Comparison of Cross-Validation Methods for Stochastic Block Model

Beau Dabbs

February 1, 2016

1 Introduction

1. Motivate Model Selection
    - Also use accurate community membership recovery as an additional motivation, since we are considering NMI.
2. Methods used for networks in general
    - BIC (Example?)
    - CV - edge (used in Hoff (2008)) and network (Chen and Lei, 2014)
3. Methods used to select communities (or among SBM)
    - Eigenanalysis (Example?)
    - Recursive Bipartition (Bickel and Sarkar, 2015) - Compared to networkCV in (Chen and Lei, 2014)
    - Community Detection Methods
        - Modularity Maximization (Find a standard way to do this for directed networks)
        - Infomap (Rosvall and Bergstrom, 2008) (Shown to be best in Lancichinetti and Fortunato (2009))
4. We consider various cross-validation methods for network data. We use the regime of Stochastic Block Models as an example.

- What method of cross-validation is preferred for Stochastic Block Models
- How does CV compare to methods designed for community detection and SBM?

2 Methods

2.1 Cross-Validation Methods

The basic method of cross-validation requires us to divide the data, once or multiple times, into validation and training sets. Model parameters are then estimated using the training set, and the model's performance is assessed on the validation set. In this paper we will focus on V-fold cross-validation where each datum, in our case observation of edge-values on node-pairs, is assigned to one of V folds, and each fold takes place once as the validation set, with the other folds belonging to the training set. Thus any cross-validation method depends on the following three choices

1. Method of assigning node-pairs to folds
2. Number of folds, V
3. Loss Function

We'll address the options for assigning node-pairs to folds in this section. In section **??** we will address the affect of the number of folds on cross-validation methods in general, and briefly discuss the choices of loss function.

Before we begin describing the various cross-validation methods we will consider, let us set up some notation. Since our dataset consists of an $n \times n$ matrix of observed edges, Y, we will assign each node pair (i, j) with $1 \leq i \neq j \leq n$ to one of V folds. We represent this with an $n \times n$ matrix A where $A_{ij} = v$ means that node pair (i, j) is assigned to fold v. To implement any cross-validation method we will do the following

1. Generate a fold assignment matrix A
2. For each model M_1,...,M_T do:
    (a) For $v = 1$ to V do:
        i. Estimate parameters $\hat{\beta}_v^{CV}$ and latent variables $\hat{Z}_v^{CV}$ for model M_t using only $Y^{-v} := \{Y_{ij} : A_{ij} \neq v\}$
        ii. Estimate $\hat{p}_{ij}^{CV,M_t} = \mathbb{P}(Y_{ij} = 1 | \hat{\beta}_v^{CV}, \hat{Z}_v^{CV})$ for each i and j with $A_{ij} = v$.
    (b) Define $\hat{R}_{M_t}^{CV}(Y) = L(Y, \hat{P}^{CV,M_t}) = \frac{1}{n(n-1)} \sum_{i \neq j} L(Y_{ij}, \hat{p}_{ij}^{CV,M_t})$
3. Return $\hat{R}_{M_1}^{CV}$, ..., $\hat{R}_{M_T}^{CV}$

To perform model selection, we then choose the model that minimizes $\hat{R}_{M_t}$. For now let us focus on processes for generating fold assignment matrices, A.

2.1.1 Random Edge CV

For this subsection and the following subsections assume that V is a fixed number of folds for which we wish to generate a fold assignment matrix. The simplest way to generate a fold assignment matrix for V folds is to assign each edge pair to a fold uniformly at random. To guarantee balance in the size of folds we start with a matrix A which has an equal number of occurrences of each fold 1 through V. We then randomize the locations of all elements of A to generate A_{edge}. We will refer to cross-validation using A_{edge} as *edgeCV*. Figure 2 shows an example of A_{edge}.

Due to the unrestricted nature of this randomization, there is no guarantee that there will not be some folds that contain all node pairs associated with a particular node, i, and thus make it difficult for any model to successfully estimate the edge values, Y_{ij} or Y_{ji} for any j. Our next method attempts to restrict the randomization process to guarantee balance of the fold assignment among nodes.

