A Bayesian approach to network modularity

Jake Hofman
Wiggins Lab
Columbia University
2007.10.19
Outline

• Past work: image data

• Extension to modular networks

• Bayesian inference and complexity control

• Generating and inferring modular networks

• Validation and preliminary applications
Overview: modeling image data

• Given an image:
  • Assign pixels to clusters (foreground/background)?
Overview: modeling image data

With a generative model of intensity histograms, rules of probability tell us how to calculate model parameters (e.g. threshold)
Overview: modular networks

- Given a network:
  - Assign nodes to modules?
  - Determine number of modules (scale/complexity)?
Overview: modular networks

With a generative model of modular networks, rules of probability tell us how to calculate model parameters (e.g. number of modules & assignments)
Networks describe objects (nodes) and interactions (edges) between the objects.
Generative models

Know model (parameters, assignment variables, complexity)

Infer model (parameters, latent variables, complexity)

Generate synthetic data

Observe real data
Bayesian inference

• Given prior belief and data D, infer posterior distribution over model parameters $\theta$ and complexity $K$

• Use Bayes’ rule to “invert” probabilities

\[
p(\theta|\mathcal{D}, K) = \frac{p(\mathcal{D}|\theta, K)p(\theta|K)}{p(\mathcal{D}|K)}
\]

where

\[
p(\mathcal{D}|K) = \int d\theta \ p(\mathcal{D}|\theta, K)p(\theta|K)
\]
Inference: Coin flips
Inference: Coin flips

prior over coin fairness
\( \alpha_0 = 2, \beta_0 = 2 \)
Inference: Coin flips

observe flips:
HTHHHHTTTTHHHHHH
Inference: Coin flips

posterior over coin fairness
\( \alpha = 10, \beta = 5 \)

\[ p(\theta | D) \]

\[ \theta \]

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
Inference: Coin flips
Inference: Coin flips

prior over coin fairness
\( \alpha_0 = 2, \beta_0 = 2 \)
Inference: Coin flips

observe flips:

\[
\text{HHHHHHHHHHHHHHHTHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
\[
\text{HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH}
\]
Inference: Coin flips

posterior over coin fairness
\(\alpha=100, \beta=5\)
Bayesian complexity control

- Given prior belief and data $D$, infer posterior distribution over model complexity $K$

- If $p(K)$ sufficiently weak, maximize evidence to find optimal complexity

$$p(K | D) = \frac{p(D | K) p(K)}{p(D)}$$

$$p(D | K) = \int d\theta \, p(D | \theta, K) p(\theta | K)$$
Likelihood always increases with complexity
Evidence incorporates complexity control
http://research.microsoft.com/~minka/statlearn/demo/
Complexity control:
Optimal number of clusters

Likelihood always increases with complexity
Evidence incorporates complexity control
Complexity control:
Optimal number of clusters

Likelihood always increases with complexity
Evidence incorporates complexity control
Overview: modular networks

• Given a network:
  • Assign nodes to modules?
  • Determine number of modules (scale/complexity)?
Overview: modular networks

With a generative model of modular networks, rules of probability tell us how to calculate model parameters (e.g. number of modules & assignments)
Generating modular networks

- For each node:
  - **Roll K-sided die** to determine \( z_i = 1, \ldots, K \), the (unobserved) module assignment for \( i \)th node

- For each pair of nodes \((i, j)\):
  - If \( z_i = z_j \), **flip “in community” coin** with bias \( \theta_c \) to determine edge
  - If \( z_i \neq z_j \), **flip “between communities” coin** with bias \( \theta_d \) to determine edge
Generating modular networks

• For each node:
  • **Roll K-sided die** to determine $z_i = 1,\ldots,K$, the (unobserved) module assignment for $i^{th}$ node

• For each pair of nodes (i,j):
  • If $z_i = z_j$, **flip “in community” coin** with bias $\theta_c$ to determine edge
  • If $z_i \neq z_j$, **flip “between communities” coin** with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  • **Roll K-sided die** to determine \( z_i = 1, \ldots, K \), the (unobserved) module assignment for \( i \)th node

• For each pair of nodes \((i,j)\):
  • If \( z_i = z_j \), **flip “in community” coin** with bias \( \theta_c \) to determine edge
  • If \( z_i \neq z_j \), **flip “between communities” coin** with bias \( \theta_d \) to determine edge
Generating modular networks

- For each node:
  - Roll K-sided die to determine \( z_i = 1, \ldots, K \), the (unobserved) module assignment for \( i \)th node

