propagation (a.k.a. the cavity method)

Each vertex $i$ sends a “message” to each of its neighbors $j$, giving $i$’s marginal distribution based on its other neighbors $k$. 
Belief propagation (a.k.a. the cavity method)

Each vertex $i$ sends a “message” to each of its neighbors $j$, giving $i$’s marginal distribution based on its other neighbors $k$

denote this message $\mu_{r \rightarrow j}^i = \text{estimate of } \Pr[t_i = r] \text{ if } j \text{ were absent}$
Belief propagation (a.k.a. the cavity method)

Each vertex $i$ sends a “message” to each of its neighbors $j$, giving $i$’s marginal distribution based on its other neighbors $k$

denote this message $\mu_{r \rightarrow j}^i = \text{estimate of } \Pr[t_i = r] \text{ if } j \text{ were absent}$

How do we update it?
Belief propagation (a.k.a. the cavity method)
Belief propagation (a.k.a. the cavity method)

\[
\mu^i_j = \frac{1}{Z_{i\rightarrow j}} q_s \prod_{k \neq j} \sum_{(i,k) \in E} \mu^k_i p_{rs} \times \prod_{k \neq j} \sum_{(i,k) \notin E} \mu^k_i (1 - p_{rs})
\]
Belief propagation (a.k.a. the cavity method)

\[
\mu_{i \rightarrow j}^s = \frac{1}{Z_{i \rightarrow j}} q_s \prod_{k \neq j} \sum_{(i,k) \in E} \mu_{k \rightarrow i}^r p_{rs} \times \prod_{k \neq j} \sum_{(i,k) \notin E} \mu_{r \rightarrow i}^k (1 - p_{rs})
\]
Belief propagation (a.k.a. the cavity method)

Belief propagation on a complete graph — takes $O(n^2)$ time to update

\[
\mu^{i \rightarrow j}_s = \frac{1}{Z_{i \rightarrow j}} q_s \prod_{k \neq j} \sum_{(i,k) \in E} \mu^{k \rightarrow i}_r p_{rs} \times \prod_{r} \sum_{k \neq j} \mu^{k \rightarrow i}_r (1 - p_{rs})
\]
Belief propagation (a.k.a. the cavity method)

\[ \mu_{s \rightarrow j} = \frac{1}{Z_{i \rightarrow j}} q_s \prod_{k \neq j} \sum_{(i,k) \in E} \mu_r^{k \rightarrow i} p_{rs} \times \prod_{k \neq j} \sum_{(i,k) \notin E} \mu_r^{k \rightarrow i} (1 - p_{rs}) \]

BP on a complete graph — takes \(O(n^2)\) time to update

can simplify by assuming that \(\mu_r^{k \rightarrow i} = \mu_r^k\) for all non-neighbors \(i\)
Belief propagation (a.k.a. the cavity method)

Belief propagation
\[
\mu_{s \rightarrow j} = \frac{1}{Z_{i \rightarrow j}} q_s \prod_{k \neq j} \sum_{(i,k) \in E} \mu_{r}^{k \rightarrow i} p_{rs} \times \prod_{r} \sum_{k \neq j} \mu_{r}^{k \rightarrow i}(1 - p_{rs})
\]

BP on a complete graph — takes \(O(n^2)\) time to update

can simplify by assuming that \(\mu_{r}^{k \rightarrow i} = \mu_{r}^{k}\) for all non-neighbors \(i\)

each vertex \(k\) applies an “external field” \(\sum_{r} \mu_{r}^{k}(1 - p_{rs})\) to all vertices of type \(s\)
Belief propagation (a.k.a. the cavity method)

\[
\mu_{i \rightarrow j}^s = \frac{1}{Z_{i \rightarrow j}} q_s \prod_{k \neq j} \sum_{r \in E_{(i,k)}} \mu_{r \rightarrow i}^k \cdot \sum_{r \in E_{(i,k)}} \mu_{r \rightarrow i}^k (1 - p_{rs})
\]
Belief propagation (a.k.a. the cavity method)

\[
\mu_{s}^{i \to j} = \frac{1}{Z_{i \to j}} q_{s} \prod_{k \neq j} \sum_{r} \mu_{r}^{k \to i} p_{rs} \times \frac{\prod_{k} \sum_{r} \mu_{r}^{k}(1 - p_{rs})}{\prod_{k: (i,k) \in E} \sum_{r} \mu_{r}^{k}(1 - p_{rs})}
\]
Belief propagation (a.k.a. the cavity method)

\[
\mu_{i \rightarrow j} = \frac{1}{Z_{i \rightarrow j}} q_s \prod_{k \neq j} \sum_{r} \mu_{r}^{k \rightarrow i} p_{rs} \times \frac{\prod_{k} \sum_{r} \mu_{r}^{k} (1 - p_{rs})}{\prod_{k:(i,k) \in E} \sum_{r} \mu_{r}^{k} (1 - p_{rs})}
\]