2.1.2 Latin CV

The goal of latinCV, inspired by latin squares, is to guarantee that each row and each column of A has an equal number of occurrences of each fold. To generate such a matrix, we begin with a matrix that trivially satisfies this criterion. Divide the matrix A into V^2 blocks where each block is a $\frac{n}{V}$ by $\frac{n}{V}$ set of edge pairs. (See figure 1.) Thus we have our matrix represented as a V by V block matrix. We then assign folds $\{1, 2, ..., V\}$ in order to the first row in the block matrix. For the second row we assign folds $\{2, ..., V, 1\}$, and so on until for the Vth row we assign the folds $\{V, 1, ..., V-1\}$. To get a random fold assignment, we then independently reorder the rows of A and then the columns of A. Both of these reorderings preserve the balance among the rows and columns, resulting in our final fold assignment matrix which we will call A_{latin}. Figure 2 shows an example of the randomized matrix.

2.1.3 networkCV

Network Cross-validation refers to a method described in Chen and Lei (2014). To begin we assign each node to one of V folds, $L_1, ..., L_n \in \{1, ..., V\}$. Assume that the nodes are ordered by their fold assignment (note this is easily accomplished by using the same permutation on both the rows and columns of the adjacency matrix). We then assign each node pair to a

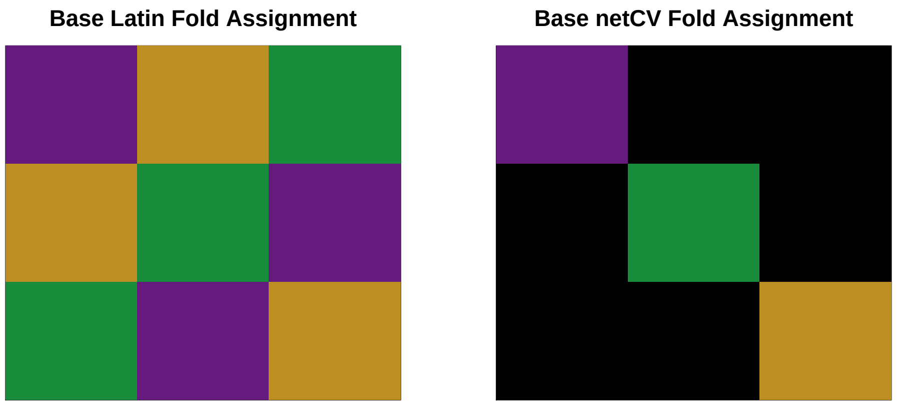

Figure 1: Base 3 fold assignments for latinCV and netCV. The colors purple, green, and gold correspond to folds 1-3. Node-pairs colored black are never in test sets, but are always in training sets for netCV.

fold by the following rule

$$A_{ij} = \begin{cases} v & L_i = L_j = v \\ 0 & L_i \neq L_j \end{cases} \tag{1}$$

This assignment results in assigning node pairs in V diagonal blocks to non-zero folds, and all of the other node pairs to fold zero. During the cross-validation procedure edges assigned to fold zero never appear in a validation set, but always appear in the training set for each fold.

One consequence of defining the folds this way is that the size of each training set is $N\frac{V^2-1}{V^2}$, where N is the total number of edges in the network, thus the training set sizes are comparable to those in V^2 fold random edge CV. However the most important consequence of this assignment procedure is that the total number of edges used in validation sets is $\frac{N}{V}$, meaning the size of the validation set decreases as we increase V.

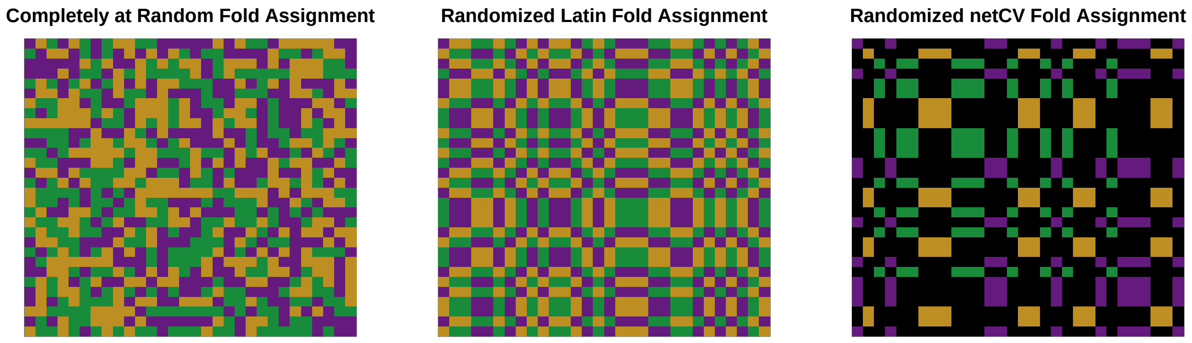
Completely at Random Fold Assignment
Randomized Latin Fold Assignment
Randomized netCV Fold Assignment