- For each pair of nodes \((i,j)\):
  - If \( z_i = z_j \), flip “in community” coin with bias \( \theta_c \) to determine edge
  - If \( z_i \neq z_j \), flip “between communities” coin with bias \( \theta_d \) to determine edge
Generating modular networks

• For each node:
  • Roll K-sided die to determine $z_i=1,...,K$, the (unobserved) module assignment for $i^{th}$ node

• For each pair of nodes (i,j):
  • If $z_i=z_j$, flip “in community” coin with bias $\theta_c$ to determine edge
  • If $z_i \neq z_j$, flip “between communities” coin with bias $\theta_d$ to determine edge
Generating modular networks

For each node:

- **Roll K-sided die** to determine $z_i=1,...,K$, the (unobserved) module assignment for $i^{th}$ node

For each pair of nodes $(i,j)$:

- If $z_i=z_j$, **flip “in community” coin** with bias $\theta_c$ to determine edge
- If $z_i\neq z_j$, **flip “between communities” coin** with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  • Roll K-sided die to determine $z_i=1,...,K$, the (unobserved) module assignment for $i$th node

• For each pair of nodes (i,j):
  • If $z_i=z_j$, flip “in community” coin with bias $\theta_c$ to determine edge
  • If $z_i \neq z_j$, flip “between communities” coin with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  
  • **Roll K-sided die** to determine $z_i = 1, \ldots, K$, the (unobserved) module assignment for $i^{th}$ node

• For each pair of nodes $(i,j)$:
  
  • If $z_i = z_j$, **flip “in community” coin** with bias $\theta_c$ to determine edge
  
  • If $z_i \neq z_j$, **flip “between communities” coin** with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  • **Roll K-sided die** to determine $z_i=1,\ldots,K$, the (unobserved) module assignment for $i^{th}$ node

• For each pair of nodes (i,j):
  • If $z_i=z_j$, **flip “in community” coin** with bias $\theta_c$ to determine edge
  • If $z_i\neq z_j$, **flip “between communities” coin** with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  • Roll K-sided die to determine $z_i=1,...,K$, the (unobserved) module assignment for $i^{\text{th}}$ node

• For each pair of nodes (i,j):
  • If $z_i=z_j$, flip “in community” coin with bias $\theta_c$ to determine edge
  • If $z_i\neq z_j$, flip “between communities” coin with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  • Roll K-sided die to determine $z_i=1,...,K$, the (unobserved) module assignment for $i$th node

• For each pair of nodes (i,j):
  • If $z_i=z_j$, flip “in community” coin with bias $\theta_c$ to determine edge
  • If $z_i\neq z_j$, flip “between communities” coin with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  • Roll K-sided die to determine $z_i=1,...,K$, the (unobserved) module assignment for $i^{th}$ node

• For each pair of nodes (i,j):
  • If $z_i=z_j$, flip “in community” coin with bias $\theta_c$ to determine edge
  • If $z_i\neq z_j$, flip “between communities” coin with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  
  • **Roll K-sided die** to determine $z_i=1,...,K$, the (unobserved) module assignment for $i^{th}$ node

• For each pair of nodes (i,j):
  
  • If $z_i=z_j$, **flip “in community” coin** with bias $\theta_c$ to determine edge
  
  • If $z_i\neq z_j$, **flip “between communities” coin** with bias $\theta_d$ to determine edge
Generating modular networks

• For each node:
  
  • Roll K-sided die to determine $z_i=1,...,K$, the (unobserved) module assignment for $i$th node

• For each pair of nodes $(i,j)$:
  
  • If $z_i=z_j$, flip “in community” coin with bias $\theta_c$ to determine edge

  • If $z_i\neq z_j$, flip “between communities” coin with bias $\theta_d$ to determine edge
Generating modular networks

Die rolling, coin flipping, and priors:

\[ p(\vec{z} | \vec{\pi}) \equiv \prod_{\mu=1}^{K} \pi_{\mu}^{n_{\mu}} \]

\[ p(A | \vec{z}, \vec{\pi}, \vec{\theta}) \equiv \theta_{c}^{c+} (1 - \theta_{c})^{c-} \theta_{d}^{d+} (1 - \theta_{d})^{d-} \]

\[ p(\vec{\theta}) \equiv \mathcal{B}(\theta_c; \tilde{c}_+, \tilde{c}_-) \mathcal{B}(\theta_d; \tilde{d}_+, \tilde{d}_-) \]

\[ p(\vec{\pi}) \equiv \mathcal{D}(\vec{\pi}; \vec{n}) \]

where counts are:

\[ c_+ = \sum_{i,j} A_{ij} \delta_{z_i, z_j} \]

\[ c_- = \sum_{i,j} (1 - A_{ij}) \delta_{z_i, z_j} \]

\[ d_+ = \sum_{i,j} A_{ij} (1 - \delta_{z_i, z_j}) \]

\[ d_- = \sum_{i,j} (1 - A_{ij}) (1 - \delta_{z_i, z_j}) \]

\[ n_\mu = \sum_{i=1}^{N} \delta_{z_i, \mu} \]