each update now takes \(O(n+m)\) time
Belief propagation (a.k.a. the cavity method)

\[
\mu_{i \rightarrow j} = \frac{1}{Z_{i \rightarrow j}} q_s \prod_{k \neq j} \sum_{(i,k) \in E} \mu_{k \rightarrow i} p_{rs} \times \frac{\prod_k \sum_r \mu_r^k (1 - p_{rs})}{\prod_{k:(i,k) \in E} \sum_r \mu_r^k (1 - p_{rs})}
\]

each update now takes \(O(n+m)\) time

update until the messages reach a fixed point
The karate club again
The karate club again

instructor

owner

Saturday, May 19, 2012
Which kind of community do you want?
Which kind of community do you want?
Two local optima

\[ -\text{free energy} \]

\[ \text{interpolation parameter } t \]

-0.2 0 0.2 0.4 0.6 0.8 1 1.2 1.4

1.5 2 2.5 3 3.5 4

(left/right) (high/low)

(i) (ii)
Degree-corrected block models
Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree
Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]
Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex $i$ has an expected degree $d_i$
Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex $i$ has an expected degree $d_i$

analogous to $p_{ij}$, a $k \times k$ matrix $w_{ij}$
Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex $i$ has an expected degree $d_i$

analogous to $p_{ij}$, a $k \times k$ matrix $w_{ij}$

for each pair $i, j$ with $t_i=r$ and $t_j=s$, the number of edges between them is

$$m_{ij} \sim \text{Poi}(d_i d_j w_{rs})$$
Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex $i$ has an expected degree $d_i$

analogous to $p_{ij}$, a $k \times k$ matrix $w_{ij}$

for each pair $i, j$ with $t_i=r$ and $t_j=s$, the number of edges between them is

$$m_{ij} \sim \text{Poi}(d_i d_j w_{rs})$$

now the degrees are parameters, not data to be explained
Degree-corrected block models

the “vanilla” block model expects vertices of the same type to have roughly the same degree

a random multigraph [Karrer & Newman, 2010]

each vertex $i$ has an expected degree $d_i$

analogous to $p_{ij}$, a $k \times k$ matrix $w_{ij}$

for each pair $i, j$ with $t_i=r$ and $t_j=s$, the number of edges between them is

$$m_{ij} \sim \text{Poi}(d_i d_j w_{rs})$$

now the degrees are parameters, not data to be explained

can again write down the BP/EM algorithm
ing corrected and uncorrected blockmodels with $K = 2$, we find the results shown in Fig. 1. As pointed out also by other authors [11, 30], the non-degree-corrected blockmodel fails to split the network into the known factions (indicated by the dashed line in the figure), instead splitting it into a group composed of high-degree vertices and another of low. The degree-corrected model, on the other hand, splits the vertices according to the known communities, except for the misidentification of one vertex on the boundary of the two groups. (The same vertex is also misplaced by a number of other commonly used community detection algorithms.)

The failure of the uncorrected model in this context is precisely because it does not take the degree sequence into account. The \textit{a priori} probability of an edge between two vertices varies as the product of their degrees, a variation that can be fit by the uncorrected blockmodel if we divide the network into high- and low-degree groups. Given that we have only one set of groups to assign, however, we are obliged to choose between this fit and the true community structure. In the present case it turns out that the division into high and low degrees gives the higher likelihood and so it is this division that the algorithm returns. In the degree-corrected blockmodel, by contrast, the variation of edge probability with degree is already included in the functional form of the likelihood, which frees up the block structure for fitting to the true communities.

Moreover it is apparent that this behavior is not limited to the case $K = 2$. For $K = 3$, the ordinary stochastic blockmodel will, for sufficiently heterogeneous degrees, be biased towards splitting into three groups by degree—high, medium, and low—and similarly for higher values of $K$. It is of course possible that the true community structure itself corresponds entirely or mainly to groups of high and low degree, but we only want our model to find this structure if it is still statistically surprising once we know about the degree sequence, and this is precisely what the corrected model does.

As a second real-world example we show in Fig. 2 an application to a network of political blogs assembled by Adamic and Glance [31]. This network is composed of blogs (i.e., personal or group web diaries) about US politics and the web links between them, as captured on a single day in 2005. The blogs have known political leanings and were labeled by Adamic and Glance as either liberal or conservative in the data set. We consider the network in undirected form and examine only the largest connected component, which has 1222 vertices. Figure 2 shows that, as with the karate club, the uncorrected stochastic blockmodel splits the vertices into high- and low-degree groups, while the degree-corrected model finds a split more aligned with the political division of the network. While not matching the known labeling exactly, the split generated by the degree-corrected model has a normalized mutual information of 0.72 with the labeling of Adamic and Glance, compared with 0.0001 for the uncorrected model.