Figure 2: The figure on the left shows a realization of a completely random fold assignment. The center plot shows an example of latinCV fold assignment after randomizing each axis independently axes. The right plot shows a netCV fold assignment after randomizing the axes together.

2.2 Model Selection Methods

2.3 Community Detection Methods

3 Simulation Study

3.1 Simulation Study Design

Our goal is to compare the performance of the methods described above on simulated data. For this simulation study we are focusing on networks generated from the *Stochastic Block Model* (SBM), initially introduced in Snijders and Nowicki (1997).

3.1.1 Data Generation Parameters

When simulating data for any experiment one important consideration is the sample size. For network data the sample size, n, is simply the number of nodes in the network. To determine a range of node sizes from which to simulate networks we surveyed the social network literature. Figure 3 shows a histogram of the number of nodes found from that survey. Most of the networks contained fewer than 300 nodes, with a few networks having more than 1000 nodes. For our simulation study we will generate networks with $n = 30, 60, 120, 300$, and 1000 nodes.

While surveying the literature, we also kept track of the number of blocks either known to be present or used to fit a model. A large majority of the networks had ten or fewer blocks, and thus we will simulate data from SBMs with $K = 1, ..., 10$ blocks.

Once we know the number of nodes and the number of blocks from which to simulate networks, we still need to know two more specifications. First we need to know how to assign memberships, π_i, for each node i. We start by considering membership assignments that assign an equal number of nodes to each of the K possible blocks. Then we consider assignments that result in a power law distribution on the block sizes (**Add specificity here**).

Lastly we need to know the probability of there being a tie between a node in block l and another node in block k for all $l, k = 1, ..., K$. This set of *block tie probabilities* is generally represented using a matrix B where the entries are each $b_{lk} = \mathbb{P}(Y_{ij} = 1 | \pi_i = l, \pi_j = k)$. Many examples where an SBM seems appropriate are those where each block corresponds to a community of nodes that are more connected with each other than with those outside of the community. For this reason we consider a set of tie probability matrices where the probability of a tie between nodes in different blocks is fixed at some constant, b, and the probability of a tie between nodes in the same block is rb for some $r > 1$. Networks generated with these types of tie probability matrices also conform to the assumptions of community detection methods such as modularity maximization reviewed in section 2.3. We also consider tie probability matrices where each element $b_{lk} \sim Uniform(0, 1)$. While tie probability matrices of this form are rare, some datasets have been found that have tie densities that have no obvious form - see for example the network of neural connections in C. Elegans (Pavlovic et al., 2014).

3.1.2 Model Estimation

Since we are simulating data from stochastic block models with 1 to 10 blocks, we chose to estimate model parameters for SBMs with 1 to 11 blocks. This set of estimating models allows the opportunity to choose more than the true number of blocks for each K in the simulation study. To estimate parameters for the Stochastic Block Model we will be using a variational EM algorithm based on the mean-field approximation (**?**). To initialize the algorithm we use spectral clustering on the adjacency matrix to obtain an initial set of block memberships, similar to the method in Chen and Lei (2014).

Since we are using a cross-validation procedure to estimate risk for these models, our estimation methods need to be able to deal with missing data. The EM algorithm can deal with missing data by simply computing the conditional likelihood marginalized over the missing ties. Due to the conditional independence of the ties, marginalizing over any single tie is equivalent to computing the likelihood without that tie. Unfortunately, spectral clustering requires a complete dissimilarity matrix to perform the singular value decomposition. Since