Stochastic block models (Holland, Laskey, Leinhardt 1983)
Inferring modular networks

- From observed graph structure, infer distributions over module assignments, model parameters, and model complexity.

\[
p(\pi, \theta|A, K) = \frac{p(A|\pi, \theta, K)p(\pi, \theta|K)}{p(A|K)}
\]

\[
p(z|A, K) = \frac{p(A|z, K)p(z|K)}{p(A|K)}
\]

\[
p(A|K) = \sum_z \int d\theta \int d\pi \ p(A, z, \pi, \theta|K)
\]
Inferring modular networks

- From observed graph structure, infer distributions over module assignments, model parameters, and model complexity

\[
p(\vec{\pi}, \vec{\theta}|A, K) = \frac{p(A|\vec{\pi}, \vec{\theta}, K)p(\vec{\pi}, \vec{\theta}|K)}{p(A|K)}
\]

\[
p(\vec{z}|A, K) = \frac{p(A|\vec{z}, K)p(\vec{z}|K)}{p(A|K)}
\]

\[
p(A|K) = \sum_{\vec{z}} \int d\vec{\theta} \int d\vec{\pi} \ p(A, \vec{z}, \vec{\pi}, \vec{\theta}|K)
\]

Can do integrals, but sum is intractable, $O(K^N)$
Approximate inference for modular networks

- Jensen’s inequality (log of expected value bounds expected value of log) for any distribution $q$

\[
- \ln p(A|K) = - \ln \sum_{\vec{z}} \int d\theta \int d\vec{\pi} \ p(A, \vec{z}, \vec{\pi}, \theta|K) \\
= - \ln \sum_{\vec{z}} \int d\theta \int d\vec{\pi} \ q(\vec{z}, \vec{\pi}, \theta) \frac{p(A, \vec{z}, \vec{\pi}, \theta|K)}{q(\vec{z}, \vec{\pi}, \theta)} \\
\leq - \sum_{\vec{z}} \int d\theta \int d\vec{\pi} \ q(\vec{z}, \vec{\pi}, \theta) \ln \frac{p(A, \vec{z}, \vec{\pi}, \theta|K)}{q(\vec{z}, \vec{\pi}, \theta)} \\
\underbrace{F\{q;A\}}
\]

Variational Bayes (Feynman 1972; MacKay, Jordan, Ghahramani, Jaakola, Saul 1999)
Approximate inference for modular networks

• F is a functional of q; find approximation to posterior by optimizing approximation to evidence

• Take $q(z, \pi, \theta) = q(z)q(\pi)q(\theta)$; $Q_{i\mu}$ is probability node i in module $\mu$

$$F\{q; A\} = -\ln \frac{Z_{\pi}Z_cZ_d}{\tilde{Z}_{\pi}\tilde{Z}_c\tilde{Z}_d} + \sum_{\mu=1}^{K} \sum_{i=1}^{N} Q_{i\mu} \ln Q_{i\mu} - (\tilde{c}_+ - (\langle c_+ \rangle + \tilde{c}_{+0}))\langle \ln \theta_c \rangle$$

$$- (\tilde{c}_- - (\langle c_- \rangle + \tilde{c}_{+0}))\langle \ln(1 - \theta_c) \rangle - (\tilde{d}_+ - (\langle d_+ \rangle + \tilde{d}_{+0}))\langle \ln \theta_d \rangle$$

$$- (\tilde{d}_- - (\langle d_- \rangle + \tilde{d}_{-0}))\langle \ln(1 - \theta_d) \rangle - \sum_{\mu=1}^{K} (\tilde{n}_\mu - (\langle n_\mu \rangle + \tilde{n}_{\mu0}))\langle \ln \pi_\mu \rangle$$

where expected counts are:

$$\langle c_+ \rangle = \frac{1}{2} Tr(Q^T A Q)$$

$$\langle c_- \rangle = \frac{1}{2} Tr(Q^T \tilde{A} Q)$$

$$\langle d_+ \rangle = M - \langle c_+ \rangle$$

$$\langle d_- \rangle = C - M - \langle c_- \rangle$$

$$\langle n_\mu \rangle = \sum_{j=1}^{N} Q_{j\mu}$$
Initialization: Initialize the $N$-by-$K$ matrix $Q = Q_0$ and set the pseudocounts $\tilde{c}_+, \tilde{c}_-, \tilde{d}_+, \tilde{d}_-, \text{and } \tilde{n}_\mu$.