(a) Without degree-correction
(b) With degree-correction

FIG. 2: Divisions of the political blog network found using the (a) uncorrected and (b) corrected blockmodels. The size of a vertex is proportional to its degree and vertex color reflects inferred group membership. The division in (b) corresponds roughly to the division between liberal and conservative blogs given in [31]. (To make sure that these results were not due to a failure of the heuristic optimization scheme, we also checked that the group assignments found by the heuristic have a higher objective score than the known group assignments, and that using the known assignments as the initial condition for the optimization recovers the same group assignments as found with random initial conditions.)

B. Generation of synthetic networks

We turn now to synthetic networks. The networks we use are themselves generated from the degree-corrected blockmodel [Karrer & Newman, 2010].
Blogs: degree-corrected block model

Figure 1: Comparison of degree-corrected and uncorrected blockmodels with $K = 2$. The non-degree-corrected block-model fails to split the network into the known factions (indicated by the dashed line in the figure), instead splitting it into a group composed of high-degree vertices and another of low. The degree-corrected model, on the other hand, splits the vertices according to the known communities, except for the misidentification of one vertex on the boundary of the two groups. (The same vertex is also misplaced by a number of other commonly used community detection algorithms.)

The failure of the uncorrected model in this context is precisely because it does not take the degree sequence into account. The prior probability of an edge between two vertices varies as the product of their degrees, a variation that can be fit by the uncorrected blockmodel if we divide the network into high- and low-degree groups. Given that we have only one set of groups to assign, however, we are obliged to choose between this fit and the true community structure. In the present case it turns out that the division into high and low degrees gives the higher likelihood and so it is this division that the algorithm returns. In the degree-corrected blockmodel, by contrast, the variation of edge probability with degree is already included in the functional form of the likelihood, which frees up the block structure for fitting to the true communities.

Moreover it is apparent that this behavior is not limited to the case $K = 2$. For $K = 3$, the ordinary stochastic blockmodel will, for sufficiently heterogeneous degrees, be biased towards splitting into three groups by degree—high, medium, and low—and similarly for higher values of $K$. It is of course possible that the true community structure itself corresponds entirely or mainly to groups of high and low degree, but we only want our model to find this structure if it is still statistically surprising once we know about the degree sequence, and this is precisely what the corrected model does.

As a second real-world example we show in Fig. 2 an application to a network of political blogs assembled by Adamic and Glance [31]. This network is composed of blogs (i.e., personal or group web diaries) about US politics and the web links between them, as captured on a single day in 2005. The blogs have known political leanings and were labeled by Adamic and Glance as either liberal or conservative in the data set. We consider the network in undirected form and examine only the largest connected component, which has 1222 vertices. Figure 2 shows that, as with the karate club, the uncorrected stochastic blockmodel splits the vertices into high- and low-degree groups, while the degree-corrected model finds a split more aligned with the political division of the network. While not matching the known labeling exactly, the split generated by the degree-corrected model has a normalized mutual information of $0.72$ with the labeling of Adamic and Glance, compared with $0.0001$ for the uncorrected model.

(a) Without degree-correction
(b) With degree-correction

FIG. 2: Divisions of the political blog network found using the (a) uncorrected and (b) corrected blockmodels. The size of a vertex is proportional to its degree and vertex color reflects inferred group membership. The division in (b) corresponds roughly to the division between liberal and conservative blogs given in [31].

(To make sure that these results were not due to a failure of the heuristic optimization scheme, we also checked that the group assignments found by the heuristic have a higher objective score than the known group assignments, and that using the known assignments as the initial condition for the optimization recovers the same group assignments as found with random initial conditions.)
Strengths and weaknesses
Strengths and weaknesses

degree-corrected models don’t mind inhomogeneous degree distributions...
Strengths and weaknesses

degree-corrected models don’t mind inhomogeneous degree distributions...
but they also can’t use the degrees to help them label the nodes
Strengths and weaknesses

degree-corrected models don’t mind inhomogeneous degree distributions...
but they also can’t use the degrees to help them label the nodes
on some networks, they perform worse than the vanilla model
Strengths and weaknesses

degree-corrected models don’t mind inhomogeneous degree distributions...
but they also can’t use the degrees to help them label the nodes
on some networks, they perform worse than the vanilla model
yet another model: first generate vertex degrees $d_i$ according to some
distribution whose parameters depend on $t_i$ (e.g. power law)
Strengths and weaknesses

degree-corrected models don’t mind inhomogeneous degree distributions...