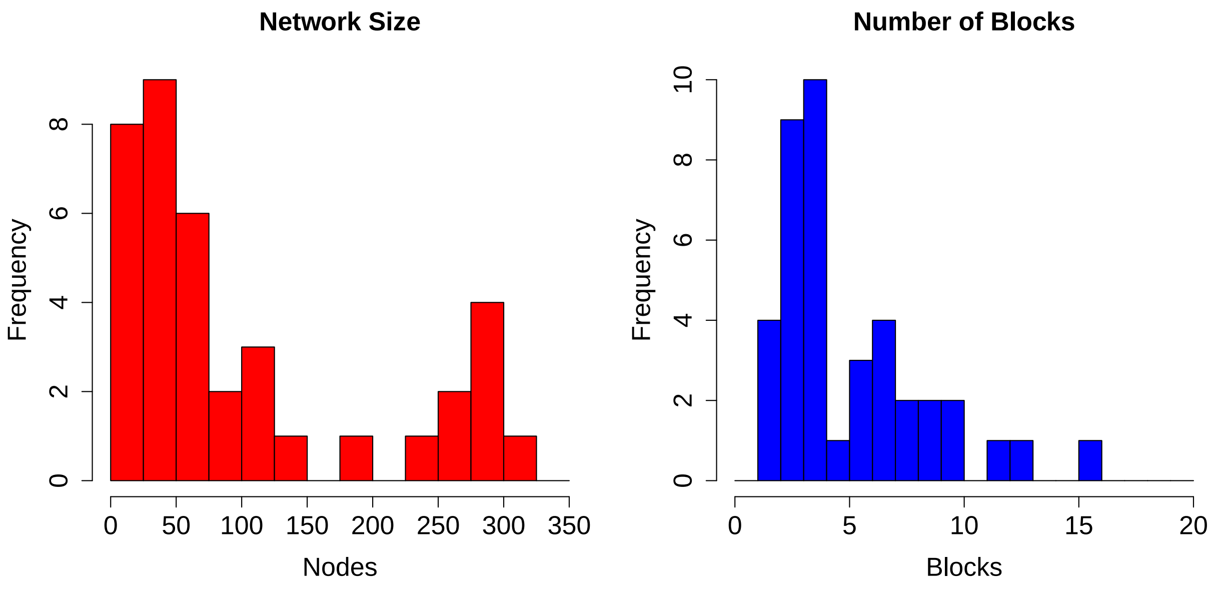

Figure 3: **Left:** Histogram of the number of nodes in a selection of networks drawn primarily from the social science and medical literature. There were additionally 4 networks with 500-2000 nodes and 4 networks with greater than 200,000 nodes in our sample of networks from the literature. **Right:** Histogram of the number of blocks found in the same selection of networks. There were additionally 3 networks with greater than 5000 blocks, and one network with 40 blocks in our sample.

we are only using spectral clustering to initialize our algorithm, we simply complete our matrix by substituting the mean tie value for each missing element in the matrix, including the diagonal entries. We also considered substituting missing values with 0's, but found that using the average tie-value resulted in better recovery of the block memberships.

3.2 Performance Measures

We consider three measures of performance. Our primary goal is to select methods that perform well at the task of model selection. Thus our first criterion is the *model selection accuracy*, defined to be the percentage of times a given method selected the true number of blocks, K. We also want to consider how well the block memberships are being estimated, whether or not a method chose the correct number of blocks.

It's possible that a method could choose the true number of blocks, but still result in block memberships that are inaccurate. Similarly, if the wrong number of blocks is chosen, the block memberships could still be similar to the true memberships. It could be that a single block was split in two or a single node was an outlier and was placed in its own block. In either case we would like to have some measure of the quality of the estimated block memberships. Following Lancichinetti and Fortunato (2009) we use the *Normalized Mutual Information* (NMI) criterion to measure the similarity between the true and estimated block memberships.

To Calculate the NMI we first need to construct a distribution for each membership assignment. To do this, suppose we have two membership assignments, V and W. We then construct random variables $V, W \in \{1, ..., K\}$ to have joint distribution

$$P(V = v, W = w) = \frac{n_{vw}}{n}, \tag{2}$$

where n_{vw} is the overlap in the two assignments. This gives us marginal distributions of $P(V = v) = n_v/n$ and $P(W = w) = n_w/n$. The mutual information is then defined to be

$$I(V, W) = \sum_{v=1}^{K} \sum_{w=1}^{K} P(v, w) \log \left(\frac{P(v, w)}{P(v)P(w)} \right). \tag{3}$$

The problem with using mutual information as a similarity criterion is that when W is a finer subdivision with more blocks, the mutual information is always equal to $H(V) := -\sum_{v=1}^{k} P(v) \log(P(v))$, the entropy of V. Thus, we obtain the same criterion no matter how fine a subdivision W is of V. To solve this problem we simply normalize by the entropy of V and W, giving us

$$I_{norm}(\mathsf{V}, \mathsf{W}) = \frac{2I(V, W)}{H(V) + H(W)}. \tag{4}$$

This normalized version equal 1 if the assignments are exactly the same, and is expected to equal 0 if the assignments came from independent distributions.