Main loop: Until convergence in $F\{q; A\}$,

1. update $\tilde{c}_+$, $\tilde{c}_-$, and $\tilde{n}_\mu$’s from expected counts and pseudocounts

   $\tilde{c}_+ = \langle c_+ \rangle + \tilde{c}_+ = \frac{1}{2} Tr(Q^T \tilde{A} Q) + \tilde{c}_+$ (14)

   $\tilde{c}_- = \langle c_- \rangle + \tilde{c}_- = \frac{1}{2} Tr(Q^T \tilde{A} Q) + \tilde{c}_-$ (15)

   $\tilde{d}_+ = \langle d_+ \rangle + \tilde{d}_+ = M - \langle c_+ \rangle + \tilde{d}_+$ (16)

   $\tilde{d}_- = \langle d_- \rangle + \tilde{d}_- = C - M - \langle c_- \rangle + \tilde{d}_-$ (17)

   $\tilde{n}_\mu = \langle n_\mu \rangle + \tilde{n}_\mu = \sum_{j=1}^{N} Q_{j\mu} + \tilde{n}_\mu$ (18)

   where $\tilde{A}$ is the logical negation of $A$, $C = N(N - 1)/2$, and $M = \frac{1}{2} \sum_{i,j} A_{ij}$;  

2. update expected value of coupling constants

   $\langle J_L \rangle = \frac{\ln \theta_d (1 - \theta_d)}{\theta_d (1 - \theta_d)} = \psi(\tilde{c}_+ + \psi(\tilde{c}_-) - \psi(\tilde{d}_+) + \psi(\tilde{d}_-) )$ (20)

   $\langle J_G \rangle = \frac{\ln (1 - \theta_d)}{(1 - \theta_d)} = -\psi(\tilde{c}_-) - \psi(\tilde{d}_-)$ (22)

   where $\psi(x)$ is the digamma function;

3. update $Q$ as

   $Q \leftarrow \frac{1}{Z} e^{\langle J_L \rangle A Q - \langle J_G \rangle \langle \tilde{n} \rangle + \langle \ln \tilde{\pi} \rangle}$ (23)

   where $\langle \ln \pi_\mu \rangle = \psi(\tilde{n}_\mu) - \psi(\sum_\mu \tilde{n}_\mu)$ and $Z$ is set by the normalization $\sum_\mu Q_{j\mu} = 1$;

4. calculate the optimal free energy under the updated parameter distributions

   $F\{q; A\} = -\ln \frac{Z_c Z_d Z_{\tilde{d}}}{Z_c Z_d Z_{\tilde{d}}} + \sum_{j=1}^{K} \sum_{i=1}^{N} Q_{j\mu} \ln Q_{j\mu}$ (24)

• Iteratively optimize $F\{q; A\}$ by updating distributions over parameters $\{\pi, \theta\}$ and latent variables $\{z\}$
Validation: Toy graph

- Automatic complexity control: probability of occupation for extraneous modules goes to zero
Validation: Toy graph

- Automatic complexity control: probability of occupation for extraneous modules goes to zero
Validation: Complexity control

- Comparison of our method (VB) to alternative method (ICL, based on BIC) for synthetic N=60 node networks and $K_{\text{True}}=3,4,5$ modules
Validation: Large-scale network
Validation: Runtime

- Main loop runtime for $10^4$ nodes in MATLAB ~30 seconds
Preliminary Applications: Zachary’s karate network
Preliminary Applications: Zachary’s karate network
Preliminary Applications: Zachary’s karate network
Conclusions

• Extended work on image data to phrase modularity as a modeling problem

• Used laws of probability to infer distributions over module assignments, model parameters, and model complexity

• Resulted in a principled, accurate, and scalable algorithm

• Validated technique on synthetic and real networks

• Future: apply to protein-protein network to determine functional modules; extend model to handle alternative network structure

• preprint: http://arxiv.org/abs/0709.3512
Acknowledgments

• Chris Wiggins

• Useful discussions:
  • Joel Bader (Johns Hopkins NDC)
  • Matt Hastings (LANL)
  • Aaron Clauset (SFI)