but they also can’t use the degrees to help them label the nodes

on some networks, they perform worse than the vanilla model

yet another model: first generate vertex degrees $d_i$ according to some distribution whose parameters depend on $t_i$ (e.g. power law)

then generate edges according to the degree-corrected model
Strengths and weaknesses

degree-corrected models don’t mind inhomogeneous degree distributions...
but they also can’t use the degrees to help them label the nodes
on some networks, they perform worse than the vanilla model

yet another model: first generate vertex degrees $d_i$ according to some
distribution whose parameters depend on $t_i$ (e.g. power law)

then generate edges according to the degree-corrected model

for some networks (e.g. word adjacency networks) works better than either
vanilla or degree-corrected model
How can we tell if we’re on the right track?

it’s easy to fit data with a fancy model... the danger is overfitting
How can we tell if we’re on the right track?

it’s easy to fit data with a fancy model... the danger is overfitting
How can we tell if we’re on the right track?

it’s easy to fit data with a fancy model... the danger is overfitting
How can we tell if we’re on the right track?

It’s easy to fit data with a fancy model... the danger is overfitting.
How can we tell if we’re on the right track?

it’s easy to fit data with a fancy model... the danger is overfitting

can we generalize from part of the data to the rest of it?
Active learning
Active learning

suppose we can learn a node’s attributes, but at a cost
Active learning

suppose we can learn a node’s attributes, but at a cost

we want to make good guesses about most of the nodes, after querying just a few of them—which ones?
Active learning

Suppose we can learn a node’s attributes, but at a cost.

We want to make good guesses about most of the nodes, after querying just a few of them — which ones?

Query the node with the largest *mutual information* between it and the others:

\[
I(v, G - v) = H(v) - H(v | G - v) \\
= H(G - v) - H(G - v | v)
\]
Active learning

suppose we can learn a node’s attributes, but at a cost

we want to make good guesses about most of the nodes, after querying just a few of them—which ones?

query the node with the largest mutual information between it and the others:

\[
I(v, G - v) = H(v) - H(v | G - v) \\
= H(G - v) - H(G - v | v)
\]

average amount of information we learn about G-v we learn by querying v
Active learning

suppose we can learn a node’s attributes, but at a cost

we want to make good guesses about most of the nodes, after querying just a few of them—which ones?

query the node with the largest \textit{mutual information} between it and the others:

\[
I(v, G - v) = H(v) - H(v | G - v)
\]

\[
= H(G - v) - H(G - v | v)
\]

average amount of information we learn about $G-v$ we learn by querying $v$

high when we’re uncertain about $v$, and when $v$ is highly correlated with others
The Karate Club again

% vertices above thresholds

# of nodes queried

0.1 0.3 0.5 0.7 0.9

Saturday, May 19, 2012
Which vertices do we query first?
An antarctic food web
An antarctic food web

% vertices above thresholds

# of nodes queried

"unknown unknowns"
Which rubber, which road?
Which rubber, which road?

these topological methods are all very nice, but...
Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the function of nodes, and their dynamics?
Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the function of nodes, and their dynamics?
do they give us good coarse-grainings of the dynamics?
Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the function of nodes, and their dynamics?
do they give us good coarse-grainings of the dynamics?
can nodes in the same group “stand in” for each other under stress?
Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the function of nodes, and their dynamics?
do they give us good coarse-grainings of the dynamics?
can nodes in the same group “stand in” for each other under stress?
are there good generative models for benchmark power grids?
Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the function of nodes, and their dynamics?
do they give us good coarse-grainings of the dynamics?
can nodes in the same group “stand in” for each other under stress?
are there good generative models for benchmark power grids?

it’s easy to add attributes, e.g. geography, but we also need to generate loads, line capacities, generators...
Which rubber, which road?

these topological methods are all very nice, but...

what do they actually tell us about the *function* of nodes, and their dynamics?
do they give us good coarse-grainings of the dynamics?
can nodes in the same group “stand in” for each other under stress?
are there good generative models for benchmark power grids?

it’s easy to add attributes, e.g. geography, but we also need to generate loads, line capacities, generators...

off-the-cuff idea: points distributed like the fraction distribution of population, e.g. according to a Levy flight, and then connected geometrically
This book rocks! You somehow manage to combine the fun of a popular book with the intellectual heft of a textbook.

— Scott Aaronson

A treasure trove of information on algorithms and complexity, presented in the most delightful way.

— Vijay Vazirani

A creative, insightful, and accessible introduction to the theory of computing, written with a keen eye toward the frontiers of the field and a vivid enthusiasm for the subject matter.

— Jon Kleinberg

Oxford University Press, 2011
Acknowledgments

and the McDonnell Foundation