The last criterion that we will use to evaluate our methods is the *mean squared-error* (MSE) loss between the estimated tie probability matrix, $\hat{P}$, and the true tie probability matrix, P. Since the diagonal elements of P are assumed to be 0 for all of our models, the MSE loss is defined to be

$$L(P, \hat{P}) = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} (p_{ij} - \hat{p}_{ij})^2. \tag{5}$$

We use this last criterion as a measure of the overall goodness of fit to the true tie probabilities underlying the observed network.

As mentioned above, the primary goal of this simulation study is determine which methods are most accurate at recovering the true model. Preliminary simulations suggest that cross-validation performs remarkably well at the task for stochastic block models. When using NMI as the performance criterion, previous work by Lancichinetti and Fortunato (2009) showed that the infomap method performed best. However, their studied used a different set of simulation parameters related to the GN bencmarks (**?**). Finally, for the task of minimizing the loss between the estimated and true tie probability matrices, we expect the cross-validation methods to perform best. Since cross-validation provides an estimate of risk (the expected loss between P and $\hat{P}$), and we choose the model that minimizes the CV estimate of risk, we would expect the loss between that estimate and P to be relatively small.

3.3 Results

3.3.1 Comparing edgeCV and latinCV

Early results suggest that latinCV has little improvement over edgeCV. Using the full results we should be able to confirm or refute this fact.

3.3.2 Comparing edgeCV and networkCV

Early results suggest that networkCV performs better than edgeCV when the accuracy of both methods is low, and edgeCV performs better when the accuracy is high. This result seems to be related to the increased variability of the networkCV estimate of risk. If the increased variability is the culprit, the difference in performance should correspond to the region where the true model minimizes the risk.

3.3.3 Comparing edgeCV and Community Detection

Currently I plan to compare edgeCV to the infomap method. The infomap method was found to be the best community detection algorithm by Lancichinetti and Fortunato (2009).

4 Conclusions and Discussion

1. Comparison of Cross-validation methods
    - networkCV appears to perform significantly worse than edgeCV in any scenario where the accuracy of either method is above 10-20 percent.
    - latinCV appears to have similar accuracy to edgeCV. Is there a slight positive effect? If so, the procedure is simple enough to warrant using as an alternative.
2. Comparison to other methods
    - BIC - Do we still see BIC performing worse than the CV methods? If so is it actually significant?
    - Infomap - Appears to do worse at the model selection task. I still need to compare performance on NMI, which the Lancichinetti paper claimed performed best on their benchmarks. (Should I compare our benchmarks?)
    - Modularity Maximization...
3. Overall CV seems to do better at model selection.
4. I need to check how well it performs at the label assignment task using NMI
5. I would guess that it will also do the best at choosing the model that minimizes distance from the true P, given that it's essentially trying to estimate that distance, and then find the minimizer of that estimate.

References

Bickel, P. J. and Sarkar, P. (2015). Hypothesis testing for automated community detection in networks. *Journal of the Royal Statistical Society: Series B (Statistical Methodology).*

Chen, K. and Lei, J. (2014). Network cross-validation for determining the number of communities in network data. *arXiv preprint arXiv:1411.1715.*

Hoff, P. (2008). Modeling homophily and stochastic equivalence in symmetric relational data. In *Advances in Neural Information Processing Systems*. 657–664.

Lancichinetti, A. and Fortunato, S. (2009). Community detection algorithms: a comparative analysis. *Physical review E*, **80** 056117.

Pavlovic, D. M., Vértes, P. E., Bullmore, E. T., Schafer, W. R. and Nichols, T. E. (2014). Stochastic blockmodeling of the modules and core of the caenorhabditis elegans connectome.

Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, **105** 1118–1123.

Snijders, T. A. B. and Nowicki, K. (1997). Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, **14** 75–100.